

# **Designing A Prototype Model of Peer Assessment for Introductory Computer Programming Courses**

Thesis by:

Amal Khalifa Alkhalifa

In Partial Fulfilment of the Requirements for the Degree of  
Doctor of Philosophy



School of Computing, Newcastle University  
Newcastle upon Tyne, UK

(07 April 2022)



## **Abstract**

This thesis aims to provide an in-depth, contextual understanding of how peer assessments can be integrated, as a learning process, within introductory programming courses. It was motivated by exploring several difficulties first-year programmers encounter in introductory programming courses. One main issue is the difficulty of getting timely feedback from teachers. With a large class, a teacher may not be able to give instant feedback. Furthermore, students often lack the confidence to complete individual tasks, because they have not yet developed an effective internal model of a computer that they can use to construct viable knowledge. This study focuses on peer assessment and its effectiveness since peers can serve as a valuable source of instant feedback. Additionally, interactions with peers can increase students' confidence and improve their problem-solving abilities.

Students and teachers can be reluctant to use peer assessment, as programming assignments are highly practical and require sufficient knowledge to complete. To identify students' receptiveness to peer assessment, as well as teachers' attitudes to implementing such an activity, this study adopted a mixed-methods approach. Statistical analyses revealed that participants were generally positive about engaging in formative peer assessment in programming courses, but they differed in some areas which is why the validity of peer assessment among students, as well as the impact of peer assessment on their performance were also examined. The results indicate that, at a moderately medium level, first-year assessors and teachers are similar in assessment; moreover, peer assessment exerts a positive influence on student performance in programming skills.

The literature has not clarified whether programming students have specific needs regarding peer assessment. A qualitative analysis of students' expectations related to implementing peer assessment provided crucial details about students' requirements in this regard. Students noted that some elements encourage them to use peer assessment, such as clear rubrics, self-assessment, rewards for their efforts and

visual feedback. Visual feedback, particularly for either author or reviewer, is an unfamiliar aspect in the context of peer assessment; hence, it is one focus of this study.

Students are mainly concerned with the credibility of the reviewers giving feedback on their work. A Balanced Allocation algorithm has been developed to retrieve a group of reviewers to assess the work of each author, with a view to show students feedback of better quality. The peer programmer prototype website is the output that contains a group of requirements that could be considered when developing a peer assessment for programming students. This type of study is invaluable for teachers who are concerned with peer assessment for students, as it informs practice and provides guidance; thus, teachers can relate this research to their own contexts.

## **Acknowledgment**

In the name of Allah, the Compassionate, the Merciful.

All praise and thanks to Allah for providing me with knowledge, health and patience. Without his blessings and guidance, this thesis would not have been completed.

I would like to acknowledge my deep thanks and unfailing gratitude to my supervisor Dr. Marie Devlin for her guidance, support, and encouragement from the first day I visited Newcastle University until I finished writing this thesis. Her support, comments and guidance were extremely beneficial, and her advice and effort greatly improved this work. I am grateful to her for being always available, before working hours to listen to me, via Skype meetings, despite the distances between us and differences in time, to help me achieve my professional goals.

My special gratitude goes to Laura Heels and Dr. Dan Nesbitt for accepting the request to conduct the experiments on the Newcastle sample students and moderating the focus groups discussions. Further, I am grateful to all the participants in the study for giving their time and contributions to the research.

All my heartfelt and deepest gratitude go to my parents, Khalifa and Nourah, for their continuous prayers and support. Their unconditional support and sublime care gave me the peace of mind that allowed my dedication and focus on my study. Many thanks to my siblings; their unlimited love and encouragement kept me moving forward.

Special thanks to my husband, Ahmad, for being my pillar of my strength and for his support to complete this study. Many thanks to my lovely daughters, Reema and Lubna, for taking care of their Mum and keeping the house silent while I was working.

Finally, the study was financially supported by Princess Nourah bint Abdulrahman University, for which I am thankful.

## Declaration

All work contained within this thesis represents the original contribution of the author. This study has given rise to three publications which are listed below.

1. Amal, K., Al-Khalifa and Marie Devlin. 2020. *Evaluating a Peer Assessment Approach in Introductory Programming Courses*. In United Kingdom & Ireland Computing Education Research conference. (UKICER '20). Association for Computing Machinery, New York, NY, USA, 51–58. DOI:<https://doi.org/10.1145/3416465.3416467>
2. Amal Alkhalifa and Marie Devlin. 2021. *Student Perspectives of Peer Assessment in Programming Courses*. United Kingdom and Ireland Computing Education Research conference. Association for Computing Machinery, New York, NY, USA, Article 8, 1–7. DOI:<https://doi.org/10.1145/3481282.3481292>
3. Amal Alkhalifa, Marie Devlin, and Mona Alkhattabi. 2022. *Matching Authors and Reviewers in Peer Assessment Based on Authors' Profiles*. *Journal of Information Technology Education: Innovations in Practice*, Accepted.

## Table of Contents

Abstract.....	i
Acknowledgment.....	iii
Declaration.....	iv
Table of Contents.....	v
List of Figures .....	ix
List of Tables .....	xi
List of Abbreviations.....	xiii
Chapter 1. Introduction .....	1
1.1 Problem overview.....	1
1.2 Peer assessment for learning.....	4
1.3 Research motivation.....	7
1.4 Research questions.....	8
1.5 Research objectives .....	8
1.6 Research significance .....	8
1.7 Thesis structure.....	9
Chapter 2. Literature review.....	12
2.1 Introduction .....	12
2.2 Why students drop introductory programming courses .....	13
2.3 Collaborative strategies for learning programming .....	18
2.3.1 Common collaborative learning programming strategies .....	19
2.4 Peer assessment.....	22
2.4.1 Key theoretical perspectives in peer assessment .....	23
2.4.2 Benefits of peer assessment for programming students .....	28
2.4.3 Barriers to using peer assessment in programming courses .....	31
2.4.4 Types of assessment.....	34
2.4.5 Methods of peer assessment.....	39
2.4.6 Comparing peer assessments with tutor assessments .....	45
2.4.7 Impact of peer assessment on learning to program .....	46
2.4.8 Peer assessment in UK and KSA higher education .....	48
2.4.9 Quantitative and qualitative studies of peer assessment .....	50
2.4.10 Evaluating common peer assessment systems .....	52
2.4.11 Controversial issues in peer assessment.....	58
2.5 Summary.....	60
Chapter 3. First phase of the study .....	62

3.1 Introduction.....	62
3.2 Mixed-methods approach .....	62
3.2.1 Explanatory design.....	64
3.3 Research design.....	65
3.3.1 Ethics procedures .....	66
3.3.2 Determining the target samples .....	68
3.4 First phase of the research .....	68
3.4.1 Questionnaire method .....	68
3.4.2 Interview method.....	75
3.4.3 Experimental method .....	77
3.5 Validity and reliability of the methods used in the first phase.....	93
3.6 Summary .....	94
Chapter 4. Results of the first phase .....	96
4.1 Introduction.....	96
4.2 Results from the questionnaires .....	96
4.2.1 Demographic data: Students and teachers' questionnaires .....	97
4.2.2 Descriptive data: Benefits and challenges of peer assessment .....	99
4.2.3 Correlation between awareness of benefits and fear of challenges .....	104
4.2.4 Comparison between students and teachers in the benefits and challenges .....	104
4.2.5 Descriptive data: How to apply peer assessment.....	105
4.2.6 Comparison between students and teachers in methods of applying peer assessment.....	108
4.2.7 Open-ended questions .....	109
4.2.8 Reliability and validity of scale.....	114
4.3 Results from the interviews .....	115
4.3.1 Demographic data .....	116
4.3.2 Presentation of the key themes .....	117
4.4 Results from experimental method .....	126
4.4.1 Demographic data: Pilot-experiment method .....	127
4.4.2 Determining the optimal marking guide form.....	128
4.4.3 Marking guide development .....	131
4.4.4 Demographic data: Pseudo-experiment method .....	133
4.4.5 Correlation between students' assessment and teacher assessment....	134
4.4.6 Impact of peer assessment activity on students' performance .....	135
4.4.7 Students' preferences regarding peer assessment feedback.....	136
4.4.8 Open-ended questions .....	137
4.4.9 Reliability and validity of scale.....	140
4.5 Summary .....	140



Chapter 5. Second phase of the study .....	142
5.1 Introduction .....	142
5.2 Second phase of the research .....	142
5.2.1 Initial design of the prototype .....	143
5.2.2 User-centred design.....	145
5.2.3 Focus group method.....	146
5.2.4 Interview method.....	154
5.3 Validity and reliability of the methods used in the second phase .....	156
5.4 Summary.....	158
Chapter 6. Results of the second phase .....	159
6.1 Introduction .....	159
6.2 Results from focus groups.....	159
6.2.1 Demographic data.....	160
6.2.2 Procedures of collecting and analysing data.....	161
6.2.3 Presentation of the main key themes.....	164
6.3 Result from interviews .....	172
6.3.1 Demographic data.....	172
6.3.2 Procedures of collecting and analysing data.....	173
6.3.3 Presentation of the main key themes.....	174
6.4 Visualising peer feedback .....	180
6.5 Final design of the prototype .....	182
6.5.1 Requirement analysis .....	183
6.5.2 Prototype diagrams.....	186
6.5.3 Design of the Peer Programmer prototype .....	194
6.6 Summary.....	220
Chapter 7. Matching authors and reviewers .....	221
7.1 Introduction .....	221
7.2 Peer assessment issue in this study .....	221
7.3 Multiple-criteria decision making .....	223
7.4 Balanced Allocation algorithm description.....	224
7.5 Pseudocode of the algorithm.....	229
7.6 Experimental results related to the algorithm .....	229
7.6.1 Algorithm implementation on a real dataset.....	229
7.6.2 Algorithm implementation on the mock-up dataset.....	233
7.7 Evaluation of the algorithm.....	236
7.7.1 Procedure of data collection .....	237

7.7.2 Data analysis.....	238
7.7.3 Evaluation results.....	238
7.8 Discussion of matching author-reviewers .....	242
7.9 Summary .....	245
Chapter 8. Discussion of findings.....	247
8.1 Introduction.....	247
8.2 Discussion of the findings in relation to the research questions .....	247
8.2.1 The perceptions of students and teachers on peer assessment .....	247
8.2.2 Accuracy and impact of peer assessment on students' performance ....	257
8.2.3 Requirements and critical issues during implementing peer assessment .....	260
8.2.4 Integrating peer assessment into introductory programming courses ....	267
8.3 Summary .....	276
Chapter 9. Conclusion.....	278
9.1 Summary of the study and main findings.....	278
9.2 Research contributions .....	281
9.3 Research limitations .....	283
9.4 Future directions for research.....	286
References.....	288
Appendix A: Ethics approval .....	308
Appendix B: Consent form for participation with focus group.....	309
Appendix C: An example of students' assessment .....	310
Appendix D: An example of entering data into SPSS.....	311
Appendix E: An example of students' feedback .....	312
Appendix F: Question form distributed in the first focus group .....	313
Appendix G: Question form distributed in the third focus group .....	314
Appendix H : An example of open coding .....	316

## List of Figures

Figure 2.1. PeerScholar system.....	54
Figure 2.2. PeerGrade system.....	55
Figure 3.1. Research design of the current study .....	66
Figure 3.2. Random sample answers to assess in the pilot-experiment .....	85
Figure 3.3. Random sample answers to assess in the pseudo-experiment.....	89
Figure 3.4. Visual feedback .....	90
Figure 3.5. Written feedback.....	90
Figure 3.6. Questions about the visual feedback .....	90
Figure 4.1. Frequencies in benefits of peer assessment-students' responses.....	100
Figure 4.2. Frequencies in challenges of peer assessment-students' responses..	101
Figure 4.3. Frequencies in benefits of peer assessment-teachers' responses .....	102
Figure 4.4. Frequencies in challenges of peer assessment-teachers' responses..	103
Figure 4.5. Students' and teachers' opinions about elements of peer assessment	107
Figure 4.6. Reasons encourage participants to perform peer assessment.....	111
Figure 4.7. Hindrance to using peer assessment from participants' perspectives..	112
Figure 4.8. Teachers' perceptions of essential criteria in the marking scheme.....	114
Figure 4.9. Comparison between PNU and Newcastle University scores .....	131
Figure 4.10. Sample of initial marking scheme form .....	132
Figure 4.11. Students' negative comments in open-ended questions.....	139
Figure 4.12. Students' positive comments in open-ended questions .....	139
Figure 5.1. Initial design of the prototype .....	144
Figure 6.1. Activity diagram of Peer Programmer prototype .....	168
Figure 6.2. Progress chart .....	170
Figure 6.3. Grade details chart .....	178
Figure 6.4. Reviewers' choices table .....	178
Figure 6.5. Colouring in details grade chart .....	179
Figure 6.6. Visualising peer assessment data (Ueki & Ohnishi, 2016).....	181
Figure 6.7. Use case diagram of the Peer Programmer prototype.....	187
Figure 6.8. Sequence diagram to add and manage the assignment.....	189
Figure 6.9. Sequence diagram of the create phase .....	190
Figure 6.10. Sequence diagram of the review phase.....	191
Figure 6.11. Sequence diagram of the feedback phase.....	192
Figure 6.12. General description of Peer Programmer prototype .....	193

Figure 6.13. Login page .....	195
Figure 6.14. Courses page in the teacher's account .....	196
Figure 6.15. Create question page .....	197
Figure 6.16. Build a marking scheme page .....	198
Figure 6.17. Assignment settings page .....	199
Figure 6.18. Summary of uploading assignment page .....	200
Figure 6.19. The assessment page in the teacher account.....	201
Figure 6.20. Teacher's feedback page .....	202
Figure 6.21. Courses page in student's account .....	203
Figure 6.22. Tasks list page .....	204
Figure 6.23. Assignment question page .....	205
Figure 6.24. Writing code page .....	206
Figure 6.25. Self-assessment page.....	207
Figure 6.26. Submission page in create phase .....	208
Figure 6.27. Matching process pop-up message .....	209
Figure 6.28. The peer's work page.....	210
Figure 6.29. Peer assessment page .....	211
Figure 6.30. Submission page in review phase .....	212
Figure 6.31. Feedback page .....	213
Figure 6.32. Interactive charts.....	214
Figure 6.33. Progress page.....	214
Figure 6.34. Rating the reviewer's page.....	216
Figure 6.35. Resubmit page .....	217
Figure 6.36. Instruction page.....	218
Figure 6.37. Who we are page .....	219
Figure 6.38. Notification page .....	219
Figure 7.1. Architecture of the Balanced Allocation algorithm .....	225
Figure 7.2. Algorithm description.....	228
Figure 7.3. Score distribution between authors and reviewers in real dataset .....	233
Figure 7.4. Score distribution between authors and reviewers in mock-up dataset.....	236
Figure 7.5. Students' distribution of assigning the ability level .....	240
Figure 7.6. Teachers' overview of the matching method .....	242
Figure 7.7. Students' overview of the matching method.....	242

## List of Tables

Table 3.1. Programming students' questionnaire about peer assessment.....	71
Table 3.2. Programming teachers' questionnaire about peer assessment.....	72
Table 3.3. Programming teachers' interview questions about peer assessment.....	77
Table 3.4. Marking scheme used in the pilot-experiment method .....	82
Table 3.5. Rubric used in the pilot-experiment method .....	84
Table 4.1. Distribution of students' demographic variables.....	98
Table 4.2. Distribution of teachers' demographic variables.....	99
Table 4.3. A Mann-Whitney U test results.....	109
Table 4.4. Interviewees' demographic variables .....	117
Table 4.5. Theme 1: Some difficulties of first-year students .....	118
Table 4.6. Theme 2: Teachers' attitudes towards peer assessment.....	120
Table 4.7. Theme 3: Strategies preferred to carry out the peer assessment .....	123
Table 4.8. Theme 4: Utilizing the data collected from the peer assessment.....	125
Table 4.9. Demographic data of pilot-experiment method .....	128
Table 4.10. Comparison between the means of the rubric and marking scheme ..	129
Table 4.11. Comparison between the means of categories .....	130
Table 4.12. Demographic data of pseudo-experiment method .....	133
Table 4.13. The effect of peer assessment on scores .....	135
Table 4.14. Students' preferences regarding peer feedback .....	136
Table 4.15. Impact of feedback on peer assessment activity.....	137
Table 5.1. Discussion guide for the first focus group .....	148
Table 5.2. Discussion guide for the second focus group.....	150
Table 5.3. Discussion guide for the third focus group .....	152
Table 5.4. Interview questions to evaluate the prototype .....	156
Table 6.1. Distribution of focus group participants' demographic variables .....	161
Table 6.2. Theme 1: Perceptions of peer assessment.....	165
Table 6.3. Theme 2: Credibility of peer assessment.....	167
Table 6.4. Theme 3: Structure of the prototype .....	169
Table 6.5. Theme 4: Appearance of the prototype.....	171
Table 6.6. Interviewees' demographic variables .....	172
Table 6.7. Theme 1: Teachers' perceptions of peer assessment .....	175
Table 6.8. Theme 2: Teachers' concerns about peer assessment .....	176
Table 6.9. Theme 3: Teachers' view about the structure of the prototype .....	177

Table 6.10. Theme 4: Teachers' view about appearance of the prototype.....	179
Table 6.11. Description of the login page.....	195
Table 6.12. Description of courses page.....	196
Table 6.13. Description of create question page.....	197
Table 6.14. Description of build a marking scheme page.....	198
Table 6.15. Description of assignment settings page.....	199
Table 6.16. Description of uploading assessment page.....	200
Table 6.17. Description of the assessment page in the teacher account.....	201
Table 6.18. Description of teacher's feedback page.....	202
Table 6.19. Description of courses page in student's account.....	203
Table 6.20. Description of the tasks list page.....	204
Table 6.21. Description of the assignment question page.....	205
Table 6.22. Description of writing code page.....	206
Table 6.23: Description of self-assessment page.....	207
Table 6.24. Description of the submission page in create phase.....	208
Table 6.25. Description of the peer's work page.....	210
Table 6.26. Description of peer assessment page.....	211
Table 6.27. Description of the submission page in review phase.....	212
Table 6.28. Description of the feedback page.....	213
Table 6.29. Description of progress page.....	215
Table 6.30. Description of rating page.....	216
Table 6.31. Description of resubmit page.....	217
Table 6.32. Description of the instruction page.....	218
Table 7.1. Weights of attributes.....	231
Table 7.2. Algorithm implementation on a real dataset.....	232
Table 7.3. Algorithm implementation on the mock-up dataset.....	234
Table 7.4. Discussion questions to evaluate the algorithm.....	237
Table 8.1. A structured form for integrating peer assessment.....	270

## **List of Abbreviations**

ALiC	Active Learning in Computing
CS1	Introductory Programming Course
CSE	Computer Science Education
ELT	Experiential Learning Theory
HCI	Human Computer Interaction
HESA	Higher Education Statistics Agency
MCDM	Multiple-Criteria Decision Making
MKO	More Knowledgeable Other
PCR	Pedagogical Code Review
STEM	Science, Technology, Engineering, and Mathematics
UCD	User-Centred Design
UML	Unified Modelling Language
WSM	Weighted Sum Model
ZPD	Zone of Proximal Development





# Chapter 1. Introduction

## 1.1 Problem overview

The high demand for computer science knowledge in various professions has led to a significant increase in enrolment in computer science courses in higher education institutions. Hundreds of thousands of students annually join computer programming courses in various countries. According to the United Kingdom's (UK) Higher Education Statistics Agency (HESA), for example, computer science programs saw the largest percentage increase in enrolment for first-year undergraduate students between 2015/16 and 2016/17. It is also evident that many disciplines are becoming increasingly dependent on large amounts of data which require good computational skills. However, the UK industry reports a lack of computer science graduates (Brown *et al.*, 2013) suggesting significant non-completion rates in the computer sciences.

Course failure, dropouts, and irregular attendance are common in computer science, particularly in introductory computer programming courses (Luxton-Reilly, 2016). According to HESA's latest statistics, in the academic year 2017/18, 9.8% of computer science undergraduates dropped out before completing their degrees (Celepkolu and Boyer, 2018). Another study illustrates that the pass rate, especially in introductory programming courses, is only 67.7% (Celepkolu and Boyer, 2018). Hawlitschek *et al.* (2020) mentioned that over 30% of students drop out of introductory computer programming courses and suggest that programming courses are among the most complex challenges students face. When students struggle and fall behind their peers, they may drop out of the discipline because they feel hopeless. Besides, students who drop out of their degree course suffer negative labour market outcomes and greater marginalization (Sosu and Pheunpha, 2019). But drop-outs do not only affect students but also teachers and institutions more generally. A dropout rate may cast teachers in a negative light; this is because potential students can construe dropout rates as a sign of deficient teaching and teachers' support (Sosu and Pheunpha, 2019). It also affects the

institution because the individual blocks a university place that another student could have taken.

High dropout rates are not only common in the computer sciences and in introductory programming courses, but also in other scientific disciplines, especially at the beginner stage. The Higher Education Research Institute in the United States, for instance, states that more than half the students who enrol in Science, Technology, Engineering, and Mathematics (STEM) disciplines leave these disciplines before graduation (Hurtado, Eagan and Chang, 2010). According to Dagley *et al.* (2016), the dropout rate in the first year in STEM disciplines is 27%; in the second year it is 16%; and in the third year it is 12%. This indicates that dropout is at its highest in the first year of higher education. According to Sithole *et al.* (2017), STEM students leavers are often high-performing students who would have made worthy additions to the workforce had they completed their studies. The same arguably holds for computer science students. Therefore, one of the most critical things to keep in mind when discussing dropout rates is this question: What should instructors do to avoid students leaving their degrees without graduating? However, the focus should be on understanding students' issues and promoting successful completion of introductory programming courses, not only on preventing dropouts.

The phenomenon of dropout is complex as it is a multifaceted phenomenon influenced by a number of diverse variables (Lázaro Alvarez, Callejas and Griol, 2020). In order to address the problem of dropout, early identification of programming students' difficulties is crucial as a first step in order to ascertain suitable interventions, since a better understanding of causes allows the development of more effective intervention. Scholars have made efforts to do so; for example, failure and dropout rates among programming students have been significant research topics in computer science education (CSE) for a long time (Bennedsen and Caspersen, 2006; Watson and Li, 2014; A. Petersen *et al.*, 2016). Some scholars have studied the studying habits of students inside and outside the classroom (e.g., reading the textbook, and working on problem-solving) to find reasons of dropout (Chinn *et al.*, 2010) and some scholars have investigated factors influencing

student dropout rates (e.g. personal value, and effort) (Rountree *et al.*, 2004; Pappas, Giannakos and L. Jaccheri, 2016). From these reasons, other scholars have investigated some interventions that might affect students' decision to remain in programming courses, such as revisiting the expected norms for introductory programming and creating a more motivational environment (Luxton-Reilly, 2016); using educational games to learn to program (Ibrahim *et al.*, 2011); using an active learning approach for computing education (Hundhausen, Agrawal and Agarwal, 2013); or developing programs that provide automatic and instant feedback to a student's programming efforts (Venables and Haywood, 2003). These studies show that CSE scholars have focused heavily on finding and addressing the causes of dropout.

Active learning and collaborative learning techniques are pedagogical approaches associated with academic success in general (Hundhausen, Agrawal and Agarwal, 2013). Student engagement in the learning process underlies the most promising dropout avoidance programs (Reschly, 2020). Active and collaborative learning develops coping skills (e.g., confidence, persistence, and strong interest in the discipline) and alleviates the high dropout rate, and are two of the seven guiding principles for good practice in undergraduate education (Gonzalez, 2006). Additionally, active and collaborative learning increases students' performance by allowing them to exchange ideas, enhances their critical thinking skills, and helps them construct knowledge (Boudia, Bengueddach and Haffaf, 2019). According to Gonzalez (2006), 70% of first-year students who had an active learning experience in their introductory programming courses ended up getting a pass grade, with only 10% dropping out. The dropout rate for those that took a regular introductory programming course without an active learning experience was 25% with a pass rate of 44%. In this vein, multiple collaborative teaching strategies have been applied in introductory programming courses, such as Pedagogical Code Review (PCR) (Hundhausen, Agrawal and Agarwal, 2013), pair programming (Lui and Chan, 2006), and peer assessment (Sitthiworachart and Joy, 2004). Of these options, this study has selected peer assessment as its focus of investigation.

## 1.2 Peer assessment for learning

Peer assessment involves learners assessing each other's performance against a set of criteria prepared by an instructor or agreed upon between the instructor and the learners. Topping (1998, p. 250) defines peer assessment as considering "the amount, level, value, worth, quality or success of learning of peers of similar status". Peer assessments engage students in two roles: 'author' and 'reviewer'. As a reviewer, a student reviews his/her peers' work and contributes feedback; as an author, the student receives, reads and acts upon reviewers' feedback to enhance his/her own work (Li and Gao, 2016). Peer assessment is assumed to lead to more meaningful, in-depth learning because it engages students in the learning process and generates deep learning (Lynch, McNamara and Seery, 2012), critical thinking (Li *et al.*, 2016), increased academic performance (Double, McGrane and Hopfenbeck, 2020), knowledge gains (Li *et al.*, 2020), verbal communication and teamwork skills (Adachi, Tai and Dawson, 2018), and it allows students to develop professional skills used in the workplace (Ecclestone and Pryor, 2003). Thus, peer assessment is a valuable approach that promotes active and collaborative learning, and it could help to increase first-year students' retention.

Some research indicates that 'peer assessment' and 'peer review' refer to the same process (García and Pardo, 2010; Luxton-Reilly, Lewis and Plimmer, 2018; Indriasari, Luxton-Reilly and Denny, 2020). However, other studies differentiate between the two terms; Søndergaard and Mulder (2012), for example, refer to peer assessment as a process in which students assess other peers' work based on explicit criteria, provide feedback in the process and sometimes mark peers' work to become part of the overall result. In contrast, peer review indicates feedback which is used purely formatively (Søndergaard and Mulder, 2012). Other scholars (e.g., Luxton-Reilly, Lewis and Plimmer, 2018; Sun *et al.*, 2019) highlight that peer review is common in industrial and open-source software development and includes team leaders and developers. The leader typically selects reviewers to write suggestions on the code produced by one or more developers on the same team to identify bugs and improve their written code. Bacchelli and Bird state that the primary goal of peer review is to produce a better product, while peer assessment

aims to help students learn about a piece of code and improve higher-order cognitive skills (2013). In line with this conception, the term peer assessment was selected throughout this thesis since this research focuses on improving students' learning performance and skills.

Programming assignments are a useful way to help students' understanding of content knowledge; engaging peer assessment in programming assignments then empowers students to develop programming knowledge and improve their programming skills (Hwang, Liang and Wang, 2016; Wang *et al.*, 2017). Programming students need to see other solutions to the programming issues they are assigned, which allows them to understand their assignments better and enables them to consider their solutions' strengths and limitations and to compare their work with other solutions. Presenting only one model answer to students does not broaden their perspectives, and it makes it difficult for them to determine how to improve their own solutions (Reinholz, 2016). When students engage in peer assessment activities, they have a chance to think about their own work while judging their peers' efforts, which encourages subsequent self-regulation. Furthermore, when students provide feedback to their colleagues, they feel more confident about their own programming skills. As a result, peer assessment in the form of constructive feedback can serve the purpose of activating students as instructional resources for one another as well as encourage them to take ownership of their own learning (Shui Ng, 2017). Peer assessment can thus motivate students to learn better in introductory classes.

Some instructors believe that assessing programming assignments is challenging, as they tend to be highly practical in nature, and as different constructs and logics can produce the same output (Stegeman, 2014); this may cause inconsistency in assessments, particularly in peer assessments, when numerous peers assess the same assignment. Some instructors also believe that first-year programmers do not have enough knowledge to complete qualified assessments (Indriasari, Luxton-Reilly and Denny, 2020). While the instructor is the most authoritative source of professional knowledge and assessment, the instructor is not the *only* source of knowledge in a class. Peer feedback then does not

conflict with the teacher's role in assessing individual students but it enhances learning effectiveness (Panadero and Brown, 2017), as this thesis suggests. Ultimately, instructors design and facilitate learning activities such as peer assessments to help students reduce the gap between where they are and their required goals. In short, to be successful, peer assessment must be effectively implemented and managed, and teachers should focus on how they can best employ peer assessment in their individual programming curricula.

This thesis aims to design a prototype based on a user-centered design. Literature often focuses on the instructor and/or a program-centered approach without considering the student's approach (e.g., students' needs and interests from the application) (Kahraman, 2010). For example, Al-Sa'di and McPhee (2021) conducted a systematic literature review that built educational applications based on a student-centered approach, whereby none of the applications were used for first-year programming students. However, it is important to include students' voices while designing a learning prototype/application, as students are the central part of the learning process. By considering the students' perspectives during the design process, the designed product could increase usage, success, and performance (Kahraman, 2010). Furthermore, at the end of the design process, the student can use the final product with minimum effort and optimum efficiency (Kahraman, 2010). Therefore, this thesis considers students to be the primary stakeholder of this learning process as they are active participants in the peer assessment and potential beneficiaries of it. Teachers' perspectives are also significant as they give insight into how teachers conceptualise to implement the activity; at what stage of the course peer assessment should be implemented; and how this information could improve the quality of the learning process. With this awareness in mind, students' and teachers' perceptions during designing the prototype form an important part of this study.

This study adopts a mixed-methods approach. Research data were generated by computer programming students and their teachers. The aim of this thesis is to provide an in-depth, contextual understanding of how peer assessments as a learning process can be integrated within introductory programming courses. This research also seeks to

design a prototype website that suggests peer assessment activities that teachers and students can then utilise.

### **1.3 Research motivation**

The researcher is a computer science instructor at Princess Nourah bint Abdulrahman University, and, so far, it has been observed that peer assessments are only utilised in graduation project courses. This course allows the student to work on an authentic and practical computing project, and it must be taken by a small group of students (4-5) who work together. The graduation project course uses peer assessments as part of the final grades because students work as a team and must assess each other as part of the assignment, not for learning purposes. Other courses in the computer science department are altogether unfamiliar with peer assessment. At Newcastle University, lecturers use peer assessment in courses based on projects and teamwork (e.g., Software Engineering module), or in courses in advanced years rather than in introductory programming courses for learning purposes or to decide the contribution of teammates (e.g., (Devlin, 2015)). Having noticed the lack of use of peer assessment in introductory programming courses, particularly in Saudi Arabia, and having become aware of the difference in students' and teachers' views on this activity - despite its many benefits especially in stimulating active learning and collaborative learning - motivated the researcher to investigate how peer assessments can be utilised in introductory programming curricula. To this end, the study's goal was to give computer programming students the chance to review each other's code, write and read comments and see how their peers address the same problems. Also, students' and teachers' requirements and their perceptions of peer assessment must be considered to build a high-quality activity. The study postulates that assessment methods that make students the main focus may ultimately create more meaningful and higher quality learning environments, and first-year programmers desperately need this kind of attention. Therefore, to support students with opportunities to learn from one another, to increase their retention in introductory programming courses, and to enhance their confidence and learning experiences, the study's primary focus was a peer assessment activity in introductory programming courses.

## **1.4 Research questions**

The main goal is to create a contextual understanding of how peer assessments can be integrated into introductory programming courses. The research questions guiding this thesis are as follows:

1. How do programming students and teachers perceive peer assessment in introductory programming courses?
2. Are first-year students who participate in peer assessment accurate and more likely to perform better on programming skills than those who do not?
3. What are student expectations and critical issues related to implementing peer assessment in introductory programming courses?
4. How can peer assessment, as a learning process, be represented in introductory programming courses?

## **1.5 Research objectives**

1. To identify the perceptions of programming students and teachers toward implementing peer assessment in introductory programming courses.
2. To investigate the accuracy of first-year students' and the impact of peer assessments (if any) on first-year programmers' performances.
3. To determine students' needs in terms of peer assessment and to ascertain the problems with which they are concerned in peer assessment.
4. To develop a prototype website with suggestions for peer assessment activities that meets programming students' and teachers' functional requirements and avoids their concerns.

## **1.6 Research significance**

The significance of this research lies in its attempt to show programming teachers the effectiveness and relevance of peer assessment in introductory programming courses with the view to encourage them to promote integrating peer assessment in their courses. The findings of this research offer teachers and practitioners insight into the viability of peer assessment in introductory programming courses for first-year programming



students. Moreover, the findings provide empirical evidence in relation to the impact of peer assessment on students' programming skills. The assumption this study is based on – in line with existing research on peer assessment as indicated above – is a) that evaluating peer assessment will increase teachers' awareness of the effectiveness of peer assessments; and b) that this activity can potentially support the achievement of learning goals. Moreover, it helps increase first-year students' engagement in the learning process, thus avoiding their dropout from introductory programming courses. The Peer Programmer prototype website, the main output of this study, identifies the functional requirements of students and teachers in peer assessment. Hence, designers of applications understand stakeholders' requirements regarding peer assessments early on, thus minimizing confusion when it is time to begin coding, and then save time and money when developing peer assessment applications. In addition, and because research on this topic is limited, this study contributes to the academic literature on how teachers can integrate peer assessment activities into introductory programming courses, at what point in the program they want to introduce it, how they want to implement it, and what technologies could support peer assessment.

## **1.7 Thesis structure**

This thesis comprises nine chapters. The Introduction provides an overview of the research problem, states the research questions, and outlines the main objectives of this study.

Chapter Two provides an overview of difficulties students face in fundamental programming courses which might cause them to drop out. The chapter investigates how collaborative learning strategies might reduce these problems. It also discusses the concept of peer assessment, including its benefits, barriers, types, methods, and impact on students' learning, particularly for first-year students. The chapter draws on qualitative and quantitative studies of peer assessment for first-year students to find appropriate methods for collecting data for this study. It evaluates the most popular peer-assessment systems, which then informs the peer assessment prototype for first-year programming students that is developed as part of this study. Finally, the chapter highlights relevant

research gaps and concerns in the literature, and addresses how this study considers these gaps.

This study employs a mixed-methods approach to explore the most efficient and effective ways for integrating a peer assessment activity in programming courses. Chapter Three describes this approach and outlines the rationale for using it. This chapter also outlines the research design of this study. The study is divided into two phases: the first phase focuses on data collection through quantitative methods; while the second phase focuses on data collection through qualitative methods. Chapter Three then presents the first phase and its methods (questionnaires and experiments). It describes how each method was applied to gather data, and introduces the data analysis tools used to analyse the collected data. The chapter also considers the reliability and validity of these methods.

Chapter Four analyses and presents the results obtained from the first phase of the study; it includes data gathered through questionnaires and experimental methods and describes the demographic data of the participants. The chapter summarises significant findings related to the first two research questions. It outlines teachers' and students' perceptions of different aspects of peer assessments (benefits, barriers, and key elements for implementation). It also evaluates first-year students' performance to validate the peer assessment activity and measures its impact on students' programming performance.

Chapter Five describes the second phase of the study. It outlines the initial prototype that was built as a result of the first phase. It then describes the qualitative methods that were used to evaluate the prototype, including focus group discussions and interviews. The reliability and validity of these methods are also considered in this chapter.

Chapter Six describes the findings of the study's second phase. It describes the demographic data of the participants of the focus groups and interviews as well as key themes that have emerged in the data analysis. The chapter outlines significant findings related to the last two research questions; it includes functional requirements of students and teachers toward peer assessment and addresses their issues. It also includes diagrams for designers (e.g., use case and sequences diagrams) who might want to

develop a peer assessment activity. Furthermore, the chapter presents the final version of the prototype website developed specifically as part of this study and describes some students' and teachers' experiences using the website.

Chapter Seven addresses a problem with peer assessment that students have identified: the credibility of peer feedback. This chapter presents an algorithm matching authors and reviewers based on the author's needs in a peer assessment scenario. The algorithm retrieves a set of reviewers for each author in order to provide personalised feedback for the selected author. The chapter investigates the algorithm's efficiency using different samples of datasets, and evaluates it by collecting the students' and teachers' perspectives using focus groups and interviews.

Chapter Eight discusses and evaluates the study's main findings. The chapter addresses and summarises significant findings in relation to each research question, connecting those to previous studies in the same field. It also provides a structural model for teachers and practitioners to follow when implementing peer assessment in introductory programming courses with first-year students.

The Conclusion summarises the study's key findings, highlights the study's main contributions, discusses limitations of the study, and offers suggestions for future research.

## Chapter 2. Literature review

### 2.1 Introduction

Identifying the problems first-year students encounter during their studies, factors in reducing dropout rates, and factors contributing to successful learning outcomes are areas that have long interested researchers and teachers in computer science (Venables and Haywood, 2003; Bennedsen and Caspersen, 2006; Watson and Li, 2014; A. Petersen *et al.*, 2016; Pappas, Giannakos and L. Jaccheri, 2016). The purpose of this chapter is to review the relevant research on issues related to first-year students, and to explore peer assessment as a suggestion to retain these students and reduce their problems in the early stages of introductory programming courses. Thus, this chapter first explores problems students face in introductory programming courses, especially issues that teachers can provide solutions for by adopting specific teaching strategies. Next, the chapter outlines common collaborative learning strategies popular in introductory programming courses, as an effective learning strategy that supports first-year students and offers solutions to problems students encounter. These collaborative learning strategies are rooted in social theories, which this chapter discusses, and how they provide mechanisms for constructing knowledge inside a learner's mind. The chapter describes what a peer assessment strategy is, its definition, types, methods, benefits, and barriers. An overview of peer assessment gives the topic's key elements and identifies the most important information. The chapter then explores the accuracy of peer assessment and its impact on the students' learning process to investigate if this type of activity fits in introductory programming courses. Besides, the chapter reviews quantitative and qualitative studies that concern peer assessment in introductory programming courses to summarise students' and teachers' needs in peer assessment and establish the research gaps. The chapter also evaluates common automated tools that are good examples for designing peer assessment prototypes. Finally, the chapter outlines controversial issues in peer assessment to determine the research position from these issues.

## **2.2 Why students drop introductory programming courses**

Identifying proper solutions to a problem involves knowing the nature of the problem. Taking action without identifying what factors contribute to the problem can result in misdirected efforts and a waste of time and resources. Therefore, finding issues enables teachers to identify and make use of opportunities in the learning environment and exert (some level of) control over these issues. Teachers must understand why first-year programming students are dropping out of their studies. What can teachers do to support first-year students? Are there successful learning strategies for first-year students? And how can teachers apply such learning strategies? Therefore, the goal of the following sections is to detail some of the difficulties first-year programmers face and to discuss what an effective learning environment and experience for students might look like. But first, the main reasons that may cause dropping out of programming courses must be considered, as discussed in the literature.

### **Difficult subject**

Programming is inherently hard to learn. The programming task is a complex cognitive task (Luxton-Reilly, 2016), and it is a new subject to many students (Rahmat *et al.*, 2012). According to Kinnunen and Malmi (2008), from the students' perspectives, the most popular reason to drop out of the course is the difficulty of the topics making the content of the introductory programming course a source for stressful situations. Teachers have determined a list of topics that they find hard to teach, which implies that those topics are hard for first-year students to learn (Dale, 2006). However, listing hard topics is not enough; it is important to understand first-year students' knowledge structures and their cognitive processes as they are viewed as an initial point for enhancing learning, which in turn is a way to reduce the dropout rate.

Some scholars attribute the difficulty of the subject to the method of learning. Learning to program can generally be split into two main categories: those with a psychological/educational perspective, and those with a software engineering perspective (Robins, Rountree and Rountree, 2003). For first-year students, learning to program is

usually focused on a psychological/educational perspective, the knowledge and skills required to program and program generation (Robins, Rountree and Rountree, 2003). This means, for example, a closer focus on how a 'for-loop' works, and less focus on programming strategies that define the use and apply the knowledge of using a 'for-loop' suitably in a program. This leads to a 'line by line' programming approach instead of using meaningful program 'chunks' or structures. This arguably limits new students to surface knowledge which might cause a lack of detailed mental models and might ultimately lead to their failure to apply relevant knowledge. The software engineering perspective is more often used with professional or experienced students who often work in teams and focus on comprehension of the program (Robins, Rountree and Rountree, 2003). Therefore, early programming learning should include the basics of good software engineering practice. Using learning strategies that employ software engineering skills (e.g., analysing programs, testing, and debugging) at early stages could help students to improve their technical skills, comprehension, and critical-thinking skills.

### **Timely support**

Students become frustrated if they do not receive support, and frustration is often associated with students' desire for help to be promptly available (Venables and Haywood, 2003; Andrew Petersen *et al.*, 2016). Even when students feel that the level of support and quality of materials they have access to are suitable, a lack of help instantly increases frustration and study time (Andrew Petersen *et al.*, 2016). Students require details of the errors they made (Wang *et al.*, 2011) to engage and improve their programming skills. They often want to know immediately (Venables and Haywood, 2003) how much progress they have made in the programming, and when students wait for a long time for teacher feedback, they tend to engage with it less once it is received (Carless, 2013). But teachers struggle to provide on-time feedback for each student (Sun *et al.*, 2019) due to large class sizes. Teacher feedback has one of the most powerful impacts on student skill and knowledge development and plays a central role in learning. However, a critical challenge for computer programming is getting effective and timely feedback to help them acquire the necessary skills. Kinnunen and Malmi (2008) found that successful

students asking for help and support was a strategy they used when faced with a complex topic; however, this was used less by the students who dropped out of the program. Some learning strategies are based on offering alternative feedback sources until students receive authorised teachers' feedback (Nicol, Thomson and Breslin, 2014). In these instances, peers can serve as a valuable source of learning and feedback, which helps students achieve their objectives in the course with those already studying the topic.

### **Low study motivation**

Several studies state that one of the main reasons for students' dropout is a lack of motivation (Bergin and Reilly, 2005; Kinnunen and Malmi, 2006; Andrew Petersen *et al.*, 2016). These studies also show that students had no study motivation in general or reduced motivation over time because the payoff of studying is imbalanced (e.g., imbalance between the amount of effort has been made and the gains achieved), or some topics or tasks in the course were too difficult. They observed that a set of secondary factors could also affect the decision to drop out, such as low comfort level with the subject; the perceived course difficulty; issues with time management and study habits; and changes in majors or career aims. Yacob *et al.* (2012) suggested a number of strategies that can potentially improve low students' motivation, such as games, simulation tools, collaborative work, visual programming, pair programming, peer review, learning by doing and cloud programming. These strategies are based on engaging students in the learning process, and they must be used at early stages to avoid the decrease of first-year students' motivation. Thus, this study engages students in a learning environment, as a student's level of motivation is reflected in their contribution and engagement in the learning process (Gopalan *et al.*, 2017) to keep their motivations high.

### **Low confidence**

One of the most significant problems in programming from the students' perspective is that they often do not have enough confidence to complete individual tasks (Rahmat *et al.*, 2012), meaning that some students do not feel confident that they are qualified to proceed with programming studies (Kinnunen and Malmi, 2006). Many factors cause a

lack of confidence in learning, for instance, the newness of the task and course and the uncertainty of being successful (Norman and Hyland, 2003). In contrast, there are many factors that can increase students' confidence. Learning, experiencing and achieving are essential factors, and social interactions with others to get help can also increase students' confidence (Norman and Hyland, 2003). First-year programmers need to enhance the confidence so that they can continue to learn. Teachers, in turn, can support novice programmers in this by finding suitable learning methods that engage them in the learning environment; by increasing social interaction; and by giving the students the chance to critique and judge others work in order to become familiar with the assessment process and its requirements. All this can lead to an increase in the students' confidence and might thus reduce dropout rates.

### **No time to study**

Teachers and students all emphasise that not allocating enough time for studying the introductory programming course can cause students to drop out (Kinnunen and Malmi, 2006; 2008). For instance, topics that are more complicated require more time to complete meaning that studying as a whole will take more time. Sometimes students think that there is an uneven balance between the workload and the payback of the programming course related to the outcome phase of the educational process (Kinnunen and Malmi, 2008). In short, the skills acquired in an introductory programming course and the credit points gained for it are not perceived as equal to the workload. The perception is that the introductory programming course is indeed taking more time than it should do compared to the credit points, or that other courses are less demanding than the introductory programming course. To reduce the time spent on a specific topic or problem, it is thus an effective strategy for the student to learn to ask for help, whether from teachers or their peers.

### **Diversification for learning**

It has been found that students think that teachers lack diversity in their teaching strategies to solve problems (Siti Rosminah and Ahmad Zamzuri, 2012; Andrew Petersen *et al.*,



2016); thus, students think teachers should use a more diverse set of teaching strategies or methods to help them learn. First-year programmers, for instance, need the ability to read a program and write a program. Using a mix of exercises that combine generation exercises as a “down-top” with comprehension exercises as a “top-down” is a possible solution to help novice students in programming courses (Robins, Rountree and Rountree, 2003). Still, studies show very little research concerned with reading a program, especially for introductory programming courses (Robins, Rountree and Rountree, 2003). Reading a program requires a debugging strategy or the ability to track or hand trace a program to predict its behaviour and construct a program model, called ‘close tracking’, defined as taking the computer’s perspective (Robins, Rountree and Rountree, 2003). Building such a model is a fundamental part of program comprehension, outlining, debugging, and testing are associated with program generation. Consequently, and to achieve such a level of ability, Robins *et al.* focus on knowledge and strategies in the implementation and evaluation program that students need to learn. This study aims to apply a learning strategy at early stages, and focuses on reading and correcting programs to help students master the skill of solving fundamental problems.

Learning to program, which is the main component in computer science, has never been an easy task; however, many issues can be solved by employing a suitable learning strategy. Among the six of the most common types of learning strategies, a collaborative learning strategy is, for instance, worth exploring (Wegner, Minnaert and Strehlke, 2021). Students in collaborative environments are given the chance to learn realistic, socially exciting, and cognitively motivating learning contexts, compared to other learning models such as integrated learning, and discovery learning (Isaac, Christian and Amana, 2021). Collaborative learning is thus an excellent method to engage students to help each other with the same assignment activity and to create a learning community that values diversity. The following section outlines what collaborative learning in the context of learning how to program might look like, and discusses common learning strategies used with programming students.

### **2.3 Collaborative strategies for learning programming**

Collaborative learning is a process in which interaction between learners includes swapping ideas, sharing information, distributing responsibilities adequately, and engaging in discussions to solve problems to achieve learning outcomes (Boudia, Bengueddach and Haffaf, 2019). It provides a conducive environment that enriches and enlivens the learning process which, in turn, sustains students' interests and fosters a natural learning habit (Isaac, Christian and Amana, 2021). It is evident from an increasing number of studies that student participatory activities are increasingly emphasised in higher education; this is the result of educational studies that have recognised the positive role played by collaborative learning (Søndergaard and Mulder, 2012). The use of collaborative strategies in higher education is common in UK universities. However, in some countries, such as in the Kingdom of Saudi Arabia (KSA), collaborative learning is less popular (Boudia, Bengueddach and Haffaf, 2019). Many of these universities' computer programming courses are primarily based on lectures and face-to-face and lab sessions (Rahmat *et al.*, 2012; Boudia, Bengueddach and Haffaf, 2019). However, good computer programs require teamwork, cooperation, and social cohesion (Isaac, Christian and Amana, 2021). It is important to incorporate collaborative strategies at an early stage in the course as such strategies do not emerge overnight; they take time, practice, and an appropriate educational environment to develop.

Pedagogy that encourages active student engagement - by way of collaboration, for example - has been shown to keep dropout rates reasonably low and improve learners' overall performance (Aman, 2009; Sithole *et al.*, 2017; Walker, 2017). Applying the idea of learning communities has been proven to support students with some aspects of the learning process, especially within the first year of university (Kinnunen and Malmi, 2006) and within computer programming subject (Ben-Ari, 2004; Boudia, Bengueddach and Haffaf, 2019). Collaborative learning combines the benefits of individual and social learning processes, contributes to group members' participation, and stimulates student learning by creating a solid motivational system that leads to better performance results (Robins, Rountree and Rountree, 2003; Boudia, Bengueddach and Haffaf, 2019). It

increases students' self-esteem and self-confidence (Lin, 2015), which is hugely important for first-year students, as has been discussed above. In addition, collaborative learning promotes deep learning, as students engage in social interaction (Scager *et al.*, 2016). In computer science education, a deep-learning approach is essential for understanding concepts and complex processes. Understanding of these concepts includes a process of conceptual change; a process especially activated in collaborative strategies, whereas learners interact by explaining knowledge to peers or by questioning one another critically. Furthermore, collaborative strategies can provide timely support and reduce students' time spent on solving a programming issue. Consequently, students enjoy the learning experience, feel motivated, have increased confidence, persistence, effort, and achievement (Scager *et al.*, 2016). As a result, collaborative strategies could help first-year programming students in different aspects when well planned.

The following section outlines some collaborative strategies used in introductory programming courses. The discussion shows how collaborative activities impact first-year students, and peer assessment as a practical programming task was considered as one such collaborative activity.

### **2.3.1 Common collaborative learning programming strategies**

As the previous sections have shown, collaborative learning is a good way to involve students to support each other, especially at early stages of a course to overcome problems associated with being a new student studying programming. This section outlines various collaborative methods that have been used for teaching programming, and which have led to increased student retention rates as well as an improvement of students' programming skills (McDowell *et al.*, 2006; Carver *et al.*, 2007).

#### **Pair programming**

Pair programming, also called *collaborative programming*, consists of two programmers working side-by-side on designing, coding and testing a particular piece of software (Lui and Chan, 2006). Two different roles are involved in generating synergy between the partners. The first is a *driver*, whose role is to type on a keyboard and concentrate on the

coding details; the second role is that of a *navigator*, who actively oversees the driver's work, looks out for tactical and strategic defects, provides alternatives, manages tasks and researches references (Williams and Upchurch, 2001). The learning that takes place between the participants in pair programming is in the form of dual apprenticeship. One partner is the teacher, and the other is being taught, and vice versa. There is a constant transfer of knowledge between the partners, ranging from tool usage tips to programming language rules, design and programming idioms, and design skills as a whole (Arisholm *et al.*, 2007). Many studies have proven the effectiveness of pair programming for novice programmers (Lui and Chan, 2006; Celepkolu and Boyer, 2018; Papadakis, 2018). Lui and Chan (2006) found that novice–novice pairs are much more productive than expert–expert pairs. This indicates that activities with first-year students are more effective than advanced students.

Pair programming has many benefits for student programmers; for example, this approach can assist students in writing code with a higher level of confidence and higher program quality than they could individually (McDowell *et al.*, 2006). In addition, students learn how to discuss problems, work alongside each other as a team, enhance team communication and effectiveness, and are less likely to hide things from one another (Cockburn and Williams, 2001). However, the main drawback of pair programming is when a proficient student is paired with a weak student. The weak student may stop being actively engaged and becomes the observer while the proficient student completes most of the coding (Hanks *et al.*, 2011). So, first-year programmers, whatever their programming level, must not just sit back and observe their colleagues; they have to participate. Also, if both partners are weak, there will be little progress. Finally, choosing two students - based on their personality and level of programming skills - to work in a pair requires teacher effort and relies on the teacher knowing the characteristics of each student. This, however, is often difficult in a large class that is just beginning to learn programming.

### **Pedagogical Code Review**

A Pedagogical Code Review (PCR) involves a small team of students managed by a trained moderator; it includes (a) going over each other's work on programming solutions;

(b) checking the code with a list of ideal coding practices; and (c) discussing and noting the issues that emerge (Hundhausen, Agrawal and Agarwal, 2013), thus developing a piece of code and iteratively refining it through critical review. Students first review a piece of code individually. A group of students then come together to discuss the issues (weaknesses and improvements) that they found. If there are differing opinions, they develop teamwork skills, which provides opportunities for more extensive learning. After this review, the authors can present their solutions again based on the feedback provided. The PCR method differs from the peer assessment method in one significant way: PCR has interactive discussions, including both students and an expert moderator, whereas the peer assessment does not involve discussion and interactions between the team (Hundhausen, Agrawal and Agarwal, 2013). Many studies have shown the success of this method for novice programmers (e.g., Hundhausen *et al.*, 2009; Hundhausen, Agrawal and Agarwal, 2013; Pon-Barry, Packard and John, 2017).

Combining individual and team reviews with one piece of code leads to identifying a broader range of mistakes, which has synergistic pedagogical advantages. With individual reviews, learners can examine their peers' code at their own pace in a quiet environment, consequently raising the learner's chances to understand it (Hundhausen, Agrawal and Agarwal, 2013). Furthermore, after learners have studied their peers' code and collected their own sets of arguments, they are qualified to compare their views with others and engage in interactive and deep discussions. Likewise, the team review process offers a chance to practice collaboration, communication and teamwork (Hundhausen *et al.*, 2009). However, the drawback of this approach is that learners might have to deal with several errors at a time. Therefore, some first-year programmers may have a negative experience as they might receive a lot of comments in front of the team which may be confusing and frustrating. As a result, the primary psychological issues, i.e., motivation and confidence, will be affected. Further, PCR requires time since an individual completes work prior to the discussions. Peer assessment is less time-consuming but similarly effective, and so the following section focuses on this learning strategy which is also the main focus of this thesis.

## 2.4 Peer assessment

'Peer assessment' refers to two concepts; the first concept, 'peer', is described in the Oxford Dictionary as "an equal in civil standing or rank or equal in any respect", while the 'assessment' concept, according to Boud and Falchikov (2007, p. 9) is a "value-laden activity surrounded by debates about academic standards, preparing students for employment, measuring quality and providing incentives." Assessment is generally acknowledged as a critical and essential part of the education process (Balla and Boyle, 1994). The assessment provides a window into what learners know and ignore, and how learners are thinking (Earl, 2012). However, the assessment of students in higher education needs improvement, especially with regard to alignment with the results of quality agencies and with learning domains (Al-Ohali and Shin, 2013). The Quality Assurance Agency for Higher Education in the United Kingdom (Quality Assurance Agency (QAA), 2003) has produced a report after visiting universities in England and Northern Ireland over nine years. When focusing on quality assessment reports determined by QAA panels, the report identified that assessment is the area most in need of improvement. The report highlighted an over-use of traditional examinations and a limited range of assessment methods (Al-Ohali and Shin, 2013). For the reason that assessment is a key component of learning and helps students to learn, it is worth using this tool as one of the learning resources for first-year students in introductory programming courses.

The importance of the partnership between teacher and learner in an assessment has been recognised increasingly (Leach, Neutze and Zepke, 2001). This has accompanied a paradigm shift in assessment practices (Orsmond, Merry and Reiling, 2002). The idea of involving learners in the assessment process instead of restricting it to the teacher has become common and acceptable, manifesting itself in the form of peer assessment and self-assessment methods. For instance, it shifts from a primarily summative assessment (for accreditation and validation) to a more formative assessment (to encourage learning). This approach perceives students as responsible not only for their learning process but also for evaluating their own and their peers' performances. Thus, these approaches

produce a student-centred curriculum, in which the student becomes a participant in the decision-making process of designing and selecting educational experiences.

Peer assessment is a process in which learners provide thoughtful criticism of their peers' products or outcomes and give feedback using specific criteria (van der Pol *et al.*, 2008). In a peer assessment, the students consider “the amount, level, value, worth, quality or success of learning of peers of similar status” (Topping, 1998, p. 250). A peer assessment includes three components: students join a task; are assessed on the results of this task; and then use their peers' feedback to develop their work quality (Kollar and Fischer, 2010). This thesis argues that it would be beneficial to follow this approach in introductory programming courses because it has the advantage of making all students participate in the assessment process in which learners can showcase their emerging potential in making judgments and they can receive instant feedback that enhances their learning.

However, peer assessment is not as common in introductory programming courses as it is in other disciplines (Ashenafi, 2017). Therefore, this study employs a peer assessment activity in introductory programming courses. But first it is important to outline learning theories that support peer assessment in order to provide a basis for understanding how students learn, and as a way to explain, describe, and analyse learning. This contributes to the design of the peer assessment as a learning strategy for introductory programming courses.

#### ***2.4.1 Key theoretical perspectives in peer assessment***

There are numerous approaches to learning theories that encourage connections between students. The learning theories discussed in the following provide information about how students enhance their learning, in order to impact student attainment and achievement in their learning. The theories relevant in this thesis - social constructivism, the theory of experimental learning, and social development theory - pertain to collaborative learning strategies and their design. The following section provides a brief introduction of each of these three learning theories.

## **Social Constructivist Theory**

One of the most widespread and popular learning theories arguably still used as a basis for defining teaching and learning approaches today is constructivism. Constructivists see learners as active rather than passive (Taylor *et al.*, 2013). Those who ascribe to this theory believe that a learner's memory is always under construction to build knowledge (Spyropoulou *et al.*, 2013). Each learner conceives of reality in a different way, depending on his or her experiences and beliefs; and his or her knowledge is constructed rather than acquired from the outside. Constructivism is thus concerned with the construction of knowledge within learners based on their personal experiences (Ertmer and Newby, 2011; Spyropoulou *et al.*, 2013). Constructivists view the teacher's role as a facilitator instead of a dictator of learning, who specifies instructional strategies and methods that support the student in becoming an active learner. Social constructivism then falls into the category of constructivist theories. Already Dewey (1938) indicated that students learn well by interacting with others; additionally, student learning is enhanced when engaged in learning activities that have meaning for them as learners. Therefore, learning is seen as a cultural, social, and motivational process drawn from communication with people who have meaning for the learner.

The social constructivism model is often assumed in computer science education and the teaching of programming (Machanick, 2007). Action learning in programming tasks follows a cycle of starting from a desired outcome "project", designing a plan to solve the issue, implementing the plan "action", and reflecting on the outcome. If the issue is not yet solved, the cycle is repeated. Action learning works well when learning programming because many programming issues can be formulated in a problem-solving style, with the possibility of creating a plan that can be tested and evaluated for reflection. In trying to learn something, the learner can work with an expert or a group. As Barak (2017) argues, the learning process depends on worthy relationships between two parties: learning with a skilful expert, such as a teacher, and learning with fellow students. Differences in learning perspectives and the gaps between learners' current understanding leads to an increased cognitive imbalance that encourages learners to question ideas, modify existing



ideas, or adopt new ideas. The construction of new understanding can be demonstrated by cognitive dissonance (e.g., conflicting perspectives between peers that leads to an alteration in one of the perspectives), which is generated by interacting with others. As a result, the social constructivist model has been suggested to computer science education as having some useful properties for learning concepts (Machanick, 2007), and peer assessment could work in the context of a social constructivism model.

### **Experiential Learning Theory**

Experiential Learning Theory (ELT) defines learning as “the process whereby knowledge is created through the transformation of experience. Knowledge results from the combination of grasping and transforming experience” (Yen and Chang, 2018, p. 63). This theory is known as Kolb’s theory and provides a descriptive pattern of the learning process. It emphasises the key role that experience plays in the learning process. ELT describes two dialectically related methods of grasping experience: abstract conceptualisation and concrete experience, and two dialectically related methods of transforming experience: active experimentation and reflective observation (Yen and Chang, 2018). In brief, the learning cycle of Kolb’s model contains four processes that must be present for learning to occur: “feeling” happens in the concrete experience; “watching” happens in reflective observation; “thinking” happens in abstract conceptualisation; and “doing” happens in active experimentation (Yen and Chang, 2018). That is, learners perceive concepts through feeling and thinking, and process it by watching and doing. Teachers organise their teaching activities in a way that addresses all four learning styles: experiencing, reflecting, thinking, and acting. There are thus four roles for the teacher: facilitator, expert, evaluator, and coach (Passarelli and Kolb, 2012). Teachers objectively construct learning spaces through the knowledge and activities they present in their course; however, this space is interpreted in the learners’ individual experiences by the lens of their learning preference.

Experiential learning theory takes on many forms in the undergraduate and graduate computer science curricula. They include strategies such as project-based coursework, presentations, formal internships, simulations and case studies (Hoxmeier, 2002). These

elements are significant in computer science curricula because they improve the learner's understanding beyond abstract conceptualisation. Many computer science educators appreciate the value of experiential development theory where the students learn by applying theory and concepts to "real-world" situations (Hoxmeier, 2002). Kolb (1984, p. 21) claimed that experiential learning theory is "a holistic integrative perspective on learning that combines experience, perception, cognition, and behavior." Thus, under the experiential learning umbrella, peer assessment could be selected to ensure advanced students' participation to complete the learning cycle process and extensive interactions (VanSchenk Hof *et al.*, 2018). Experiential learning theory emphasises that participants have to deal with conflicting decisions and adapt knowledge to their own views, thus embracing reflection in the form of peer assessment. As a result, an experiential environment is an opportunity for differences that allow solving conflicts and for development through interaction within the peer groups.

### **Social Development Theory**

Social development theory was introduced by Lev Vygotsky (1896-1934). This theory maintains three main assumptions: 1) social interactions that play an essential role in social learning will precede the development and cognitive development process, additionally carrying among them the origins of higher mental functions; 2) learning occurs through the use of a mediator, indicated by some researchers as a More Knowledgeable Other (MKO), which refers to signs from a person who has a higher ability, higher skills, or better understanding than the learner with regard to a specific task – what normally comes to mind is a teacher or peer, but the MKO can also be a tool such as computers; 3) and the Zone of Proximal Development (ZPD) is where mediation can cause learners to advance (Barak, 2017). Vygotsky refers to the ZPD as "the distance between the actual development level as determined by independent problem solving and the level of potential development as determined through problem-solving under adult guidance or in collaboration with more capable peer" (Vygotsky, 1980, p. 86). What learners can do on their own is their level of current development, and what they can do with support is their level of potential development (Mishra, 2013). With this, Vygotsky indicates that learners

can be transferred from the current or actual level of development to the next attainable level by using environmental tools of a capable adult or peer facilitation. Thus, the ZPD contains the variety of cognitive processes that a learner initially cannot achieve independently and without assistance from others; then, the learning process responsibility incrementally increases for the learner to become independent.

Vygotsky (1980) describes the role of the teacher as assisting learners in the identification of decontextualised, systematic knowledge. Vygotsky's theory indicates that the teacher's main role is planning out instruction. When teachers plan well, they will consider the knowledge and skills that their learners are expected to master and decide the order in which to teach these concepts/skills. There are concepts requiring previous knowledge that learners may not yet have in some situations, especially novice students. In this case, this concept is now outside of their zone of proximal development; thus, the teacher could use scaffolding - a teaching method that assists student learning through teacher support or that of an advanced student - to assist students in mastering the concepts in sequential order. Considering, for example, if teachers use scaffolding with first-year students in peer assessment. In that case, teachers could prepare a rubric that guides students during the assessment to increase the likelihood of novice students meeting instructional objectives.

Although it is hard to determine a theory underlying peer assessment on account of its great variations, many researchers posit that peer assessment might be grounded in social development theory (Topping, 1998; Li and Gao, 2016; Li *et al.*, 2020). Vygotsky suggests that peer interactions are a fundamental part of the learning process and claims that pairing more competent students with less skilled ones leads learners to learn new skills. Accordingly, ZPD considers students' individual differences and encourages diversity in interactions to enhance students' learning. The idea is that students learn best when working with peers through collaboration. During such collaboration with more skilled students, learners internalise new knowledge, psychological tools, and skills (Yu, Hu and Zhang, 2013). So, since some problem-solving tasks in programming assignments can be more difficult for some students than others, it is essential to let students work with

more competent peers to solve a task. Considering diversity when pairing students in peer assessment is therefore critical, and this thesis examines that more closely.

All the learning theories discussed in this section play a part in understanding learning processes, although they vary in their competing understanding of how learning occurs. All of the theories can be used to develop instructional experiences for learners in an introductory programming course; however, each theory applies to specific situations more effectively than others. This project's most appropriate learning theory can be identified based on the requirements and learning situations envisioned for this study. As this study aims to integrate a peer assessment activity into introductory programming courses, Vygotsky's social development theory has been selected because it is best suited to introductory courses. This theory distinguishes students' capabilities and skills, making it suitable for first-year students. It builds knowledge from what the learner can do alone to what the learner can do with assistance using scaffolding and peers. For instance, some programming tasks are difficult for some students and not difficult for others. This study suggests accordingly that drawing on students who do not have difficulties with a task to help their peers who have difficulties can be an effective way for students to learn. Therefore, matching between different students' abilities in peer assessment can arguably support meaningful feedback. When all peers are at about the same level, they can help each other transfer from one level of ability to the next.

Peer assessment offers a range of learning benefits. These learning benefits offered by peer assessment are quite significant and warrant additional research. Productive benefits may be achieved through both receiving feedback and providing meaningful feedback to peers. The following section outlines some benefits of peer assessment, especially for programming students.

#### ***2.4.2 Benefits of peer assessment for programming students***

The discussion of benefits helps to ascertain the importance of applying peer assessment activities in programming courses, and encourages teachers to implement peer

assessment to their advantage. The following list outlines key benefits mentioned in the literature.

### **Improved knowledge and skills development**

Peer assessment can improve many programming skills, such as code review skills, problem-solving skills, and knowledge of coding standards (Indriasari, Luxton-Reilly and Denny, 2020). Peer assessment helps students learn such skills by organising, analysing, evaluating their peers' code and solving problems in it, leading to improvement in students' cognitive sophistication (Lynch, McNamara and Seery, 2012). Bloom's (1969) influential taxonomy puts evaluation at the top of the cognitive skills hierarchy; thus peer assessment could lead to higher-level learning skills. Students can also develop their skills of evaluating and justifying by assessing their peers work (Vickerman, 2009). This then leads to building students' confidence (Oldfield and Macalpine, 1995). Communication, teamwork and collaboration skills, also known as 'soft' skills, can also be improved (Hundhausen, Agrawal and Agarwal, 2013). In short, peer assessment helps students develop programming skills, employability skills and improve students' knowledge.

### **It supports learning**

Making judgments and offering feedback for peers' works provide opportunities to make the author consider what constitutes a good or bad program and what needs to be done to improve their programming style; it also encourages the reviewer to reflect on their own work (Sitthiworachart and Joy, 2003). As a result, their knowledge and comprehension of the subject matter are improved, their learning is facilitated, and they understand the assessment process more comprehensively (Vickerman, 2009; Gikandi, Morrow and Davis, 2011). Besides, it helps move learning from the surface phase to the deep stage because students extend their ideas, apply their knowledge and skills in new contexts, and are critical of arguments provided by their peers (Xie, Ke and Sharma, 2008). Peer assessment also helps to re-frame how students might view feedback from a constructivist perspective (Nicol, Thomson and Breslin, 2014). Students are not just learning by constructing meaning from their peers' feedback; they learn by making meaning for

themselves. By reviewing their peers' work, students build feedback 'meanings' for themselves as they provide it for others; that is, it is not an external input. Instead, it is an input created directly by the students themselves because they make critical judgments. As a result, peer assessment enables students to learn by engagement in receiving and providing feedback.

### **Improve the quality of the product**

With many peers engaged, learners can receive a variety of different feedback (Topping, 1998). Topping (1998) and Nicol *et al.* (2014) argue that more directive feedback – telling someone what to do – as well as more non-directive feedback – allowing the recipient to create their own solutions and actions – led to higher-quality draft assignments than when learners received a single comment from the teacher or one peer. This non-directive feedback in particular is valuable because it is positively connected with complex repairs in meaning. It does not need to be an expert in the field who gives feedback. When applying peer assessment, novice students can receive input from different peers. Feedback from multiple peers who have different perspectives can also sensitise students and make them aware of themselves as authors (Nicol, Thomson and Breslin, 2014), which allows them to choose which feedback is acceptable and which is not. By receiving different feedback, peer assessment can ultimately enhance coding quality, but it also makes students more aware of the quality of feedback they receive.

### **Social benefits**

Peer assessment is fundamentally a social process whose core activity is feedback given to and received from others. Social and assertion skills include becoming aware of giving and receiving criticism, explaining one's stance and refuting suggestions (Topping, 1998). Therefore, students feel they are more connected with their peers. They can collaborate with peers and have a significant discussion about a task which enhances verbal communication skills, diplomacy, negotiation and teamwork skills (Topping, 1998). In programming, teams are the foundation for the organisation of software development. Professionals in programming must have the knowledge and technical skills and

contribute and cooperate successfully within teams (Sancho-Thomas, Fuentes-Fernández and Fernández-Manjón, 2009). This thesis suggests that peer assessment can improve these social factors that are required in effective teamwork, such as collaboration, leadership and conflict management.

Therefore, programming teachers should be encouraged to use peer assessment activities in introductory programming curricula as the activity can help resolve or at least mitigate some of the problems identified previously (see section 2.2). For example, peer assessment can simplify the content difficulties students face in early weeks of starting a programming course by building a social environment and asking peers for help. Providing feedback to their peers themselves can lead to an increase in self-esteem and self-confidence as this creates a motivational environment that leads to better performance. Effective and timely feedback also helps students acquire necessary skills. Peer assessment is an unconventional learning strategy to solve problems. Peer assessment in programming courses follows a debugging strategy and tracks the written program to predict its behaviour and construct a program model. As a result, peer assessment could be an effective learning strategy to support first-year students.

However, some studies have highlighted significant barriers that hinder the use of peer assessment. The main obstacles are discussed below.

#### **2.4.3 Barriers to using peer assessment in programming courses**

The purpose of this section is to outline some barriers to the use of peer assessment with programming students and to explore how these potential issues can be avoided when designing a peer assessment in this study.

##### **Lack of knowledge and ability**

Some programming teachers have claimed that peer assessment is a complex teaching method because it requires both assessment qualities and content knowledge (Wang *et al.*, 2012; Li, Fu and Yang, 2017). Indriasari *et al.* (2020) categorised this issue as one of the most common computer programming barriers in higher education. Students – especially first-year programmers – may not have enough knowledge to assess their

peers and critique their code. However, the value of providing and receiving feedback in programming courses, especially with first-year programmers, is much greater than focusing only on the students' background knowledge or quality of feedback. This method has many benefits – as discussed above – in regard to developing skills and improving their knowledge by engaging in such a process rather than focusing only on improving the quality of the code that could need knowledge. Therefore, using this method is suggested as a form of assessment *for* learning rather than the assessment *of* learning. This study thus investigates how the quality of first-year programmers' assessments compares with tutors' assessments in terms of validity. In addition, it measures the impact of peer assessments (if any) on first-year programmers' performances.

### **Low engagement**

Active learning and students' engagement are crucial for improved student learning and ultimately retention, and thus significantly influences perceived learning and students' satisfaction (Gray and DiLoreto, 2016). Unfortunately, there is a low engagement in peer assessment activities from programming students (Indriasari, Luxton-Reilly and Denny, 2020). Some novice students think they have little time to review others' code (Stalhane *et al.*, 2004). Other students justify not engaging in peer assessment due to low enthusiasm or feeling rushed, meaning they do not have a helpful learning experience (Indriasari, Luxton-Reilly and Denny, 2020). There are many affective factors connected to student engagement: motivation, self-confidence, personality, attitude, and effort (Gray and DiLoreto, 2016). Thus, when students are motivated to perform well in their courses, participate or invest in their desire to learn, and are willing to make the effort expected by their teachers, they are more likely to be engaged in their learning (Gray and DiLoreto, 2016). It is important to discuss ways to encourage students to participate in peer assessment activities. This can be achieved by surveying students and interviewing them, as in this study, in order to improve engagement in peer assessment practices for future students.



### **Low review quality**

The reviews produced by students in some studies were not consistently trustworthy; there were, for example, instances of inaccurate assessments or low-quality feedback (Indriasari, Luxton-Reilly and Denny, 2020). Turner *et al.* (2008) found meaningless or unhelpful comments in some student review reports, and Hämäläinen *et al.* (2011) found substantial differences between students' feedback and instructor feedback. Some students may assess peers carelessly or have difficulty with completing the assessments. Thus, assessees may doubt the assessors' comments, meaning the peer assessment strategy ultimately negatively influences the learning process (Li, Fu and Yang, 2017). The quality of the feedback produced by students can clearly differ (Indriasari, Luxton-Reilly and Denny, 2020) based on their abilities; since there are differences between the students' abilities in learning, there will certainly be differences in the students' abilities in the assessment. Teachers should therefore guide students by providing detailed instructions and clear marking criteria. In addition, multiple reviewers may increase the reliability of the assessment (Indriasari, Luxton-Reilly and Denny, 2020). For instance, assigning more than one assessor to each assessee can help students see various opinions about their work, so if one assessor offers poor feedback, another assessor might be more constructive. This study seeks to find a solution that could improve the quality of review feedback.

Despite the diversity in academic literature that discusses benefits and barriers of peer assessment in programming courses, the source of the data used in these studies were primarily students' views that were gathered through surveys, interviews or experimental methods. The views of programming teachers regarding the benefits and barriers of peer assessment in introductory programming courses have been largely absent from this debate. This is why this study seeks to take into account the perspectives of both, programming students and programming teachers to compare their viewpoints. This responds to the first research objective: to identify the opinions of programming students and teachers in regard to peer assessment in introductory programming courses.

After clarifying the benefits and barriers of peer assessment, which concluded that peer assessment could be seen as a tool for learning in programming courses to set this goal, the following section describes peer assessment types to seek out the best ways to apply peer assessment in line with this study goal.

#### ***2.4.4 Types of assessment***

There are many types of assessment; the two most commonly used methods are summative assessment and formative assessment. Summative and formative assessments have different rules of engagement because they have contrasting intentions. The following section shows the difference between summative and formative assessment, and explores the idea of using formative assessment with first-year programming students in this study.

##### **Summative assessment**

A summative assessment process is defined as a judgment that includes all evidence up to a specific point (Taras, 2005). This point of judgment is indicated as a finality. As stated by the Quality Assurance Agency (QAA), which is an independent body that checks standards and quality in UK higher education, summative assessment refers to the extent of a student's success in meeting the criteria of evaluation to measure the intended learning outcomes of a course or module. A summative assessment is generally comprised of scoring carried out to assign a grade or other accreditation type (Gikandi, Morrow and Davis, 2011). Therefore, grades and classifications are initially performance indicators for the learner, the department, and the institution. The purpose of summative assessment is to measure learners' achievement at a particular time, resulting in the award of credits or qualifications for the students themselves, parents, teachers and other concerned parties such as school boards or governors (Harlen and James, 1997). This assessment is usually performed at specified intervals, that is, the end of a course or instructional unit or after a given period. Thus, a summative assessment has two characteristics: it relates to educational progress against public criteria, and it is used at specific periods when performance has to be reported.

Summative peer assessments are often high-stake, in that they result in significant decisions and actions for the subject and have a high point value. Topping (2005) suggests that summative marking a peer's work might make learners feel uncomfortable, but it makes them take on the teaching role. However, this approach may restrict learning effectiveness because simply deciding a mark may fail to include diagnoses of strengths and progressive aspects when assessing a piece of work (Butler and McMunn, 2006). It could be argued that, in a programming science subject, summative assessments could be used in project-based courses as students work in groups and know each other's participation, so they can make decisions on each other's work. However, as summative peer assessment involves evaluating peers' products and contributions as part of a grade; it is not a good fit for the research objectives in this study as this study aims to use assessment for learning to support first-year students.

### **Formative assessment**

Definitions of formative assessment have differed among scholars. Bloom (1969, p. 26) defines formative assessment as an assessment that seeks "to offer feedback and corrective measures at every stage in the teaching/learning process". Taras (2005) defines formative assessment as imposing feedback that indicates the gap between the learner's current level and the required standard. And Baleni (2015) defines formative assessment as an iterative process used to determine what, how much and how well students have learned in terms of the objectives and expected results. It becomes clear that the critical factor of formative assessment according to these scholars is the use of feedback to feedforward, in addition to helping learners improve their performance over time by reflecting on their goals and learning processes. Black and William (2009), on the other hand, do not make feedback a central issue in formative assessment, arguing that it is more important to focus on the theoretical models of learning and its organisation and implementation. They define formative assessment by illustrating it and its relation to and function of formative assessment. They consider class practices to be formative to the extent that evidence about a student's performance is extracted, interpreted, and handled by teachers, students or peers. They describe formative assessment aspects with three

agents (teacher, peer, student) and three processes: where the learner is right now, where the learner is going, and how the learner will get there. They clarify teachers' roles as three tasks: 1) explaining learning goals and sharing criteria for success; 2) organising practical tasks, activities and discussions that elicit evidence of learning; and 3) producing feedback that leads learners forward. The students' tasks focus on two points: 1) understanding learning goals and their criteria for success; and 2) activating themselves as an owner of their learning by discovering how to learn (e.g., self-assessment that requires learners to observe the quality of their work and organise their education). There are two tasks for the peer: 1) understanding and sharing learning goals and their criteria for success; and 2) activating peers as educational resources for each other (e.g., peer assessment, pair programming). According to Black and William, teachers, peers, and students can use activities to achieve the learning process and provide formative assessment. These are not sequential procedures but can be integrated into the learning context. A pedagogical technique that involves collaboration between the teacher, students and the individual learner can thus be categorised as a formative assessment.

Teachers in programming courses usually prepare assignments and their criteria as an individual practice; they support students individually in labs. However, these assignments are suitable to become tools for formative assessments. Teachers can explain the goals of specific tasks and their criteria for success, and organise activities that support formative practices such as peer assessment. Meanwhile, programming students should understand the requirements and learning goals and then observe the quality of their work by considering the teacher's goals and criteria. Peers in programming labs can help each other; seeing other solutions to the programming assignments can help them comprehend content knowledge better. In this way, students compare multiple solutions, judge their peers' work and think about the strengths and weaknesses of their own solutions. This study follows the definition of formative assessment provided by Black and William, as it fits well with peer assessment for learning.

The benefits of formative assessment are well documented, and literature has shown that formative assessment practices are supplementary with enhanced educational

achievement (Black and Wiliam, 2009; Baleni, 2015). Formative assessment reduces students' nervousness; provides students with a feeling of agency as they develop; and supports the knowledge of the course contents (Baleni, 2015). Unlike summative assessment, formative assessment has a purpose more closely connected to teaching outcomes and offers a potential for refining learners' learning that is more instantaneously clear, as well as educationally appropriate. Many studies assert that formative assessment is most suitable for novice programmers (Sitthiworachart and Joy, 2003; Stegeman, Barendsen and Smetsers, 2014; Sun *et al.*, 2019). Regular and frequent formative assessment, particularly in the early weeks of the first year of higher education, is one of the pre-conditions for student success as it reduces the chances of students dropping out (Nicol, 2009; Baleni, 2015). Using formative peer assessment aims to monitor student learning through peers who provide instant feedback, which, in turn, leads to improvements in understanding, self-esteem, and confidence and encourages a higher level of achievement and thus motivates students to remain in programming courses. Formative peer assessment also has low stakes as there is no point value on students' official course works. This study, therefore, suggests that an effective amalgamation of formative assessment in introductory programming courses could offer a suitable learning method.

### **Self-assessment**

Although this research applies peer assessment for learning, it is worth mentioning self-assessment and its relation to peer assessment. Both peer and self-assessment are essential methods of lifelong learning. Teachers who implement peer assessments usually aim to facilitate self-assessment by helping students reflect on their own performance, increasing their self-improvement and self-reflection skills (Carless, 2011). When first-year programmers review their peers' work in a specific assignment, it is mainly a self-learning process with students teaching themselves, as they can see how another student has solved a particular task and how it can be applied to their work. Therefore, novice students can develop their knowledge by thinking about a peer's piece of work and how to improve it, then apply it in the context of their own work. Whenever students

produce a piece of work, they give internal feedback – they reflect on their own performance - even in the teacher's absence. As students themselves have reported (Nicol, Thomson and Breslin, 2014), reviewing others puts the feedback processes in their hands. It decreases the student's need to receive feedback from others, which is demanding in programming courses, as students often need timely support, which is sometimes not available. This feedback is an outcome of task engagement; it is acquired from the student's inner monitoring and assessing the differences between current and intended performance (Butler and Winne, 1995). Therefore, to guide such internal feedback, self-assessment is a good tool.

Receiving external feedback from a teacher or peer, in contrast, is then similar to getting told what to do and how to do it. External feedback, however, is more effective when it is accompanied by the learner's internal feedback to confirm, supplement or sometimes conflict with it. Thus, it is effective to employ self-assessment activities to improve programming students' performance and reduce external feedback (Nicol, Thomson and Breslin, 2014). However, studies on feedback have neglected these internal feedback processes (Nicol, Thomson and Breslin, 2014). Research on programming students (Chou and Zou, 2020), for example, has shown that there is poor internal feedback in programming courses, which leads to poor learning performance. This study addresses this issue by highlighting how self-assessment might improve internal feedback processes and enable students to compare and calibrate internal and external feedback in ways that support their learning. Accordingly, this study uses self-assessment alongside peer assessment for two reasons: The study wants to encourage students to understand assessment criteria and improve their internal feedback; therefore, when they assess themselves, they will consider each criterion. Besides, assessing themselves helps to assess peers fairly because they will put themselves in the position of the assessee.

To sum up, there are alternative ways to assess students' progress and enhance learning, which is deemed an imperative part of the education process. The main aim of summative assessment is to measure students learning and award credits (or equivalent). Formative assessment aims to use feedback to feed-forward and help students improve their

performance. While self-assessment seeks to help the individual know the extent of their abilities and to enhance upon them. This study should determine the type of assessment based on its objectives. Since the main objective is assessment for learning, the study employs a formative peer assessment alongside self-assessment. The next section then provides common methods that can be used in peer assessment in order to find the most suitable method to encourage peers for learning.

#### ***2.4.5 Methods of peer assessment***

The choice of appropriate assessment methods in a particular course is determined by several factors: the course objectives, the course background, the anticipated learning outcomes, the available resources, the students' characteristics, and the level of study (The University of New South Wales, 2015). As peer assessment methods can be classified into two main approaches, the following section reviews both and explains which one may be best for use with first-year programmers.

#### **Holistic assessment**

Holistic assessment is a general approach to assessing a student's learning outcome. It is defined as one-dimensional criteria used to assess students' overall achievement on a task based on predefined success levels (Sadler, 1989). In a holistic assessment, the assessor responds to a learner's work as a whole, then decides the quality of the work based on a notional point on the grade scale (Sadler, 2009). The assessor must clarify reasons to decide the assessment grade for a learner's work (Akubuilu, 2012), and the feedback contains a summary of the significant points in the student's work. This type of assessment can be summative or formative. As with any assessment method, the holistic method has pros and cons. This approach is more suitable for facilitating creativity in the assessor because he/she justifies the assessment decision, allowing work to be appreciated for its characteristics (Cateté, Snider and Barnes, 2016). However, holistic grading is sometimes described as impressionistic or intuitive (Sadler, 2009) because it may be based on subjective perspectives that could be presented unsystematically. Thus, it is not easy to apply such methods in peer assessment with first-year programmers as

they have not yet built their confidence to receive such impressionistic feedback. Moreover, a holistic assessment requires a knowledgeable background to provide feedback. Therefore, it is difficult to convincingly justify grading by some of first-year undergraduate students, which is especially true in programming courses because students at this stage do not have enough knowledge.

### **Category-based assessment**

A category-based assessment is a two-dimensional method that displays criteria with performance levels as columns and assessment criteria as rows (Lejk and Wyvill, 2001). A category-based assessment should produce multiple categories and criteria that can be evaluated separately, thus assessing different aspects of a student's performance. The most significant feature of a category-based assessment is that it provides a nuanced and detailed image of a learner's performance by considering different aspects of quality levels. Lejk and Wyvill (2001) have compared holistic and category-based approaches to peer assessment; they found that a holistic assessment supports the objectives of a summative peer assessment within team project work better than a category-based assessment. But they argue that a category-based assessment is helpful for formative evaluation. Thus, as this study uses a formative assessment, a category-based assessment was selected with first-year students.

Although a category-based assessment does not give a reviewer freedom of criticism as it guides students to assess based on selected criteria, it is less biased because every decision the reviewer makes depends on a specific criterion. In the context of this study, this would help first-year programmers make judgments and justify them based on criteria. Category-based assessments provide detailed explanations of what is expected of the assessee and are usually less prone to bias (Kavanagh and Luxton-Reilly, 2016). A category-based assessment is also more suitable for those who do not have significant background knowledge, as criteria can guide assessors when assessing peers. It can thus be used as a learning tool, not just as a list of criteria for assessment. Since this study seeks to improve the student learning process in programming rather than summative evaluations of students, a category-based assessment was selected. This approach has



been used in multiple peer assessment studies with programming students previously (Sitthiworachart and Joy, 2003; Sitthiworachart and Joy, 2008; Hamer *et al.*, 2009) which suggests this to be a suitable approach. The following section describes a rubric which is an important tool in the category-based assessment.

### ***Using a rubric in peer assessment***

To analytically assess students' work, teachers have used various techniques in the past, the most common one being a checklist in which criteria are itemised and can be monitored to show performance. Another common technique is a rubric that defines various degrees of achievement and clarify all combinations of criteria and levels. A rubric then is defined as a "document that describes varying levels of quality, from excellent to poor, for a specific assignment" (Andrade, 2000, p. 13). Though the form of a rubric can differ, all instructional rubrics contain two aspects: 1) a criteria list related to an assignment or project; and 2) scales of quality, with descriptions of excellent, average, and poor student work. Assessment criteria are essential in ensuring that educators and learners have mutual knowledge of what is being evaluated (Jones and Alcock, 2014). A study conducted by Song and Hu (2015) demonstrates the effectiveness of rubrics in helping students understand assignments' purposes, providing instant feedback and motivating learners to complete their tasks. Furthermore, studies show that the instructional rubric is predominantly designed to be a teaching plan rather than a scoring tool (Stegeman, 2014; Cateté, Snider and Barnes, 2016), because it clarifies what the expectations are for a high level of performance on a given assignment, and how students can be met.

Rubrics are easy to use and to explain (Andrade, 2000). They generally make sense to students quickly and are brief and digestible. They also set out teachers' expectations clearly (Andrade, 2000). The rubric also provides students with feedback and detailed evaluations of their own work (Andrade, 2000). So, students become aware of their own strengths and weaknesses by applying these criteria to their work. Rubrics thus support learning and the development of skills, and help develop understanding (Andrade, 2000) by asking students to think about the criteria prepared by the teacher and to understand the assignment's requirements. Since this research is concerned with learning, a

category-based rubric was selected to aid first-year students to understand criteria required by their teachers, and then aid in assessing their peers' programming assignments. However, teachers have to consider how to build an effective rubric that supports first-year students. The following section reviews literature that focuses on criteria that can be used in rubrics for students.

### ***Rubric criteria for programming assignments***

Various studies in programming courses have used rubrics in peer assessment activities; as a systematic review clarified, whereas around 65% of the studies used rubrics to help students to review the peers' code (Indriasari, Luxton-Reilly and Denny, 2020). Since this study develops a rubric model for peer assessment, this section presents studies in which different rubric models were used in peer assessments in introductory programming courses. The focus is on the criteria used in these previous studies so to inform an appropriate criterion for this study.

Sitthiworachart and Joy (2004) used eight criteria: comments, consistently indented, handling errors, naming variables, using exit statuses, appropriate utilities, considering the structure of the program, and the ease of program flow. The aspects covered in the rubric are basic criteria that are easy and quick to use by first-year students. In another study by the same scholars, three categories with the following criteria were described (Sitthiworachart and Joy, 2008):

1. Readability – considers comments, indentations and variable names;
2. Correctness – focuses on meeting the specifications, handling errors and exit status;
3. Style – considers appropriate utilities, program structures and easy-to-follow code.

This rubric is more accurate than the one described in the previous study (2004) due to the use of categories. The scholars also used scales ranging from 1 to 5 to describe the quality of work. They found novice students practiced in marking became familiar with what is required to be accomplished and the assessment process. They also found that subjective criteria, for example, 'easy to follow' and 'helpfulness of comments', were difficult to mark consistently when compared with objective criteria. They thus recommend

avoiding subjective phrases when building criteria. Based on the outcomes of this study, the decision to use categories was taken to organise the rubric in the current study and avoid subjective sentences.

Another study, conducted by Park *et al.* (2017), described the following rubric dimensions:

1. Correctness – correct program, no errors and meets specifications;
2. Program design – well-structured program and easy to follow;
3. Efficiency – good use of algorithms and language features;
4. Readability – understandable code, well-organised and clean;
5. Assignment specifications – program follows the details of the assignment.

The students examined as part of this study assessed their peers' code in three ways: 1) assign a numerical score for each criterion; 2) write feedback for each criterion; and 3) directly comment on the code snippet. These categories are basic, easy, and quick to use by first-year students. The scholars found that it is important to set up criteria for peer assessment in programming courses that are relevant and can be understood by students. They also acknowledged that some criteria are subjective, such as 'good use of algorithms' and 'well-organised'. The study also asked students to assign a numerical score for each criterion rather than select a level on a scale.

The rubric model used in the final study under discussion here differs in detail and specificity. Hamer *et al.* (2009) used criteria with several descriptive levels to describe scale quality specified for each assignment question. It allowed students to use superior performance differentiation and make an accurate judgment. They used two general dimensions:

1. Correctness – that focuses on the output and specific criteria based on the question,
2. Style – that considers comments, indentation, variable names, variable identifiers, and use of symbolic constants.

They used a rubric containing a section for selecting numerical grades and a section for adding comments. The rubrics they used were varied for each exercise, but they followed the same structure: correctness category; and programming style category. They found

that differences in rubrics might account for the differences in lexical sophistication between assignments. They also found a difference in the quantity of comments written between engineering students and computer science students. Engineering students wrote more comments and scholars related that to previous marking experiences in those subject contexts. To construct a detailed rubric, especially for each assignment, it may be an extra load for teachers. Still, investigating the customization of specific criteria for each assignment with a descriptive scale must be considered with programming teachers.

An effective rubric allows novice students to learn three things: the meaning of good performance on an assignment; the relationship between their performance and good performance; and the means of closing the 'gap' between them (Stegeman, 2014). From these previous studies, it can be concluded that a category-based rubric is suitable for computer programming courses requiring students to learn from good performance criteria. Likert scales were commonly used in these rubrics, requiring students to select the option that best aligns with their perspective rather than assigning scores. However, scales have not been considered in the literature where the focus has been on criteria and categories rather than on a suitable scale. In some studies, simple lists of criteria without levels were used in conjunction with Likert scales to find the option that best aligns with their perspective (Sitthiworachart and Joy, 2003; Stegeman, 2014; Shui Ng, 2017). A study conducted by Hamar *et al.* (2009) uses a set of criteria with several descriptive levels and marking scales, and applies detailed rubrics, allowing for superior differentiation of performance. Thus, it is important to formulate assessment standards that ensure learners will know exactly what is being evaluated (Lejk and Wyvill, 2001), and discover a suitable quality scale that is fit for first-year students assessors, thereby improving the quality of the peer assessment (Panadero and Brown, 2017). As a result of the varying outcomes of previous studies, different rubrics forms must be compared to ascertain which one is most effective for first-year students in enabling them to assess peers in this study. This has not been explored in other studies previously. In addition, this study aims to determine the rubric elements that should be considered during peer assessment to find the most important criteria for first-year students.

So far, previous sections have provided an overview of peer assessment, its definition, benefits, barriers, types, and methods. Getting an overview helped to find a source of information that gives the reader a simple understanding of peer assessment without telling all details. The following sections show some details of peer assessment, and determine relevant gaps. The next section focuses on the validity of the peer assessment and investigates the impact of its application on the performance of programming students. The evidence of the validity of a peer assessment is a prerequisite for ensuring its integrity and quality.

#### **2.4.6 Comparing peer assessments with tutor assessments**

The agreement between teachers' total marks and peers' total marks in previous studies varied (Li *et al.*, 2016), at least regarding subject matter, educational context and student experience (Hamer *et al.*, 2009). For instance, Kovach *et al.* (2016) used a rubric and found that the correlation between peer assessment and tutor assessment in an undergraduate internal medicine subject was weak ( $r = 0.29$ ). Another study by Cho *et al.* (2006) used a rubric and found that the correlation between peer and teacher agreement was medium ( $r = 0.60$ ) in an undergraduate psychology subject. Furthermore, a study by Harris (2011) used a rubric, and there was a high correlation between peer and teacher ratings in a physiology subject ( $r = 0.97$  and  $0.98$ , respectively). These opposing results could be because they are different subjects. As a result, the correlation between peer assessments and tutor assessment may relate to the subject matter and could differ from one subject to another.

In computer science subjects, the correlation between tutors' grades and peers' grades is slightly different. Li and Gao (2016), for instance, used a rubric in their study, and they found the correlations between the peer and teacher assessment ratings agreeing with each other at a medium level ( $r = 0.63$ ), and in the study conducted by Hamer *et al.* (2009) study, the correlation was ( $r = 0.78$ ). This inconsistency needs to be addressed in order to determine an accurate peer assessment in the context of programming courses - as researchers consistently focus on final marking without considering the detailed marking in each category/dimension - which is what this study set out to do. This study also

considered what dimensions of program quality first-year programmers may be able to assess accurately and consistently in line with tutor assessments of the same work. Thus, this study explored the consistency and the correlation between tutor and peer marking using a marking guide (the second objective) investigating the accuracy of first-year students in peer assessment. In addition, the study examined the correlation between rubric's dimensions in order to establish to what extent student feedback may be useful and valid instead of whether it is correct or incorrect.

As a result, peer assessment in programming assignments could provide consistent results with teachers' reviews across the assessment criteria and tasks, leading to higher student achievement. Evaluating the students' achievement following a peer assessment activity is about measuring the growth in students' performance by comparing where they were at an earlier time with where they are now or with other groups of students. The following section focuses on the impact of a peer assessment activity on students' performance, with consideration of the measurements used.

#### **2.4.7 Impact of peer assessment on learning to program**

This study's second objective is to investigate the impact of peer assessments (if any) on first-year programmers' performances. There are fewer studies of peer assessment in the context of introductory programming courses in comparison to other subjects, because peer assessment is not used much in introductory programming courses. This is why the study explores the impact of peer assessment on different subjects in addition to programming courses to determine a suitable method to measure the impact of peer assessment on the performance of first-year students. Generally, previous studies on peer assessment with undergraduate students have confirmed that it has positive influences, especially concerning its impact on learning (Li and Gao, 2016), because it leads to successful critical thinking (Li *et al.*, 2016), deep understanding (Lynch, McNamara and Seery, 2012), higher quality learning outcomes (Lynch, McNamara and Seery, 2012), improved academic performance (Sadler and Good, 2006; Shui Ng, 2017; Double, McGrane and Hopfenbeck, 2020), and learning gains (Li *et al.*, 2020).

The implementation of peer assessment methods is often inconsistent (Topping, 2010), and studies also differ in how they measure the impact of peer assessment on student learning. For instance, Li and Gao (2016) used an experimental method to measure the impact of peer assessment on students' project performance. Their research measured the quality of learners' work and then split learners into low, average, and high levels based on learning performance on a specific project in the course. Their findings confirm that the impact of peer assessment varies according to students' learning levels. They found low and average levels of students showed a significant positive enhancement in the learning performance following a peer assessment activity. This finding is important because this study aims to integrate peer assessment in daily education practices in programming courses, and students in these classes have different learning levels; therefore, diversity in the abilities should be considered when designing peer assessment for first-year students.

Huisman *et al.* (2019) performed a meta-analysis to examine the impact of peer assessment on writing performance with higher education students. They compared students who received feedback from peers with those who received no feedback at all. They focused on the effect sizes (standardised mean differences) that were computed based on reported group means and standard deviations. The researchers found better writing enhancement with students who participated in peer feedback than those who did not. While their study focused on academic writing performance, this study could use the same method (dividing students into two groups and applying peer assessment in one group) to investigate the effect of peer assessment on students' performance.

Regarding computer programming, Wing-Shu (2017) found a positive effect of peer assessment on programming skills when using a t-test to find the correlation mean of quiz results. The study involved a small sample size (only 16 participants), so it was not conclusive. King (2018) compared homework scores before and after peer assessment activities. He found that the impact of peer assessment on the programming students' ability can be seen through the improved performance in the second homework assignment. He analysed the two homework performances by using the Pearson chi-

square and used multinomial regression to compare the effect of peer assessment's acceptability. He found students enhanced their homework performance level in the second homework. However, homework assignments are not a suitable tool to compare, as many students do not put as much effort into homework as they do into exams. Also, the research did not clarify the time between the first and second assignments, which means that the impact of external influencing factors cannot be ignored. This study then should use official variables (e.g., pre-test and post-test) to measure the impact and should compare between two groups to ensure the effect of the intervention.

This study uses an experimental method with a large sample size, and uses a pre/post test to accurately investigate the impact of peer assessment on students' programming performance. This can be achieved by comparing the effects on the performances of students who have assessed their peers to those who have not participated in the peer assessment (using a mock exercise that does not impact grades). Official measurement could be used, such as a quiz or midterm, instead of conducting a pre/post-test to make sure students are well prepared for these exams; this ensures accurate result when measuring the impact of peer assessment on students' performance.

To sum up, measurement of the accuracy of peer assessment and its impact on students' performance means increasing the credibility of the activity in the context of introductory programming courses and more encouragement for programming teachers to apply peer assessment. However, the sample that would be selected to measure their accuracy in the peer assessment and the activity's impact on their performance must be considered. The following section describes the use of peer assessment in the targeted population. The target population is the group of people that the intervention intends to study and draw conclusions from. Characteristics of the target population should be described clearly to ensure accurate findings are obtained.

#### ***2.4.8 Peer assessment in UK and KSA higher education***

Active and collaborative learning that incorporates learners to participate in the learning process is common in UK universities due to the increasing demand for lifelong practices,



more responsible activities, thoughtful and critical professionals, and a preference for approaches that relate learning with its assessment (Iglesias Pérez, Vidal-Puga and Pino Juste, 2020). Particularly, self- and peer assessment are becoming core factors of student-centred evaluation processes in the field of higher education because they have an instructive impact (Wanner and Palmer, 2018). Both forms of assessment are valuable for developing critical skills in learners, such as developing a better understanding of the subject's content, developing critical reflection skills, taking responsibility for their learning, and evaluation criteria and their judgments and values. Besides, peer assessment allows interaction and collaborative in the group, making learners a critical subject. Therefore, self-assessment and peer assessment are common and beneficial learning tools in higher education in the UK, as many studies have suggested (e.g., Ashenafi, 2017; Shui Ng, 2017; Panadero, 2019) and concluded that it is an effective tool for students to support their learning process (Wang, Liang and Liu, 2012; Carbonaro and Ravaioli, 2017; Panadero and Brown, 2017).

However, in Asian countries, there is a lack of critical thinking, evaluation, and intellectual skills being deliberately taught or developed in the programming curriculum. Teaching focuses more heavily on rote passive learning techniques (Shaheen, 2016). For instance, in Saudi Arabia, assessments often follow traditional teacher-centred approaches to student assessment that mainly focus on examinations (Al-Ohali and Shin, 2013). Traditional assessment methods and a norm-referenced assessment culture continue to prevail and are largely dominant in Saudi Arabia (Al-Ohali and Shin, 2013). For instance, the Ministry of Education regulations in Saudi Arabia declared that 60% of the final course grade must be allocated to the final exam, and 40% to midterm exams and all other activities and assignments (regulations did differ during the Covid-19 pandemic). There is still a broad view held by teachers in higher education in these pedagogical cultures that summative assessment should be the main form of assessment. The focus on assessment through traditional assessment methods forces students to concentrate on their grades and marks instead of the actual learning process. To challenge this culture

and to improve students' skills, especially higher-level thinking and soft skills (Al-Ohali and Shin, 2013), more proactive methods for students' training and development are needed.

Therefore, this study uses peer assessment approach to learning. The target samples focus on Saudi students as a sample of those unfamiliar with such approaches and compare them with UK students who may be more familiar with them. The result can potentially support the use of peer assessments if first-year programming students from two countries held similar perspectives regarding the values and importance of this method. It could also provide invaluable information about the value of peer assessment activities in introductory programming courses that can be generalised for practice in other institutions and settings. In general, comparative studies try to explain whether certain behaviour patterns are specific to a particular culture or group, or are valid cross-culturally.

As one of the milestones of this study is integrating a peer assessment into programming courses, how to apply this peer assessment must be considered. Therefore, the following section reviews qualitative and quantitative studies to determine how previous studies have implemented peer assessments. The section particularly explores students' and teachers' perceptions of implementing peer assessments in programming courses, from which the factors they require and the problems with which they are concerned can be deduced.

#### ***2.4.9 Quantitative and qualitative studies of peer assessment***

Student experiences of peer assessment in the context of higher education using quantitative methods have been widely studied (Hwang *et al.*, 2008; Sitthiworachart and Joy, 2008; Shui Ng, 2017); however, there is little qualitative research on this topic, particularly regarding the use of peer assessment in introductory programming curricula. As with any learning strategy, some students may be reluctant to assess others' work (Li and Gao, 2016), thus first-year students may need specific requirements when they apply peer assessment (e.g., model answer instead of rubric, multiple assessors work together to assess a specific work). Understanding programming students' perspectives is in fact necessary to tailor such an activity to a programming course.

Current qualitative research that aims to incorporate the peer assessment process into learning practice, especially among programming students, pays little attention to students' perspectives. Some qualitative studies have examined students' opinions on peer grading (Sitthiworachart and Joy, 2004; Stegeman, 2014); another study has detailed their perceptions about completing peer assessments (Park and Williams, 2016); and another study has observed student interactions during the peer assessment process (Lynch, McNamara and Seery, 2012). However, none have clarified how students prefer to apply peer assessment in programming courses: what main factors do programming students consider to be essential in peer assessments? And what tasks do they need? Thus, this study focuses on students' expectations and their concerns regarding implementing peer assessment in introductory programming courses (the third objective).

Teachers play a significant role in preparing and moderating peer assessments; their roles, experiences, and attitudes towards peer assessment are critical factors when using this type of activity with students (Panadero and Brown, 2017). However, no quantitative studies and only a small number of qualitative studies consider programming teachers' attitudes towards peer assessment. King (2018), for example, conducted interviews with teachers to assess the acceptability of peer assessments for coding assignments in a large lecture. They stated that peer assessments in homework assignments are feasible in large computer programming classes. The interviews discovered that the peer assessment alleviated the teachers' workload associated with providing significant feedback to students on their assignments given limited time resources. However, some teachers believe that peer assessment of programming code, especially with first-year programmers, is difficult because various constructs and logics can produce the same output (Stegeman, 2014); therefore, this may produce inconsistency in the assessment. The differing opinions of programming teachers about using peer assessment with first-year students lead to the necessity for collecting their opinions to determine their position on peer assessment in introductory programming course accuracy. Furthermore, past research paid attention to teachers' perspectives of the peer assessment process, and neglected consideration of teachers' requirements to implement such activities; this may

be the reason why teachers appear to have different opinions. As a result, teachers' perspectives are important in this study. Qualitative and quantitative methods should be used together to find teachers' perspectives, expectations and concerns towards peer assessment. Using different methods to collect data in this area make results more credible, as programming teachers' perspectives on the use of peer assessment with first-year students is not clear in the literature.

Since this study seeks to design a prototype that represents how to implement peer assessment in programming courses, the following section describes some automated web-based peer assessment systems. This study explores these systems to utilise their advantages and avoid their disadvantages when developing the peer assessment prototype for first-year programmers.

#### ***2.4.10 Evaluating common peer assessment systems***

Several peer assessment platforms have been proposed in the literature, but many systems are confined to a particular course, area of research, or domain. This section focuses on common peer assessment systems available for a variety of courses and accredited by various educational organisations. This section describes the following aspects of these systems:

1. What are the functionalities users can use?
2. How does the user interact with this system?
3. What is the pedagogical rationale behind the system?
4. Could the system be used to evaluate code?
5. How satisfied are users with the system?

#### **PeerScholar**

A widely used system is PeerScholar; it is an online peer and self-assessment system designed for peer evaluation in different courses to improve students' learning skills. Steve Joordens and Dwayne Pare co-founded PeerScholar, and they first started using PeerScholar in the classroom in 2008 (Chris, 2016). Joordens is a professor of psychology and the director of the Advanced Learning Technologies Lab at the University

of Toronto's Scarborough. So, this system was designed by an academic teacher interested in peer assessment who had teaching experience. The system contains three main functions for students: create, assess, and reflect. For example, the student is first required to write and submit their essays (create phase). Then, anonymous fellow students randomly evaluate the student's essay by adding comments and scoring it using an instructor's rubric (assess phase). After that, this tool allows the student to rate the reviews they received (reflect phase) (Mogessie, Firlab and Riccardi, 2016). Figure 2.1 shows the reflect phases of the PeerScholar system. For the teacher, the system contains three main functions: build an assignment, set up the assignment, and customise grading. The system gives the teacher the freedom to decide the calculation method.

Regarding the pedagogical theories behind PeerScholar, Joordens, Shakinaz and Paré (2009) highlight the learning environment that PeerScholar provides in terms of Bloom's hierarchy, the 5Es of effective learning, and the notion of constructivist learning. The developers were particularly focused on individual learning theories rather than collaborative learning theories. Students' attitudes towards the PeerScholar system tend to be positive (Collimore, Pare and Joordens, 2015) as they found the peer assessment process helpful and felt that they benefited from the system. Teachers who have used the system have described it as an ideal tool (Rawn, 2021). However, the system was examined by the researcher and found that it is not suitable for debugging codes; it allows users to enter normal texts (e.g., essays) but not script codes. What is more, this tool fails to consider the quality of the peer reviews, which may decrease students' motivation to use it. The program also lacks focus on new computer technologies, such as adaptation, data analytics, data mining, and predicting.

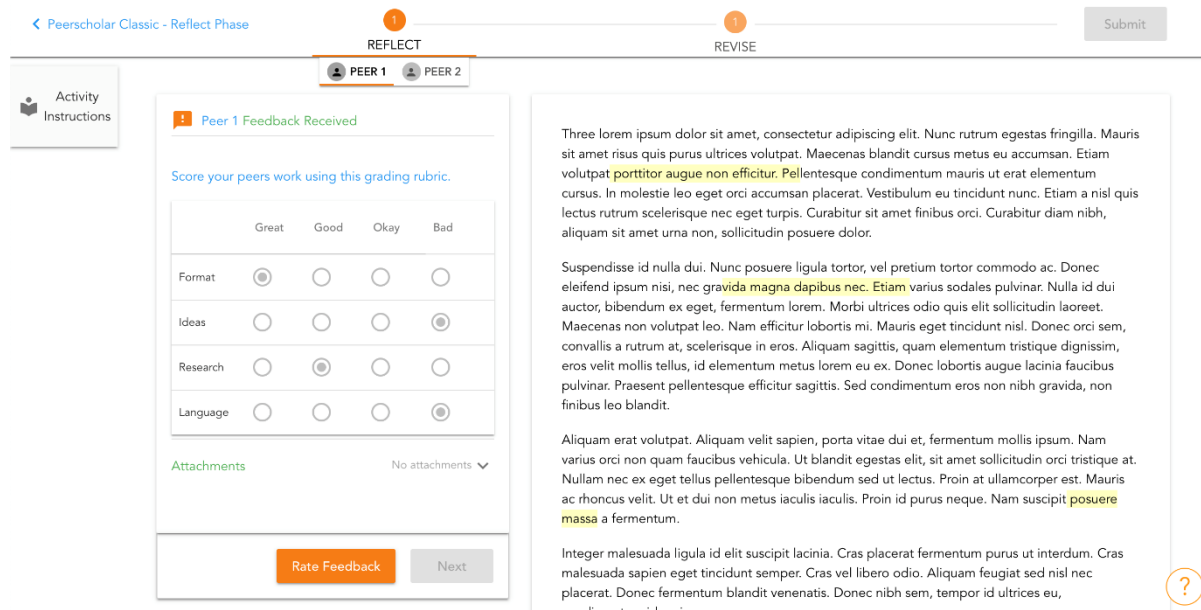
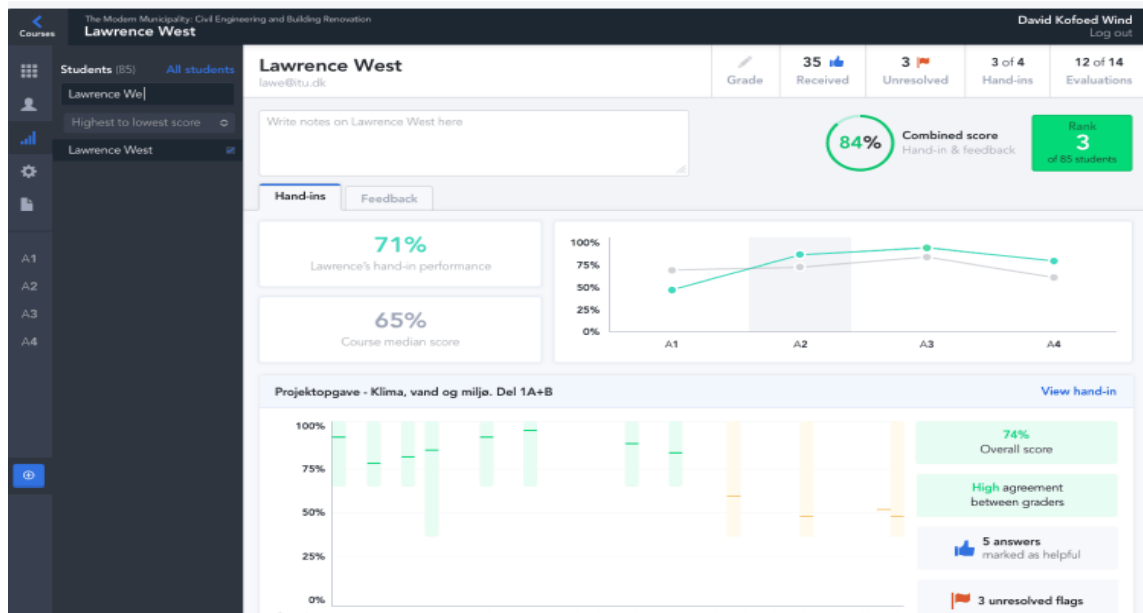


Figure 2.1. PeerScholar system

## PeerGrade

PeerGrade is an online educational platform designed to make it easier for teachers to manage peer assessment and facilitate peer feedback with students (Graham, 2017). The founder of PeerGrade is David Kofoed Wind, who has a PhD in applied math and computer science from the Technical University of Denmark; he launched the site in 2018. For teachers, PeerGrade offer to create an assignment for the students and set up the assignment as desired, as well as calibrate and moderate reviews provided by students to get a different insight into the reviews (Wind, Jørgensen and Hansen, 2018). It also helps teachers establish consistency in the evaluations by designing rubrics containing broad evaluation categories and assigning the assignment's due dates. Students can submit their work, assess their peers anonymously, and view the rubric before submitting the task (Wind, Jørgensen and Hansen, 2018). If the assignment requires self-assessment, students assess themselves only after giving feedback to their peers. Students can edit submissions until the closing deadline and can evaluate their peers' feedback.

The structure of PeerGrade utilises the development of computer technologies. For example, the system allocates reviewers for submissions one by one as they finish their last review to make sure all submissions receive approximately the same number of reviews. Besides, the system displays visual feedback for the teacher only as in Figure 2.2. However, it is unclear what pedagogical theories underpin the system. More importantly perhaps, the system is not suitable for debugging codes as there is no specific code format. The self-assessment may also be less accurate when completed following reviewing peers as students may compare their solution with their peers' solutions rather than considering the criteria of a good performance. However, when the PeerGrade system was evaluated by students (Sharma and Potey, 2018), it was found that nearly 92% of students were satisfied with their peers' feedback, and they found it helpful to them.



**Figure 2.2. PeerGrade system**

Previous peer assessment systems have been explored in regard to five aspects: functionalities, users' interactions, pedagogical rationale, ability to evaluate code, and users' satisfaction. All these systems consist of three phases: a create phase, an assess phase, and a reflect phase. Only the PeerScholar system was built based on pedagogical theories but none of them is suitable for programming assignments that need a specific

format to upload code scripts. All of the systems obtained a high level of satisfaction. However, they were designed with a focus on learning goals, features, and technological capabilities but ignored the most important part of the process – the end-user (student or teachers). Scholars should always think from the user's perspective and strive to research, design and implement solutions that never make the user think about what they need to do in the system (Baxter, Courage and Caine, 2015). As a result, this study designs the prototype based on stakeholders' opinions and their requirements.

Furthermore, these peer assessment systems did not adapt to the growing developments of computer technologies (Ashenafi, 2017); particularly, in the field of learning analytics (Wise and Vytasek, 2017). Peer assessment consists of many datasets that can help students improve and allowing teachers to gain insights into the students' learning. For example, learning analytics can be beneficial for students to create more effective learning environments, enhance the learning experience, and increase learners' retention rates (Mah, 2016). This study explores how peer assessment datasets could be utilised to improve students' and teachers' experience in peer assessment.

Before implementing a peer assessment activity with students, the prototype development should be thought about and a suitable tool to develop the prototype should be found. The following section discusses the prototype development and the selected tool to develop the prototype.

### **Prototype development**

A prototype is an object in the design process with specified characteristics and details (Jensen, Elverum and Steinert, 2017). A prototype can be categorised as a phase in the design process; it can also be described as a tool to develop a product; and as a tool to evaluate a process (Lauff, Menold and Wood, 2019). Prototyping is an important step in the design phase; in fact, according to Wall *et al.* (1992, p. 163), "prototyping is one of the most critical activities in new product development". Human Computer Interaction research has indeed verified the need for prototypes when communicating design ideas to end users. They can improve users' technical understanding of the design space,



explore essential user insights, and aid designers to discover “unknown unknowns” (Jensen, Elverum and Steinert, 2017). Additionally, prototypes can alert of incorrect design assumptions and help users visualise problems (Andreasen and Hein, 1987). Furthermore, prototypes can improve both technical and social skills development (Lauff, Menold and Wood, 2019), enhance design performance (Neeley *et al.*, 2013), gather user feedback (Menold, Jablokow and Simpson, 2017), complement designers’ mental models (Lemons *et al.*, 2010), and improve design outcomes (Bucciarelli, 2002).

Many tools can be used to design a prototype, some tools are suitable for the initial view of a future website/application (low fidelity prototypes) to make sure the content is correct. Other tools create a high-quality interactive prototype or develop HTML/CSS pages (High fidelity prototypes). Since this study aims to allow users to have realistic (mouse-keyboard) user interactions, the prototype is assumed to be much more effective in collecting true human performance data (e.g., simulating a task) and demonstrating actual product users. Therefore, this study selects the Adobe XD program to design a high-fidelity prototype website. The following section outlines the advantages of Adobe XD.

### **Adobe XD tool**

Adobe XD is a platform tool that allows users to design an interactive prototype for a website, mobile application or presentation with professional quality results (Wood, 2020). It allows to preview and share the website for feedback via commenting and annotations (Wood, 2020). Adobe XD is thus very similar to the experience of using the actual website. There are various reasons for using Adobe XD in this study. Firstly, one of the most significant aspects of XD is the prototyping functionality; this feature allows the developer to create a mock-up of the students’ and teachers’ journey and navigate through various pages, so participants can explore the prototype website and navigate from one page to another. Secondly, it is easy to create interactive prototypes of the key pages that are similar to the real website/application without writing any code when designing pages. Thirdly, users can open the prototype in any browser without uploading any application. Finally, XD has a built-in commenting feature that allows users to ask questions or to add feedback on any page. All of these reasons encouraged the use of Adobe XD in this study.

Finally, the researcher thought about the controversial aspects of peer assessment that were raised by other researchers in the literature to determine the study's position of these aspects. The following section outlines the most common controversial aspects of peer assessment.

#### **2.4.11 Controversial issues in peer assessment**

A controversial issue is one that results in disagreement and dispute due to a difference of perspectives. Peer assessment in higher education contains several variables with a range of controversial issues. This section outlines the range of variances for some variables that have been found in the literature (e.g., formative/summative, anonymous/non-anonymous, individual/teamwork). These controversial issues should be discussed with programming teachers and students to determine their attitudes before implementing a peer assessment activity, as the implementation is based on their perspectives of these elements. This study considers the quality of their argument because it leads to a deeper understanding of the issues, and greater respect for decision-making processes. The following section outlines common controversial variables in peer assessments:

1. **Privacy:** This is a sensitive issue within peer assessment, whether a student is an assessor or assessee (Li and Gao, 2016). Although there are no definitive decisions about whether the impact of peers' anonymity is beneficial or harmful, some studies found that students are more comfortable and feel more positive if they are anonymous when assessing peers and that this guarantees honesty and accuracy in a peer assessment (Panadero and Brown, 2017). Rotsaert *et al.*, (2018) have established that the provision of anonymity results in more favourable opinions on peer assessment and makes students more prepared. Moreover, peer pressure can be significantly reduced when students do not know the owner of the work they are marking or the feedback source. In contrast, other studies found that non-anonymity could lead reviewers to take the assessment more seriously, thus generating an accurate assessment (Li *et al.*, 2016). Regarding computer studies, many studies apply anonymity (Sitthiworachart and Joy, 2003; Park *et al.*, 2017; King, 2018), but the

perspectives of programming students and teachers on this has not been explored. Thus, asking programming students and teachers about their views on privacy in peer assessments has been useful in this study.

2. **Official weight:** There is much debate about whether peer grading can be included as part of the student's final grade or should be used as a feedback activity only. This matter may move peer assessment from a simple learning exercise to a form of summative assessment (Panadero and Brown, 2017). It is crucial that students' grades are trustworthy, thus, Panadero and Brown (2017) found that it is better to avoid including peer assessment as part of students' final grades. However, as programming students' and teachers' perspectives are significant, stakeholders' opinions were considered to decide if there is an official weight to students' assessors grading in the peer assessment activity.
3. **Standard used:** Learners require a guideline or a clear marking scheme to carry out an assessment (Ng and Fai, 2017). The student is excited when they are provided with a marking rubric and activity specifications (Kavanagh and Luxton-Reilly, 2016). Thus, marking criteria must be clear so that learners can make fair and accurate judgments (Sitthiworachart and Joy, 2004; Panadero and Brown, 2017). However, holistic peer assessment results are closer to teachers' assessment than category-based assessment (Lejk and Wyvill, 2001). Another controversial issue is participating in creating assessment criteria. According to Fraile, Panadero and Pardo (2017), students might have a chance to better internalise marking criteria by co-creating them. Thus, asking students' and teachers' views about these points would be useful.
4. **Teachers' role:** Teachers should focus on how to best implement peer assessment. Teachers can prepare students' guidelines, manage learning tasks, build marking criteria and provide information on working in groups (Wride, 2017). Panadero and Brown (2017) observe that successful peer assessment calls for more input from the teachers. However, what is the exact role of teachers in peer assessment? Does their role include monitoring reviews provided by students, or managing the peer assessment process? This study examines the role of teachers in the peer assessment activity.

5. **Place and time:** Places and times for peer assessments vary (Topping, 1998). They usually take place during class time to facilitate monitoring by staff (Topping, 1998), but can also occur online. An online peer assessment can increase learning effectiveness; it gives students flexibility in terms of when and where to do the assessment, and promote students' attitudes toward peer assessment by utilising anonymous online marking and feedback (Wen and Tsai, 2006). This study discusses the suitability of peer assessment as an online application with relevant stakeholders. Further, it discusses when to use peer assessment in the semester, in the beginning, middle, or at the end.
6. **Peer configuration:** Students can review their peers' work either individually (individual peer assessment) or in groups (collaborative peer assessment). Individual peer reviews are more common than collaborative reviews (Indriasari, Luxton-Reilly and Denny, 2020). In individual peer assessment, students can take their time to analyse and critique their peers' work. In contrast, in a collaborative peer assessment, students can interact with each other during the assessment process. This study investigates students' and teachers' preferences in regard to peer configuration.

## 2.5 Summary

This chapter started by describing the main issues related to introductory programming courses that cause dropout. This discussion suggested that educational tools to engage students in the learning process in order to enhance their learning and thus increase students' retention in introductory programming courses are required. Previous research has shown that the use of collaborative learning strategies in the teaching of programming reduces students' dropout and improves students' learning and academic performance. Many collaborative learning strategies are suited to introductory programming course, but the focus of this study is peer assessment. This chapter then described learning theories related to peer assessment that can help make more informed decisions around the design and development of a peer assessment. This was followed by a discussion of the definition, types, methods, benefits, and barriers of peer assessment. However, there were a lack of programming teachers' and students' perspectives in many aspects in peer

assessment. Further, this chapter showed the importance of formative assessment for learning and category-based assessment to guide students during the assessment. This chapter presented several studies that focused on a formative assessment and used a category-based assessment. However, most of the studies focused on rubrics with their criteria and categories, with less attention given to quality scales.

Additionally, this chapter explored the accuracy of peer assessment and its impact on students' performance, particularly in introductory programming courses. There is an inconsistency between the correlation of students' assessments and teachers' assessments in some studies, which indicates the importance of measuring the correlation in detail, and there is little research on the impact of peer assessment on students' programming skills. Furthermore, most studies have focused on quantitative methods that examine the accuracy of peer assessments but there remains a lack of in-depth qualitative investigation informed by the perspectives of students and teachers toward the implementation of a peer assessment. Additionally, a distinct absence of studies of Saudi Arabia, particularly studies that used peer assessment in introductory programming courses, was noted. Moreover, since one of the main objectives of this study was designing a prototype, this chapter reviewed and evaluated some existing peer assessment systems. This chapter has also shown that current peer assessment activities have not yet benefited from technologies in computer science, such as learning analytics. Finally, the chapter outlined some controversial issues in peer assessment, such as privacy, grading, peer configuration, and marking guide. Programming teachers and students should determine their position on these issues to implement peer assessment in introductory programming courses based on their perspectives.

This chapter primarily summarised previous research of peer assessment in introductory programming courses. It aimed to identify areas of controversy and highlighted gaps in the current research. The following chapter describes the methods that were used to collect and analyse the data required to address the research objectives outlined in Chapter 1, section 1.5.

## **Chapter 3. First phase of the study**

### **3.1 Introduction**

Every researcher aspires to develop research strategies that answer research questions and yield results in reliable and valid ways. The decision on the appropriate approach is reached by considering a range of research methods that can help to answer the research questions. This study explores an effective approach to integrating a peer assessment activity as a learning process into introductory programming courses. The study was divided into two phases to achieve the research aim. This chapter focuses on the first phase and outlines the mixed-method approach used to achieve the study's objectives, and the process to acquire ethics approval prior to the research. This chapter also discusses the research methods used in the first phase. It includes questionnaires and interviews methods to ascertain programming students' and teachers' perspectives toward peer assessment. It also includes experimental methods to find a suitable marking guide, to evaluate students' accuracy in the peer assessment, and to assess the activity's impact on students' performance. The chapter explains the application of each method with more detail as well as the research instruments used to provide the results. Finally, it discusses the validity and reliability of the data to ensure the study is replicable.

### **3.2 Mixed-methods approach**

There is no particular research method that is intrinsically better than another method. Many authors, such as Newby (2010) or Cohen, Manion and Morrison (2012), have claimed that using a combination of research methods in educational research helps take advantage of the most helpful features of each method. This mixed-methods approach is defined as “the collection or analysis of both quantitative and/or qualitative data in a single study in which the data are collected concurrently or sequentially, are given a priority and involve the integration of the data at one or more stages in the process of research” (Hanson *et al.*, 2005, p. 224). Mixed-methods research consisting of qualitative and quantitative methods presents a robust paradigm choice that usually offers helpful, complete, balanced, and informative research results (Johnson, Onwuegbuzie and

Turner, 2007). Consequently, as part of the quantitative approach to part of this study, comprehensive information regarding students' and teachers' viewpoints of peer assessment can be captured. In addition, evidence for how effective the use of peer assessments with first-year students can be obtained.

The main rationale for choosing the mixed-methods approach in this study is the complementarity feature (Greene, Carcelli and Graham, 1989), which refers to clarifying and improving findings from one method with the results from another method. Integrating quantitative and qualitative methods is suitable for providing a holistic insight into the topic under investigation. A mixed-methods approach can also enhance the generalisability of the results as qualitative research often has a smaller sample size and is thus not generalisable, but findings with a large sample size using quantitative methods can be generalised (Creswell and Clark, 2007). Besides, a mixed-methods approach can also help to verify the results (Greene, Carcelli and Graham, 1989); that is to say, if the qualitative and quantitative data converge, this strengthens the validity of the results. This process is called triangulation (Cohen, Manion and Morrison, 2012). A mixed-methods approach was thus used in this study to increase the generalisability and validity of the findings.

However, there are some limitations in mixed-method approaches. According to Bryman (2008), one problem with a mixed-methods approach is the structure, which might hinder the integration of both methods. A further issue is that the mixed-methods approach may be more costly than the single-method approach, considering the time required for collecting and analysing data and the cost of materials (Bryman, 2008). Besides, when different methods are combined, they may not always be aligned which might mean that the result is not reflected in how the mixed-methods approach is used, or that the practice does not match the rationale (Bryman, 2008). Despite these concerns, Creswell and Clark (2007) argue, different paradigms can be employed in the mixed-methods approach as long as the researcher honours each paradigm and is clear about when it is to be employed. While qualitative and quantitative approaches are sometimes presented as if they were in binary opposition to one another, they can also be used to complement each

other (Cohen, Manion and Morrison, 2012) (e.g., illustration, development, explanation of the findings derived from one method with findings from the other method, seeking elaboration). Therefore, the decision to blend quantitative and qualitative methods was taken, as this approach presented the best fit for the study and the research questions.

The mixed-methods approach consists of four major types of designs: the triangulation design, the embedded design, the explanatory design, and the exploratory design (Creswell and Clark, 2007). According to Creswell and Clark (2007), there are key factors that help researchers select a type of mixed method design: What will the timing of the quantitative and qualitative methods be? What will the weighting of the quantitative and qualitative methods be? And how will the quantitative and qualitative methods be mixed? Researchers must answer the questions to select a suitable design. The explanatory design was chosen to collect data in this research because it is ideal for studies that require qualitative data to expand upon the results of the quantitative data analysis (Creswell and Clark, 2007). Methods were applied in a sequential time (i.e., quantitative methods have been used sequentially, then qualitative methods), and the weight of the data were unequal as there were more qualitative data than quantitative data. The next section discusses the explanatory design that has been followed in this study.

### ***3.2.1 Explanatory design***

The strengths of an explanatory design lie in the following: (1) the researcher organises the quantitative and qualitative methods in isolated phases and gathers only one type of data at a time; (2) the results can be written in two steps, making them simple to write and offering a clear explanation for readers; and (3) this design demands quantitative research because it often begins with a robust quantitative method (Creswell and Clark, 2007). The explanatory design applied in this study then consists of three distinct phases. The first phase involves collecting and analysing quantitative data. Then, using qualitative data as a second phase to interpret the results from the quantitative data, so that each method is isolated from the other during the process of collecting and analysing data. All datasets are then integrated into the interpretation phase, i.e., the third phase, to paint a complete picture of the problem. There are two models an explanatory design: follow-up and

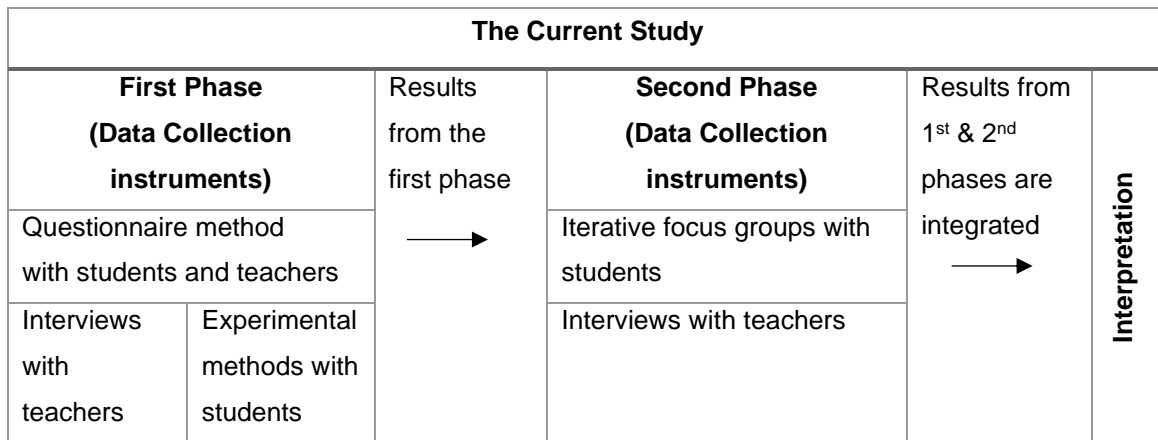


participant selection (Creswell and Clark, 2007). Since one phase (e.g., questionnaires) is followed by another phase (e.g., interviews), the design choice was the follow-up explanations model. The following section outlines the research design of this study and shows how the explanatory design has been employed.

### **3.3 Research design**

As mentioned earlier in the chapter, this research adopted a combination of quantitative (questionnaires and experiment methods) and qualitative (interviews and focus groups) methods to explore the best approaches for integrating a peer assessment activity, as a learning process, into introductory programming courses. If data are collected in the right way and implemented according to students' needs, this thesis argues, the experience of peer assessment with first-year students could improve learning outcomes. Therefore, the study was divided into two phases: the first phase utilised mainly quantitative methods and was designed to explore what stakeholders' perspectives are, how accurate students in peer assessment are, and the activity's impact on students' performance. The questionnaire method helped identify the perspectives of first-year students and programming teachers. Interviews with programming teachers then expanded the collected data and helped clarify some aspects that were assessed to conduct the peer assessment activity (e.g., building a rubric based on teachers' perspective). Furthermore, an experimental method was used that found the best marking guide form and examined students' accuracy by finding the correlation between teacher assessment and student assessment. Two marking guides were prepared for the experimental methods: 1) a marking scheme that comprised a set of criteria without descriptive levels but with Likert scales; and 2) a rubric that consisted of a set of criteria with several descriptive levels and marking scales. All of them are evaluation tools used to promote the compatible programming code with learning objectives, or learning standards, to measure their attainment against a set of criteria. Then, another experiment method helped find the effect of the peer assessment intervention on programming skills for those who took part. The results gained from the first phase provided a general insight into the peer assessment activity with first-year students in a programming assignment. Consequently,

the primary milestone of the first phase was developing an initial prototype website that presents a way of using peer assessment for first-year students containing the main elements of peer assessment. However, the prototype needed evaluation, and some results needed more elaboration which is why qualitative methods were used to clarify findings and boost the output of the quantitative results. The second phase then aimed to evaluate, refine and release the final design of the prototype. Thus, the second phase contained iterative focus groups discussions with programming students and interviews with teachers to determine functional requirements, evaluate the peer assessment prototype, and release it. Figure 3.1 illustrates the design that was followed in this study.



**Figure 3.1. Research design of the current study**

In short, this study integrated multiple methods to provide a comprehensive answer to the research problem. This chapter outlines the first phase of the methods used in this study, and later this thesis outlines the second phase with its results. Before explaining any method, it is essential to outline the procedure of the ethical approval that has been obtained to conduct this study.

### **3.3.1 Ethics procedures**

Research ethics is an important term in any research. It includes the application of fundamental ethical principles to a range of subjects – from the design to the implementation of the research (Anderson and Arsenault, 1998). Drake (2010) asserts that attention must be paid to ethical considerations in academic research because of the

public sensibilities of the limits of investigation and concerns regarding data protection and human rights. This is the ethical procedure which was followed before collecting any data:

1. Newcastle University ethics policy and procedures are posted on the University's website, commensurate to the university vision. First, the Newcastle University Online Ethics Form was completed. The form contains a series of questions that help the researcher identify whether the project is 'low risk' or 'high risk', the latter requiring further formal ethical review by a research ethics committee.
2. The form was submitted, and then the researcher received a notification from the committee based on the answers. When the university is satisfied that the project meets its ethical expectations, it is approved. Thus, a permission from the research ethics committee at Newcastle University to conduct this research was received (See Appendix A).
3. Furthermore, because the researcher was working at Princess Nourah bint Abdulrahman University (PNU), an ethical approval from PNU needed to be obtained. The process of ethical approval at PNU is similar to that at Newcastle University. A 'Facilitate the Task of Researcher' application with the research proposal and research instruments was submitted to the Deanship of Scientific Research.
4. The approval to conduct this study and use its instruments was received from the PNU. This approval facilitates conducting studies with other Saudi universities, as this approval was sent to the specified university and thus obtaining the approval of that university.

Newcastle University supports researchers by providing some examples of project documentation, which can be used as a guide for questionnaires, interviews, focus groups, and experiments. For example, participant information sheet and consent form were used. The sheet presents participants with information about the study so that they are able to make an informed choice and decide if they want to participate. If they do, they sign the consent form. The participant information sheet and consent form in this study were created based on the examples on the Newcastle University website (Appendix B).

### **3.3.2 Determining the target samples**

Sampling allows researchers to obtain adequate data to answer the research questions without having to query the whole population – thus saving time and money. The method of sampling followed in this study was stratified random sampling (Nachmias and Nachmias, 1996), dividing a population into smaller groups based on universities they belong to, then selecting random samples from computing schools. Stratified random sampling provides better coverage of the population because the researchers can control the subgroups to ensure all of them are represented in the sampling. Since this study aimed to integrate a learning activity into programming courses, the most important population are students studying introductory programming courses and teachers who are teaching programming courses. This study divided the population into two main groups: samples from Saudi universities who are unfamiliar with peer assessment, and samples from Newcastle university who are familiar with peer assessment. After that, students and teachers were randomly selected to participate in the study. More details about the sample selection mechanism are explained within the discussion of each method.

### **3.4 First phase of the research**

The first phase mainly focused on quantitative methods, i.e., questionnaires and experiments, and is designed to achieve three goals; to discover students' and teachers' perspectives toward peer assessment in introductory programming courses; to investigate the correlation between students' and teachers' assessment; and to examine the impact of peer assessment on students' programming performance. However, the qualitative method (interviews) was used in the first phase to understand teachers' viewpoints and build the rubric tool based on them. The following sections outline each method used in the first phase.

#### **3.4.1 Questionnaire method**

A questionnaire has many features: it allows the economical collection of certain information, and it is a valuable method for investigating frequencies in the sample (Baxter, Courage and Caine, 2015). Furthermore, it allows anonymity and arguably

increases the rate of response, enhances the answers' reliability, and can be distributed to large numbers of participants simultaneously, therefore saving the researcher time and effort (Baxter, Courage and Caine, 2015). Consequently, the use of questionnaires was the initial method to obtain general perspectives from stakeholders on peer assessments in introductory programming courses in this study in an attempt to address the first research question: How do programming students and teachers perceive peer assessments in introductory programming courses? The questionnaires were designed to measure the diversity of viewpoints about peer assessment activities between programming students and teachers.

### **Study setting**

The students' questionnaire had 23 questions concerning peer assessment practices in programming assignments. The questionnaire was organised around five blocks: (1) the demographic data, (2) the benefits of peer assessment; (3) the challenges of peer assessment; (4) suitable methods for applying peer assessments in programming courses, with five sections: marking criteria, privacy, online assessment, teamwork assessment, and grading; and (5) open-ended questions about what will encourage students to do peer assessments, and concerns that would prevent students from performing peer assessments in programming courses. For the second, third and fourth blocks a 5-point Likert scale was used to show how well the statement described them (strongly disagree, disagree, neutral, agree, strongly agree). The Jisc website (Jisc, 2021) was used to create the online survey. It was estimated that the questionnaire would take approximately 15 minutes to complete. Since the samples were from different countries, the questionnaire was developed in English and Arabic based on the native languages of participants to prevent misinterpretation of the questionnaire items. The English version was developed first, the Arabic version was then translated by a professional translator to make each item in the original and the translation equivalent to each other. Table 3.1 shows the English version of the students' questionnaire. Table 3.2 shows the questionnaire that was distributed to programming teachers; it is similar to the students'

questionnaire but contains elements of marking criteria / rubrics that should be considered while constructing the rubric that guide students when assessing their peers.

A pilot test of the questionnaire was conducted prior to the main study. First, two faculty staff members in the computer science college reviewed the questions. This input was used to revise some of the items. Ten computer science students from PNU then filled out the revised questionnaire to simulate the distribution of the online questionnaire.

<b>Students' Questionnaire</b>	
<b>Demographic data</b>	University
	Gender
	Subject
	Year of study
	Peer assessment experience
<b>Benefits of peer assessment</b>	Peer assessment allows me to learn how others think and solve the assignment.
	Peer assessment allows me to compare my solution with other solutions.
	When assessing peers, I will evaluate myself as well.
	Peer assessment helps to understand programming assignments better.
<b>Challenges of peer assessment</b>	I feel pressure when assessing peers.
	I do not think I'm qualified to assess peers in programming assignments.
	I can't trust someone at the same level to assess my assignment.
	Not all colleagues are able to assess their peers.
<b>A suitable method to apply peer assessment</b>	The marking criteria remind me what I must complete for the assessment.
	The criteria should be detailed so the grade can be easily determined.
	Students and teachers can participate in creating the marking criteria.
	I prefer to assess my peers according to my understanding and my standards.
	I think it is important that feedback and grades are submitted anonymously.
	Online peer assessment is an effective way to assess programming assignments.
	Evaluating peers' work individually is better than a collective discussion between assessors.
	I prefer assessing my peers by providing feedback without giving a grade.
<b>Open-ended questions</b>	What are the things that would encourage you to perform peer assessment with programming assignments?

	What are the concerns that would prevent you from performing peer assessment in programming courses?
--	--

**Table 3.1. Programming students' questionnaire about peer assessment**

<b>Teachers' Questionnaire</b>	
<b>Demographic data</b>	University
	What work experience do you have?
	Have you used peer assessment with first-year students in programming courses?
	How long have you been using peer assessment in a programming course?
	If you have used peer assessment, how would you consider your experience?
<b>Benefits of peer assessment</b>	Peer assessment creates active participation for first-year students in a programming course.
	Peer assessment makes first-year students more aware and responsible.
	Peer assessment expands students' knowledge by looking at different solutions to the same problem
	Peer assessment provides timely support to first-year students.
<b>Challenges of peer assessment</b>	I do not like the idea of someone who is not qualified and is not an expert assessing the assignment.
	Peer assessment causes problems for the teacher's authority.
	Peer assessment is not reliable.
	Not all first-year students are able to assess their peers.
<b>A suitable method to apply peer assessment</b>	I must provide first-year students with a rubric to assess their peers.
	The scale of rubric must be descriptive details to each criterion
	Students should participate in creating a rubric.
	The assessee and assessor should be anonymous.
	Online peer assessment is an effective way to assess programming assignments.
	Evaluating peers' work individually is better than a collective discussion between assessors.
	Peer assessment should be a feedback activity rather than a graded practice.
<b>Open-ended questions</b>	What would encourage you to perform peer assessments with first-year programming students?

	What concerns during peer assessment would prevent you from performing peer assessment in programming courses?
	What elements of the rubric should be considered during peer assessment?

**Table 3.2. Programming teachers' questionnaire about peer assessment**

Thus, questions were created for both students and teachers to determine their perspectives on peer assessment in introductory programming courses, with slight differences in some questions based on the participants' roles. The following section outlines the data analysis tools that were then used to analyse the data.

### **Data analysis tools**

The questionnaire data were analysed with the assistance of the Statistical Package for the Social Sciences (SPSS). Likert scale is often seen as ordinal data; each level on the scale is assigned a numeric value. Thus, participants' responses were assigned numerical codes so that they could be easily entered into the computer for analysis: strongly agree=5, agree=4, neutral=3, disagree=2, and strongly disagree=1. After that, data were checked for errors (e.g., reviewing the code that is out of the range). Once the data file was cleaned, the process of inspecting data and exploring the nature of variables was conducted.

Scholars of Statistics have for a long time grappled with the conflicting issues of using parametric tests for Likert responses. This is because parametric tests require data of interval or ratio type (Jamieson, 2004), and means and standard deviation have unclear meanings when used in Likert scale responses (Sullivan and Artino, 2013). However, nonparametric tests compared to parametric tests are less powerful and require a bigger sample size (Sullivan and Artino, 2013). Therefore, parametric tests have been used for data that can calculate the mean and standard deviation (e.g., benefits and challenges of peer assessment). In contrast, nonparametric tests and median have been used with data that have unclear meaning with mean and standard deviation (e.g., best methods to apply peer assessment). The following sections outline the main tools:



1. **Descriptive statistics:** Descriptive statistics are defined as quantitative descriptions of the most important elements of a collection of information, or as quantitative descriptions themselves (Pallant, 2001). They can help us simplify large amounts of data sensibly by reducing the amount of data to a more concise summary. The aim of using descriptive statistics was to answer the first question which intended to provide information about the respondents' trends, and to describe overall opinions toward peer assessment in introductory programming courses. To this end, the following descriptive statistical methods were used: Frequency, Central Tendency, and Variation.
2. **Correlation coefficients:** The correlation technique is a statistical tool used to evaluate the strength and direction of the linear relationship between quantitative variables (Pallant, 2001). There are numerous types of correlation coefficients; in this study, Pearson's ( $r$ ) is used to describe the direction of the relationship (if any) between awareness of benefits and fear of challenges when applying peer assessment in introductory programming courses.
3. **Regression:** Regression can be used to discover the relationship between one continuous dependent variable and a number of independent variables or predictors (Pallant, 2001). Linear regression was used to examine the relationship between benefits and challenges in students' and teachers' questionnaires. Does greater awareness of the benefits reduce the fear of the challenges? Or does it increase fear of the challenges?
4. **Welch's t-test:** Welch's t-test (also called unequal variances t-test) is a two-sample location test that can be used to test the hypothesis that two populations have equal means. It is suitable for unequal sample sizes (Zimmerman and Zumbo, 1993). The Welch's test was used to compare the means of two different groups - students and teachers - in two aspects: benefits of peer assessment and challenges of peer assessments because the sample size of students is much larger than the sample size of teachers.
5. **Mann-Whitney U test:** The Mann-Whitney U test is used when there are differences between two independent groups on a continuous measure (Pallant, 2001). This test

is the non-parametric alternative to the t-test for independent samples, and it can be used with unequal sample sizes. The Mann-Whitney U test compares medians instead of comparing the means of two groups (Pallant, 2001). The Mann-Whitney U was used to answer the following question: Is there a significant difference in the responses between teachers and students to the question point “suitable method in applying peer assessment section”?

6. **Cronbach Alpha:** The Cronbach Alpha offers a coefficient of inter-item correlations, that is, the correlation of every item with the sum of all the other related items, and is suitable for multi-item scales (Cohen, Manion and Morrison, 2012). Cronbach’s Alpha coefficient is the most commonly used indicator of internal consistency, which indicates how closely related a set of items are as a group (Pallant, 2001). It is an alternative measure of reliability and the degree to which the items that make up the scale ‘hang together’.
7. **Cohen's Kappa:** Cohen's coefficient Kappa is the most popular measure that is used for inter-rater reliability (Straub and Gefen, 2004). It measures the level of agreement among raters for categorical or nominal scales. The Kappa statistic varies from 0 to 1. The closer to 1, the higher the agreement. Generally, a Kappa value  $< 0.4$  represents poor, Kappa values of 0.4 to 0.6 are considered moderate, and a Kappa value of  $>0.7$  represents excellent agreement.
8. **Open-ended questions:** An affinity diagram (Baxter, Courage and Caine, 2015) has been used to organise ideas into groups according to their affinity or similarity for data derived from open-ended questions. Firstly, significant ideas for each question were recorded in separate sticky notes. Next, all ideas that seemed to be related were grouped together. Thirdly, the groups were reviewed by a volunteer researcher. Finally, groups were combined into ‘supergroups’ if appropriate.

In summary, these are the statistical tools that have been used on the data derived from the questionnaire to determine patterns, relationships or trends in data and then interpret the results.

### **3.4.2 Interview method**

The decision to use interviews as a method to collect data is in line with Ely *et al.* (2003, p. 4) who argue that “qualitative researchers want those who are studied to speak for themselves, to provide their perspectives in words and other actions”. Since lack of teachers’ perspectives regarding implementing peer assessment in introductory programming courses, direct contact with teachers can help to understand their perspectives. To this end, they were asked to answer the following question: what is your view towards peer assessment in introductory programming courses? The interview method is valuable in terms of assessing teachers’ thoughts and experiences as their attitudes are significant when integrating peer assessment into programming courses effectively. The purpose of using interviews was the same as using questionnaires: to gather information from programming teachers about peer assessment in introductory programming courses. The interview was also designed to allow for the development of a marking scheme form for the peer assessment. Interviews were conducted face-to-face to allow the researcher to discuss and gather detailed information about participants’ attitudes.

#### **Study setting**

As stated in Bryman (2008), there are many different interview methods: structured, semi-structured, unstructured, intensive, standardised, in-depth, qualitative focused group, and life history. The semi-structured interview has been used in this study; it is an interview method in which some questions are closed, and some are open-ended. The interviews included 15 questions concerning peer assessment activity in programming courses. The interview questions were organised around four themes: (1) Personal perspectives on programming courses; (2) Benefits and challenges of peer assessments in programming courses; (3) Suggestions for a suitable method to apply peer assessment in programming courses; and (4) Questions in relation to the data collected in the peer assessment. Table 3.3 shows the questions of discussion in the interview. Before conducting the interviews, two volunteer experts from the faculty staff checked the questions. They are lecturers at the Computer Science and Information College, PNU. They gave feedback before

collecting the data, and their feedback was taken into consideration during the actual interview.

Topic	Question	Duration	Goal
<b>Teachers' perspective on programming courses</b>	From your experience, what are the difficulties of first-year programmers?	~15 min	Collect teachers' experiences on programming courses
	Do you use peer assessment as a practice strategy in your courses? What is your experience?		
<b>Possible benefits and challenges of peer assessment</b>	What do you think are the reasons to incorporate peer assessment as a part of the learning process in programming courses?	~ 7 min	Let users think about the benefits of peer assessment
	Do you think peer assessment is practical with first-year students in programming assignments?		
	What are your beliefs about the challenges of peer assessment with first-year students? How to cope with these challenges?	~ 7 min	Identify issues of peer assessment
	Are there any barriers that could hinder using peer assessment in programming courses?		
<b>A suitable method to apply peer assessment</b>	What is the role of the teacher during peer assessment?	~15 min	Identify teachers' opinions about aspects of implementing peer assessment
	What are your preferences regarding the privacy of peer assessment?		
	Should peer grading be included as part of the student's final grade?		
	What elements of the marking criteria/rubric should be considered during peer assessment?		
	What is your opinion about the suggested rubrics? <i>(Display two forms of rubrics)</i>		
	What scales can be used?		
	Do you think there are conditions when applying peer assessment in programming courses?		

Topic	Question	Duration	Goal
<b>Data analytics in peer assessment</b>	Can teachers use student feedback from peer assessments to measure the quality of educational content, students who struggle with a particular topic, or identify students at risk of attrition?	~7 min	Gain knowledge about data analytics
	Does peer assessment involve pointers to students' progress at an early stage?		

**Table 3.3. Programming teachers' interview questions about peer assessment**

### **Data analysis tool**

Thematic analysis has been used to identify patterns in the data. Each interview took approximately 40–60 minutes; the audio recordings were then transcribed. There are three ways to transcribe audio recordings into text: verbatim, edited, or summarised (Baxter, Courage and Caine, 2015). Verbatim transcripts accurately capture what was said by both the interviewer and participants, including ‘ahs’, ‘ums’, and misstatements; edited transcripts naturally do not consist of word crutches or misstatements; and summarised transcripts typically include an edited and condensed version of the questions asked or topics mentioned by the interviewer because these are known in advance, along with the participants’ comments (Baxter, Courage and Caine, 2015). In this study, ‘summarised transcripts’ were produced because teachers’ perceptions do not need to be precise. Due to the sampling from two different countries, transcripts were generated in English and Arabic. After the transcription process, the transcriptions were reviewed in regard to accuracy and then reread to identify the main themes. Lastly, codes were developed in English to unify the themes. An initial, open, axial, and selective coding was used following the techniques described by Saldaña (2016). More detail on the coding process and the procedure used for the thematic analysis are outlined in the discussion of the second phase of the study.

### **3.4.3 Experimental method**

Experimental research is research that adheres to scientific research methods. It involves a hypothesis; a variable that the researcher can manipulate; and variables that can be

compared, calculated, and measured (Harland, 2015). Experimental methods are especially useful in addressing evaluation questions about the impact and effectiveness of activities (Gribbons and Herman, 1997). Additionally, when a comparative analysis of data takes place, it is more likely that the findings relate to a particular program or innovation rather than being a function of extraneous variables or events. In this study, an experimental method was used to answer the second research question: Are first-year students who participated in peer assessment more likely to perform significantly better on programming skills than those who do not? This method measured the accuracy of first-year students' judgements during a peer assessment activity by finding the correlation between peer and teacher assessment, and it measured the impact of peer assessments on first-year programmers' performances.

There are many types of experimental methods; in this study, two types were applied, pilot-experiment and pseudo-experiment. Pilot-experiment research affords the least amount of control because it deals with phenomena happening in natural environments; this type tries to find correlations among some variables without manipulating them (Mildner, 2019). A pseudo-experiment research is often used in applied studies, where it allows additional control of independent variables and includes manipulation; however, it lacks randomness in the selection of subjects (Mildner, 2019). Each of them has a different aim and samples, but they complete each other as described below.

### **Pilot-experiment method**

The pilot-experiment method is used before conducting the pseudo-experiment. A researcher who uses the pilot-experiment method often tries to evaluate the effect of a social program on a small group of people before they obtain funding and invest time in the actual experiment (Thyer, 2001). In this study, the pilot-experiment method was used to measure the accuracy of the peer assessment activity and to explore what constitutes a suitable category-based assessment for peer assessment in programming courses. As observed in the literature (Sitthiworachart and Joy, 2008; Hamer *et al.*, 2009; Park *et al.*, 2017), criteria and scales for marking schemes in programming assignments are either detailed or brief. In order to find a suitable design for a marking scheme form with a higher

correlation between peer assessment and teacher assessment, the pilot-experiment method was deemed valuable to this study. From the Human Computer Interaction (HCI) perspective, it is an effective way to compare two or more designs and determine which one performs better. One of the standard tests is A/B testing; this is a user experience research method that contains two variants, A and B. By analysing participants' performance in one design against another design, researchers can determine which of the two variants is more effective (Baxter, Courage and Caine, 2015). To analyse data, statistical analysis can be used to find significant differences among the variants. According to Baxter, Courage and Caine (2015), some matrices can then be considered in the summative evaluation of the data, such as 1) length of time to complete a task or experiment; 2) the number of errors made when finishing a task or conducting a study; 3) participants' rate of finishing a task/experiment successfully; 4) participants' satisfaction with a specific task or with the product as a whole; and 5) conversion measure of whether or not participants effectively finished their desired task.

In this study, two marking guides were designed – A, a rubric; and B, a marking scheme – to establish which one was more effective for first-year programmers. The forms were measured from the following perspectives: (1) Which method gives the best correlation between student and teacher assessment, a rubric form or a marking scheme form? And (2) How long does it take to complete a task or experiment? The experiment employed two designs (the rubric and marking scheme) and applied the experiment with the same participants' group; one design being applied after another. The experiment was designed as a pilot study because this pilot-experiment was designed to determine the appropriate marking guide for peer assessment with small samples, to identify the potential problems of the marking guide tool. This allowed the researcher to collect data in preparation for a more extensive study.

### ***Study setting***

Two marking designs were created: the first was a marking scheme, which comprised a set of criteria without descriptive levels but with Likert scales (Table 3.4). The second was a rubric, which consisted of a set of criteria with several descriptive levels and marking

scales (Table 3.5). Both marking designs were divided into five categories: correctness, structure, clarity, layout, and exceptions. Correctness covers three areas: correct running, complete specification, and handling errors. Structure covers two points: selection of variable types and use of abstract methods. Clarity contains the names of the variables and language capabilities. Layout includes indentations and comments, and exception looks at exit status. The differences between the dimensions are evident in the scales and levels of detail. For example, the marking scheme has five scales: yes, partly, no, not applicable, and I don't know. The rubric has six scales: unsatisfactory, satisfactory, good, excellent, not applicable, and I don't know—with more detail for each criterion. Moreover, three open-ended questions were added to all forms: What did the programmer do right? What did the programmer do wrong? And how can it be improved in the future? The last question has five dimensions (correctness, structure, clarity, layout, and exceptions) to guide students' feedback and avoid generalisation in the student's reviewer's feedback.

There are ten criteria in each rubric or marking scheme. If the reviewer's choice corresponded to the teacher's choice, the student was awarded one score on this criterion. In some situations, the tutor gave half a point for the choice nearest to the tutor's choice. There are also detailed scores for each category. For instance, the total score for the correctness category is out of 3; the total score for structure, clarity, and layout is out of 2; and for exceptions, the total score is 1. Regarding the sample answers that students should review, the teacher who teaches the programming module chose random historical anonymous sample answers to a question from an earlier academic year that appeared on an actual quiz in a Java programming course at PNU to simulate reality in assessing peers. An example of a selected answer is shown in Figure 3.2. All participants evaluated the same set of random answers to the same question.



Category	Criterion	Yes	Partly	No	Not applicable	I don't know
<b>Correctness</b>	I think the program will run correctly.					
	The program produces the correct output as specified in the problem description.					
	I think the program is free from errors.					
<b>Structure</b>	The choice of variable types and data structure is correct (e.g., array/ linked list).					
	The code used abstracted methods (e.g., loops used to repeat coding patterns).					
<b>Clarity</b>	The variables' names make it clear what they are used for.					
	The code is understandable and uses appropriate language.					
<b>Layout</b>	The code is indented helpfully and consistently.					
	Code comments are used in different parts (e.g., header, and line comments).					
<b>Exceptions</b>	Exit status are employed moderately, not as a tool for every job.					
<b>Reviewer's feedback</b>						
<b>What did the programmer do right?</b>						
.....						
.....						
.....						

**What did the programmer do wrong?**

.....

.....

.....

**How it be improved in the future?**

**Correctness:** .....

**Structure:** .....

**Clarity:** .....

**Layout:** .....

**Exceptions:** .....

**Table 3.4. Marking scheme used in the pilot-experiment method**

Category	Unsatisfactory	Satisfactory	Good	Excellent	Reviewer
<b>Correctness</b>	I think the program would not run at all.	I think the program will run but mostly incorrectly.	I think the program will run, but slightly incorrectly.	I think the program will run the correct output for legal input.	<input type="radio"/> <i>Unsatisfactory</i> <input type="radio"/> <i>Satisfactory</i> <input type="radio"/> <i>Good</i> <input type="radio"/> <i>Excellent</i> <input type="radio"/> <i>Not applicable</i> <input type="radio"/> <i>I don't know</i>
	Completed less than 50% of the specification.	Completed between 50 and 69% of the specification	Completed between 70 and 89% of the specification	The program satisfies between 90 and 100% of the specification.	<input type="radio"/> <i>Unsatisfactory</i> <input type="radio"/> <i>Satisfactory</i> <input type="radio"/> <i>Good</i> <input type="radio"/> <i>Excellent</i> <input type="radio"/> <i>Not applicable</i> <input type="radio"/> <i>I don't know</i>
	I think the code violates more than 50% of errors.	I think code violates between 50 and 69% of errors.	I think code violates between 70 and 89% of errors.	I think the program is free from errors.	<input type="radio"/> <i>Unsatisfactory</i> <input type="radio"/> <i>Satisfactory</i> <input type="radio"/> <i>Good</i> <input type="radio"/> <i>Excellent</i> <input type="radio"/> <i>Not applicable</i> <input type="radio"/> <i>I don't know</i>
<b>Structure</b>	Less than 50% of the variable types and data	Between 50 and 69% of variable	Between 70 and 89% of variable	Appropriate choice of variable	<input type="radio"/> <i>Unsatisfactory</i> <input type="radio"/> <i>Satisfactory</i> <input type="radio"/> <i>Good</i>

	structure (e.g., array/ linked list) are chosen correctly.	types and data structure are chosen correctly.	types and data structure are chosen correctly.	types, globals, parameters and data structure.	<ul style="list-style-type: none"> <li>⊖ <i>Excellent</i></li> <li>⊖ <i>Not applicable</i></li> <li>⊖ <i>I don't know</i></li> </ul>
	Code does not use abstracted methods (e.g., loops).	Code uses abstracted methods in some situations.	Code uses abstracted methods in most situations.	Code uses abstracted methods in all situations.	<ul style="list-style-type: none"> <li>⊖ <i>Unsatisfactory</i></li> <li>⊖ <i>Satisfactory</i></li> <li>⊖ <i>Good</i></li> <li>⊖ <i>Excellent</i></li> <li>⊖ <i>Not applicable</i></li> <li>⊖ <i>I don't know</i></li> </ul>
<b>Clarity</b>	Many global variables and ambiguous variables are missing.	Few global variables, few ambiguous naming.	No global variables, unambiguous naming.	Variables' names make it clear what they are used for, and global variables are used sparingly and appropriately.	<ul style="list-style-type: none"> <li>⊖ <i>Unsatisfactory</i></li> <li>⊖ <i>Satisfactory</i></li> <li>⊖ <i>Good</i></li> <li>⊖ <i>Excellent</i></li> <li>⊖ <i>Not applicable</i></li> <li>⊖ <i>I don't know</i></li> </ul>
	Code is incomprehensible, appropriate language capabilities not used.	Code hard to follow in one reading; poor use of language capabilities.	Language capabilities are used but are hard to follow in one reading.	Well-formatted, understandable code; appropriate use of language capabilities.	<ul style="list-style-type: none"> <li>⊖ <i>Unsatisfactory</i></li> <li>⊖ <i>Satisfactory</i></li> <li>⊖ <i>Good</i></li> <li>⊖ <i>Excellent</i></li> <li>⊖ <i>Not applicable</i></li> <li>⊖ <i>I don't know</i></li> </ul>
<b>Layout</b>	No use of white space (e.g., indentation, blank lines).	White space used but inconsistent indentation.	White space makes program fairly easy to read.	Indentation broken down into appropriate smaller logical units.	<ul style="list-style-type: none"> <li>⊖ <i>Unsatisfactory</i></li> <li>⊖ <i>Satisfactory</i></li> <li>⊖ <i>Good</i></li> <li>⊖ <i>Excellent</i></li> <li>⊖ <i>Not applicable</i></li> <li>⊖ <i>I don't know</i></li> </ul>

	No comments at all.	Wordy, unnecessary, incorrect, or badly formatted comments.	Partial, poorly written or poorly formatted comments.	Concise, meaningful, and well-formatted comments.	<input type="radio"/> <i>Unsatisfactory</i> <input type="radio"/> <i>Satisfactory</i> <input type="radio"/> <i>Good</i> <input type="radio"/> <i>Excellent</i> <input type="radio"/> <i>Not applicable</i> <input type="radio"/> <i>I don't know</i>
<b>Exceptions</b>	No exit status used in this code.	Exit status used, but they are scarce.	Exit status are used but plenty in every job.	Exit status employed moderately but not as a tool for every job.	<input type="radio"/> <i>Unsatisfactory</i> <input type="radio"/> <i>Satisfactory</i> <input type="radio"/> <i>Good</i> <input type="radio"/> <i>Excellent</i> <input type="radio"/> <i>Not applicable</i> <input type="radio"/> <i>I don't know</i>
<b>Reviewer's feedback</b>					
<b>What did the programmer do right?</b>					
.....					
.....					
.....					
<b>What did the programmer do wrong?</b>					
.....					
.....					
.....					
<b>How it be improved in the future?</b>					
<b>Correctness:</b> .....					
<b>Structure:</b> .....					
<b>Clarity:</b> .....					
<b>Layout:</b> .....					
<b>Exceptions:</b> .....					

Table 3.5. Rubric used in the pilot-experiment method

**Q: Write a java application for the following UML. (Note: `goUp()` returns “The bird is going up”). Create an `ArrayList` of `Fly` in the test class, create an object of `Bird` and add it to the list.**

```

classDiagram
    class Bird {
        +goUp():String
    }
    class Fly {
        +goUp():String
    }
    class Test {
        +main(String []args) void
    }
    Bird ..> Fly
  
```

**Student answer:**

```

package javaapplication55;

class Fly {

    Public String goUp(){

    }

}

|

class Bird extends Fly{

    public String goUp (){

        return " Bird is going up";

    }

}

public class test {

    public static void main(String[] args) {

        ArrayList a = new ArrayList ();

        a.add(new Bird());

    }

}
  
```

Computer Sciences department 2

**Figure 3.2. Random anonymous sample answers to assess in the pilot-experiment**

This method allows students to experiment with peer assessment using two marking scheme forms and then select the most appropriate form based on the results obtained. The following section outlines the data analysis tools used in this stage of the study.

### ***Data analysis tools***

The data from the experimental method were collated and analysed using SPSS. The following tests were applied to the data:

1. **Paired-sample t-test:** A t-test is useful to find out whether there is a significant difference between two groups. A paired samples t-test compares the means of two measurements taken from the same individual, object, or related units (Pallant, 2001). These “paired” measurements can represent things like: two different times, two different conditions, or two halves or sides of a subject or experimental unit. In this study, the same group of participants assessed a sample of peer answers using the marking scheme form then assessed another sample answer using the rubric form. Then, all peers’ responses were compared with the teacher’s responses for each criterion, either according to the rubric or the marking scheme, to determine the accuracy of the students’ judgments and the reviewers’ scores. After that, the paired-sample t-test was used to compare the mean of each form in order to establish which of the forms was more effective for students. Furthermore, the correlations between tutors’ and peers’ scores for each category were calculated, which meant the paired-sample t-test could be used to compare scores for each category, either according to categories in the rubric or the marking scheme. By using the paired-sample t-test in each category, the category with the highest correlation between the tutors’ and the peers’ scores in the selected group of samples was found.
2. **Multiple regression:** The regression analysis tool is a statistical technique that examines the relationship between two or more interest variables. Multiple regression analysis can be applied to explore the relationship between one continuous dependent variable and several independent variables or predictors (Pallant, 2001). Furthermore, it indicates how well a group of variables can predict a specific outcome (Pallant, 2001). There are several types of multiple regression analysis; the three major types are standard or simultaneous; hierarchical or sequential; and stepwise. In standard multiple regression, all independent variables (or predictor) are entered into equation simultaneously. This approach is used if the researcher has a set of variables and wants to predict the value of a variable based on the value of two or more other variables. This type can also explain how much unique variance in the dependent variable each of the independent variables defined (Pallant, 2001). In this study,

standard multiple regression was used in the pilot-experiment method to examine the relationship among personal variables and peer assessment scores.

3. **Open-ended Questions:** The answers in open-ended questions were coded based on marking scheme categories; there were five categories: correctness, structure, clarity, layout, and exceptions. Thus, students' answers were classified according to these categories and it was then determined how similar the answer was to the model answer prepared by the tutor. The level of similarity was divided into three levels (low, medium, and high). Low level means the comment of the student's assessor is different from the comment of the teacher in two aspects, the category and criterion; medium level means the comment of the student's assessor is similar to the teacher's comment in category or criterion; high level means the comment of the student's assessor is similar to the teacher's comment in category and criterion. Hence, students' responses to each question were compared with a model answer and their level of similarity was determined. Then, descriptive statistics (e.g., frequencies and percentages) were used to find which categories the students' reviewers had in common based on the answers' level of similarity with the model answer.

### **Pseudo-experiment method**

The pseudo-experiment is applied to describe studies that have at least one independent variable that is empirically manipulated and at least one dependent variable (Mildner, 2019). Pseudo-experiment methods allow for the strongest comparisons, with subjects (e.g. students, classrooms, schools, teachers) being randomly assigned to program and comparison groups (Gribbons and Herman, 1997). The aim of employing this method in this study is to find the effect peer assessment had (if any) on subsequent learning and achievement for those students who took part. Additionally, this study was able to assess the accuracy of peer assessment due to a larger sample size which was based on the best marking guide reflected in the peer assessment related to the teacher assessment. The pseudo-experiment method was based on the results of the previous pilot study, as the pilot study found the best marking guide form based on the higher correlation between the students' assessor assessment and the teacher's assessment.

Three main criteria should be met in a pseudo-experiment group: first, the pseudo-experiment method should have two groups, the experimental and the control group. Second, in a pseudo-experiment, the researcher can manipulate hypothesised variables to affect the outcome variable being studied. Third, participants must be randomly assigned to either the experimental or the control group. All of these conditions have been followed in this method as outlined below.

### ***Study setting***

In the pilot-experiment method, a paired-sample t-test was conducted to determine which marking scheme best correlates the students' assessment and the teacher assessment – rubric or the marking scheme, and participants got higher means using the marking scheme form than the rubric form. Many students expressed their preference for the marking scheme, as they were unfamiliar with the use of the rubric form in programming courses, and they took longer to read and prolonged the decision-making using the rubric. Thus, the marking scheme form was used as part of the pseudo-experiment method. The form contains the five categories as indicated above: correctness, structure, clarity, layout, and exceptions, and three open-ended questions. The empirical study contains two groups: the experimental group which applied the peer assessment activity as an external activity, and the control group which did not participate in the peer assessment activity. Both groups were created through random sampling, whereby the peer assessment activity was used in some labs which were selected randomly by the teacher, while other labs completed their work as usual. The teacher also selected random historical anonymous samples from a short quiz (Figure 3.3), which included a single question requiring students to write three classes with its properties showed as UML diagram. The experiment took place at PNU between two midterms exams that were set three weeks apart - a midterm exam to measure students' understanding of the learning materials and to identify weaknesses that require attention – and which were conducted by their teacher. The scores from these midterm exams were chosen because they provide a formal way to measure the peers' knowledge instead of applying a pre- or post-test.



**Q: Write a java application for the following UML. (Note: move() returns “the car is moving”). Create an ArrayList of Vehicle in the test class, create an object of Car and add it to the list.**

```

classDiagram
    class Car {
        +move():String
    }
    class Vehicle {
        +move():String
    }
    class Test {
        +main(String []args) :void
    }
    Car ..> Vehicle
  
```

**Student answer:**

```

package javaapplication55;
class Vehicle {

    Public String move(){
    }
}

class Car extends Vehicle {

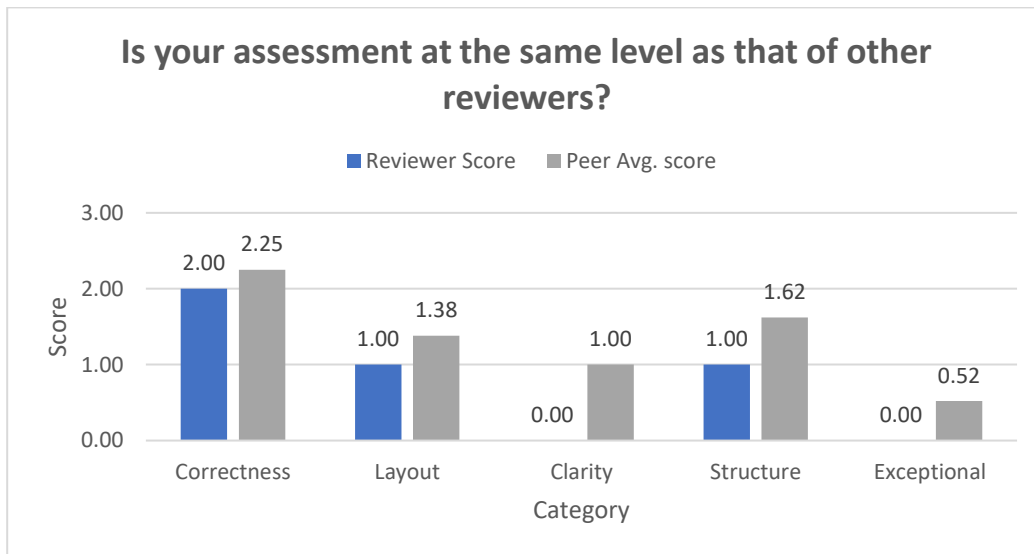
    public String move (){
    return " The car is moving";
    }
}

public class test {

    public static void main(String[] args) {
        ArrayList a = new ArrayList ();
        a.add(new Car());
    }
}
  
```

**Figure 3.3. Random anonymous sample answers to assess in the pseudo-experiment**

Next, based on the results of the peer assessment activity, feedback was prepared for all of the students' reviewers who participated in this activity in order to find a suitable form of feedback for students' reviewers. The feedback contains some results describing the level of the students' reviewers: visual results and text results. For example, Figure 3.4 indicates the visual feedback that compared the mean of a reviewer score in each category with all reviewers who assessed this activity, and Figure 3.5 is an example of written feedback that analysed the open-ended questions and compared the reviewer's feedback with the tutor feedback. For each result, there are three questions to make sure students understood the feedback results (Figure 3.6). In half of the samples, the forms of the results were swapped; text results were converted into visual results and vice versa. For example, the written feedback explained the reviewer's mean scores in each category and compared them with the mean of all reviewers who assessed this work. The visual feedback illustrates the reviewer's opinions and the tutor's opinion in a word cloud. Participants then expressed their satisfaction with the visual and written feedback, and the researcher calculated the frequencies of students' preferences.



**Figure 3.4. Visual feedback**

### What about text comments?

**The similarity between your view of program errors and the teacher's point of view.**

You find one error in the code which the programmer does not write the program correctly, but you do not specify the errors exactly as the teacher specifies.

Noting that the teacher has identified the following errors in the code:

*"Regarding the correctness category: errors occurred in Inherent, Composition, constructor, and get and set. In the Structure category: Variables were defined incorrectly, and in the Layout category, the programmer has ignored comments."*

Finally, in order to improve in the future, we recommend that you focus on the errors identified by the teacher, and you do not cover them in your text notes.

**Figure 3.5. Written feedback**

Which category did you assess better than your peers?	Which category did you assess worse than your peers?	How to improve for the future?
<ul style="list-style-type: none"> <li>• Correctness</li> <li>• Structure</li> <li>• Clarity</li> <li>• Layout</li> <li>• Exceptions</li> <li>• None of them</li> </ul>	<ul style="list-style-type: none"> <li>• Correctness</li> <li>• Structure</li> <li>• Clarity</li> <li>• Layout</li> <li>• Exceptions</li> <li>• None of them</li> </ul>	<ul style="list-style-type: none"> <li>• Focus on Correctness</li> <li>• Focus on Structure</li> <li>• Focus on Clarity</li> <li>• Focus on Layout</li> <li>• Focus on Exceptions</li> <li>• None of them</li> </ul>

**Figure 3.6. Questions about the visual feedback**

### ***Data analysis tools***

SPSS was used to analyse the data statistically. However, before applying statistical tools (e.g., t-tests, analysis of variance (ANOVA), and analysis of covariance (ANCOVA)), it required to outline a set of assumptions that statisticians make about data that are common to all the tools used in this research. The following general assumptions were made as described by Pallant (2001):

- a. **Normality:** It is assumed that the populations from which samples are taken is normally distributed.
- b. **Linearity:** The relationship between variables must have a straight line.
- c. **Homoscedasticity:** Samples are obtained from populations of equal variances.
- d. **Independence of observations:** The observations that make up the data should be independent of one another.

In SPSS, many statistical techniques can be used to explore relationships among variables and compare groups. Quantitative statistical techniques used in this experimental method are explained in this section; it is explained why and how techniques were used in this method.

1. **Independent sample t-test:** An independent sample t-test is an inferential statistical test used to compare the mean scores of two unrelated groups of people or conditions (Pallant, 2001). Based on demographic data that have been collected, participants were categorised into groups (e.g., familiar with peer assessment, unfamiliar). In the pseudo-experiment method in this study, an independent sample t-test was conducted to compare the total scores in the peer assessment for students who are familiar with peer assessment and students who are unfamiliar with peer assessment. It was also conducted to compare the total score for students who undertook a peer assessment twice and those who undertook it once.
2. **Paired sample t-test:** The paired sample t-test is a statistical method used to compare the mean scores for the same sets of people on two different occasions, or for

comparing matched pairs (Pallant, 2001). In a paired sample t-test, each entity or subject is measured twice, resulting in pairs of observations. For example, in the pseudo-experiment method, a paired sample t-test was conducted to evaluate the impact of the peer assessment activity on students' midterm scores. Moreover, a paired sample t-test was used between the first peer assessment experiment and the second peer assessment experiment to find out whether the repetition of the peer assessment improved the students' assessments.

3. **Analysis of Covariance (ANCOVA):** This is employed to examine the main effects and the interaction effects of categorical variables on a continuous dependent variable while controlling the impact of other selected continuous variables that co-vary with the dependent. A one-way between-group ANCOVA can be applied if the researcher has two groups pre-test design, for example, comparing the impact of two various interventions, taking before-and-after measures for each group (Pallant, 2001). In the pseudo-experiment method, a one-way between-group ANCOVA was conducted to compare the impact of the peer assessment activity on subsequent learning for those students who took part. First, the students were divided into two groups: students who participated in the peer assessment experiment (experimental group) and those who did not (control group). The independent variable was students' scores in the first midterm exam, and the dependent variable was students' scores in the second midterm exam. The students' first midterm scores were used to see if they had equivalent skills, and the second midterm scores were used to evaluate their programming skills after the peer assessment activity. The peer assessment activity took place between these midterm exams.
4. **Open-ended questions:** They were analysed in the same way as the open-ended questions in the pilot-experiment method.
5. **Students' preferences on peer feedback:** The frequency was used to find the form of feedback preferred by participants in the peer assessment activity: visual feedback or written feedback.

### **3.5 Validity and reliability of the methods used in the first phase**

Validity and reliability are concepts used to assess the quality of research. These concepts indicate how well a method measures something. The validity of a scale means the degree to which the researcher has measured methods (accuracy of a measure) (Pallant, 2001). The reliability of a scale is defined as how free it is from random errors and the ability of an instrument to generate reproducible results (the consistency of a measure) (Pallant, 2001). Choosing appropriate measurement scales can influence the collected data and guarantee high-quality results that measure exactly what the researcher wants to measure. Additionally, appropriate sampling methods must be selected. Clearly defining the population that one is researching produces valid, generalisable, and reliable results. Beyond this, random anonymous sampling can decrease the likelihood that members of a sample are different from its population (Chatti *et al.*, 2012). Besides, using mixed methods of data collection that include using different data collection methods, varying data sources, various analyses or theories help to ensure the validity and accuracy of the findings (Lesser *et al.*, 2016). All of these approaches have been followed to increase the reliability and validity of the study.

Complete validity in research is impossible as one of the characteristics of quantitative data is its degree of uncontrolled error; hence, validity can only be obtained to a certain degree (Cohen, Manion and Morrison, 2012). For the questionnaire, face validity was used (Taherdoost, 2018) by evaluating the items and their categories, the appearance of the questionnaire in terms of clarity, consistency and readability. Then, using Cohen's Kappa Index, the inter-rater agreement between two experts were calculated. It is difficult to verify the validity of qualitative data for the interview method; however, more detail regarding validity concerns are discussed in the section on the second phase because it concerns mainly qualitative data. For the experimental method, concurrent validity (Taherdoost, 2018) was used, which refers to the extent to which the results obtained in this test are compatible with those of a previously found and established measurement for the same construct.

Reliability is measured in three types of consistency: 1) stability: to ensure that the same outcomes are achieved when repeating the measurement; 2) internal consistency: to make sure all subparts of an instrument measure the same characteristic; and 3) equivalence: which refers to two observers conducting the exact same experiments and getting the same result. Reliability must be considered during the data collection process. The results must be precise, stable, and reproducible. To measure stability, the test-retest method can be used. The test-retest method, conducting the same test twice over a period with a group of participants was adopted in this study. To measure internal consistency, Cronbach's Alpha is often used; it is a statistic calculated from the pairwise correlations between items (Pallant, 2001). In addition, inter-rater reliability - the extent to which two raters agree - was employed in the qualitative data using consensus coding. An external Arabic researcher volunteered to analyse half of the interviews to confirm the codes and categories. The researcher and this volunteer researcher discussed, and agreed upon, the codes and themes. More detail regarding this method are discussed in the section on the second phase. Reliability and validity of the first phase of this study were considered, evidently.

### **3.6 Summary**

This chapter introduced the mixed methods approach that was adopted to address the research questions. The chapter described the study's research design and clarified that this study was divided into two phases. This chapter deals with the first phase of the research. It outlined the methods that were used to achieve the first two objectives of the study; identifying student and teacher perspectives of peer assessment, and evaluating the effectiveness and impact of peer assessment on participating students. This outline includes details of the data collection, the nature of the data used, and the analytical instruments used for each method. Methods that were selected included questionnaires, interviews, and experiments. The questionnaire method allowed to determine the receptivity of students and teachers to peer assessment and to determine their position on some controversial elements when implementing a peer assessment in programming courses. As for the interviews with teachers, results determined the appropriate marking

scheme form for the peer assessment. Experiments evaluated students' performances in the peer assessment and determined whether or not this activity was suitable for first-year students. Finally, the chapter described the steps needed to ensure that the results are valid and reliable. The combined results, regardless of their limitations, allow to achieve the first two research objectives of this study as described in the first chapter of this thesis. These results are presented and discussed in the next chapter.

## **Chapter 4. Results of the first phase**

### **4.1 Introduction**

This chapter presents the results of the first phase of this study. It outlines stakeholders' perceptions (teachers and students) regarding peer assessment and evaluates students' performances to validate the peer assessment activity. The following questions were investigated:

1. How do programming students and teachers perceive peer assessment in introductory programming courses?
2. Are first-year students who participate in peer assessment more likely to perform significantly better on programming skills than those who do not?

In this phase, quantitative data collection methods in the form of questionnaires and experimental methods were used primarily. In addition, semi-structured interviews with teachers were used to explore ambiguities that had emerged when analysing the teachers' questionnaires. This chapter presents statistical results derived from the quantitative data. Further, it presents key themes that were identified as part of the qualitative method.

### **4.2 Results from the questionnaires**

A questionnaire method was chosen to address the first research question: How do programming students and teachers perceive peer assessment in introductory programming courses? The questionnaire questions were organised in five blocks (see section 3.4.1, Table 3.1, and Table 3.2): (1) personal information; (2) benefits of peer assessment; (3) challenges of peer assessment; (4) suitable method for applying peer assessments in programming courses, with five sub-sections: marking criteria, privacy, online assessment, teamwork assessment, and grading; and (5) open-ended questions.



#### 4.2.1 Demographic data: Students and teachers' questionnaires

##### Students' questionnaire

The students' questionnaire was distributed between 5 July and 4 November 2018. A total of 244 participants completed the questionnaire; (n=226, 93%) undergraduate students were recruited from the College of Computer and Information Sciences of several Saudi universities, and (n=18, 7%) undergraduate students came from the School of Computing at Newcastle University in the UK. Most of the participants (n=231, 95%) were female with only (n=12, 5%) being male. This is because most of the responses were from PNU, a female-only university. There are responses from other universities, but it was difficult to distribute the questionnaire in male colleges due to social constraints in Saudi Arabia which stipulate that men and woman are taught separately. The largest group of respondents were students of Computer Science (n=102, 42%), followed by Information Technology (n=61, 25%), Information Systems (n=46, 19%), Software Engineering (n=32, 13%) and Game Engineering (n=3, 1%). They were in their first (n=110, 45%), second (n=58, 24%), third (n=50, 20%), and fourth year (n=23, 10%), respectively, so they had either finished studying the introductory programming course (CS1) or were currently studying CS1. Most of the students (n=236, 97%) had no prior experience of peer assessments in programming courses. Table 4.1 shows the distribution of students' demographic variables.

Variable	Item	Frequency	Percentage
<b>University</b>	Saudi Universities	226	93%
	Newcastle	18	7%
<b>Gender</b>	Male	12	5%
	Female	231	95%
<b>Subject</b>	CS	102	42%
	IS	46	19%
	IT	61	25%
	SE	32	13%
	GE	3	1%

<b>Year of Study</b>	First	110	45%
	Second	58	24%
	Third	50	20%
	Fourth	23	10%
	Missing	3	1%
<b>Peer Assessment experience</b>	No	236	97%
	Yes	8	3%

**Table 4.1. Distribution of students' demographic variables**

In summary, more than 200 participants filled in the students' questionnaire but there is a significant difference between groups within the variables (university, gender, and peer assessment experience). This indicates the difficulty of comparing groups in variables. Many of the participating students were in their first year, and peer assessment was a new pedagogical experience for most participants.

### **Teachers' questionnaire**

The teachers' questionnaire was distributed between the 5 July and 4 November 2018. A total of 48 participants completed the teacher questionnaire; (n=43, 90%) programming teachers were from the College of Computer and Information Sciences of several Saudi universities, and (n=5, 10%) programming teachers were from the School of Computing at Newcastle University, UK. Around half of the respondents (n=20, 42%) were teachers who had more than five years' experience. However, most participants had not used peer assessments (n=41, 85%) before; only (n=7, 15%) had experience with peer assessments. Since only three participants from Saudi universities had used peer assessment in programming courses, additional variables to collect accurate peer assessment experiences, such as how long peer assessment had been used, were added. Ultimately the data were only collected from Newcastle University participants (n=4, 57%). Two teachers from Newcastle University had used peer assessment for more than one year, and two had used it for less than one month. Two teachers who had used it for a long time described it as a positive experience "*It has worked well in my course and has helped students learn*"; one teacher found it difficult to use "*It was difficult to*

*administer but worked fairly well*"; and one teacher had a negative experience: *"It was poorly received by students and did not add value to their learning"*. Table 4.2 shows the distribution of teachers' demographic variables.

variable	Item	Frequency	Percentage
<b>Country</b>	Saudi universities	43	90%
	Newcastle University	5	10%
<b>Experience</b>	0-3	11	23%
	3-6	17	35%
	>5	20	42%
<b>Using peer assessment</b>	Yes	7	15%
	No	41	85%
<b>Period of using peer assessment</b>	Never used	41	85%
	Less than month	2	4%
	A year	0	-
	More than one year	2	4%
	Missing data	3	7%

**Table 4.2. Distribution of teachers' demographic variables**

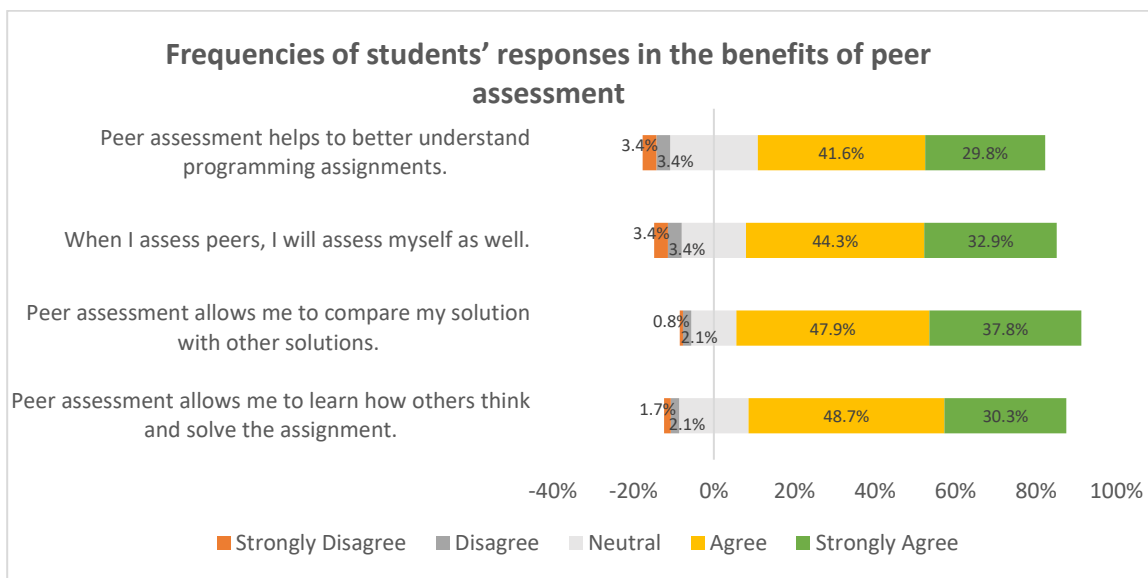
In summary, the number of participants for the questionnaire should be more than 60 participants following (Baxter, Courage and Caine, 2015); however, fewer than 60 teachers filled in the questionnaire, and there was a significant difference between groups within the variables (e.g., country and use of peer assessment). Besides, peer assessment activities were not common in Saudi universities; they were more common at Newcastle University. Still, most of the responses were from Saudi universities.

#### **4.2.2 Descriptive data: Benefits and challenges of peer assessment**

##### **Students' perceptions**

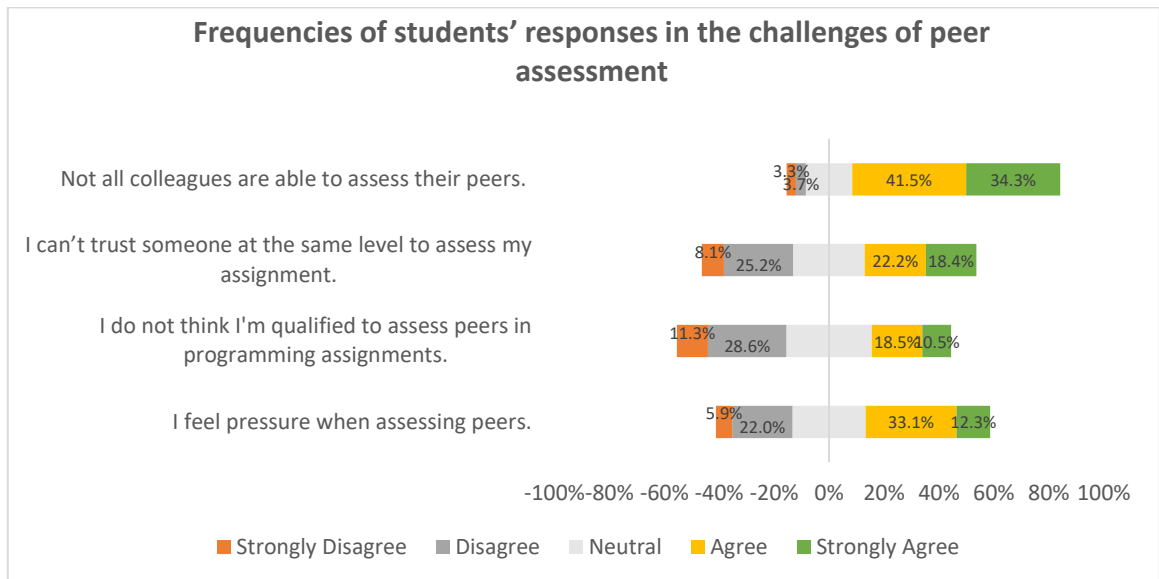
The frequencies of the benefits of peer assessments in programming courses (broadening perspectives, comparing solutions, self-assessment, and better understanding) have been calculated and visualised, as shown in Figure 4.1. The median in all items was "Agree". This indicates that more than 75% of the students were aware of the benefits of peer assessment in programming assignments - as they selected "Agree" or "Strongly

agree” - and believed in the importance of peer assessment in programming courses. In contrast, less than 7% of the students selected “Disagree” or “Strongly Disagree” for this section. When analysing each item separately, the greatest benefit agreed by students was “Peer assessment allows me to compare my solution with other solutions” (37.8% Strongly agree, 47.9% Agree). The results demonstrate that the majority of programming students in this group of samples believed in the benefits of peer assessment in programming assignments.



**Figure 4.1. Frequencies in benefits of peer assessment-students' responses**

Using peer assessment in programming courses might present a number of challenges as illustrated in Figure 4.2 (e.g., pressure, qualifying to assess, trust, and ability). Students' opinions in this regard were varied so there was no clear agreement about the challenges related to peer assessment. The median of all items was “Neutral”, except for the following item: “Not all colleagues are able to assess their peers”, as its median was “Agree”. This indicates that students' opinions differ quite widely between individuals. It also shows that they did not have a significant problem with peer assessment except in one item, as many students believed that not all reviewers can assess their peers (34.3% Strongly agree, 41.5% Agree). Therefore, more insights into students' perspectives regarding reviewers' abilities must be collected.



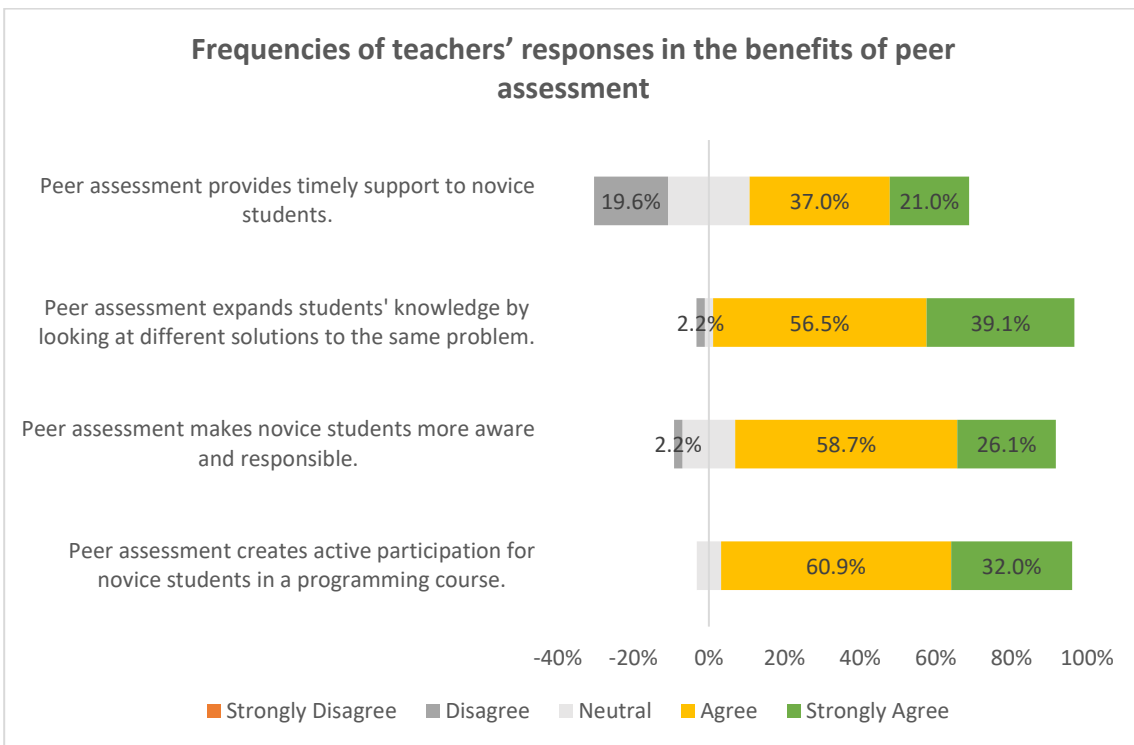
**Figure 4.2. Frequencies in challenges of peer assessment-students' responses**

These results indicate that students were willing to engage in a peer assessment practice with programming assignments as they believed in the benefits of peer assessment and did not have significant issues with the activity. However, the different abilities between students could hinder the assessment process; thus, more information must be collected about the abilities of reviewers when applying peer assessment.

### Teachers' perceptions

The frequencies of the benefits of peer assessments in programming courses (e.g., active participation, responsibility, expanding knowledge, and timely support) have been calculated and visualised, as shown in Figure 4.3. The median in all items was "Agree", then "Strongly Agree". This indicates that more than 78% of teachers "Strongly Agree" or "Agree" with the items that stated the benefits of peer assessment. When analysing each item separately, the greatest benefit agreed by teachers was "Peer assessment expands students' knowledge by looking to other solutions" (39.1% Strongly agree, 56.5% Agree). In contrast, no participants chose the "Strongly Disagree" option for the benefits of peer assessment; this indicates teachers understand the importance of peer assessment in programming courses. Thus, they value the learning aspect of peer assessment rather

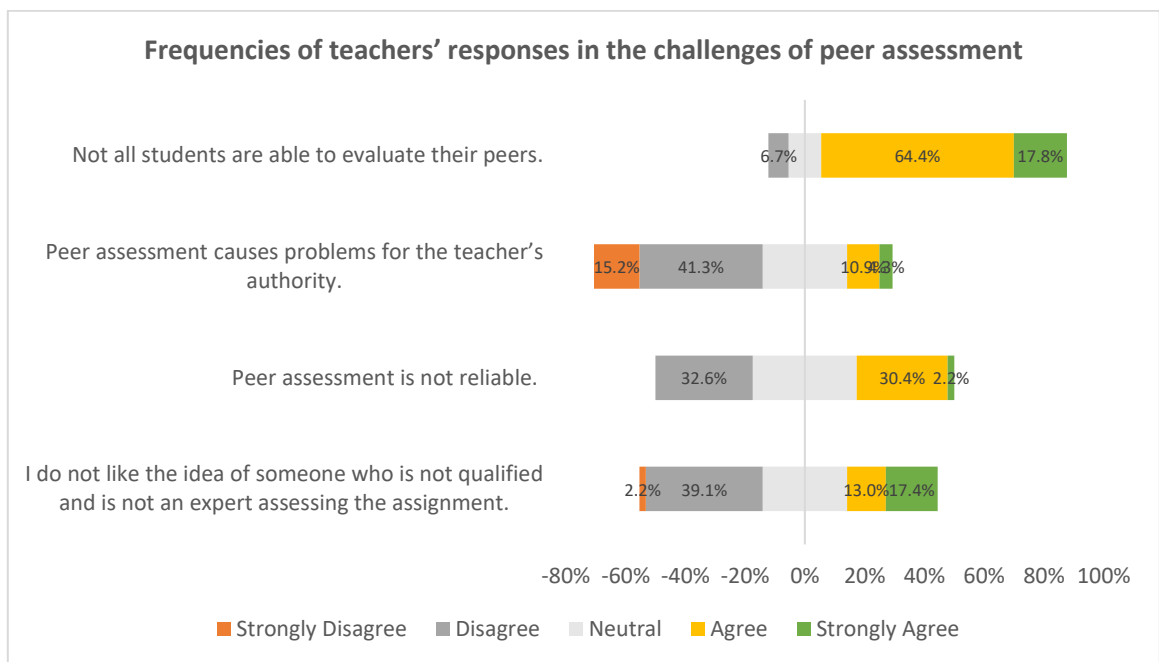
than the assessment aspect. However, there is a disagreement (19.6% Disagree) in the following item, “Peer assessment provides timely support to first-year students”. The researcher did not mean that it should be considered a replacement for the developmental feedback provided by teachers, but peer assessment could satisfy timeliness as it is difficult for teachers to provide timely assessment feedback to students, especially in a class with a large number of students.



**Figure 4.3. Frequencies in benefits of peer assessment-teachers' responses**

Figure 4.4 shows the teachers' perspectives on some common issues such as qualification, reliability, authority, and reviewers' skills. There was a distribution of teachers' opinions and no clear central tendency in the items. The median of the following sentences: “I do not like the idea of someone who is not qualified and is not an expert assessing the assignment”, and “Peer assessment is not reliable” were “Neutral”, while the median of the following item: “Peer assessment causes problems for the teacher's authority” was “Disagree”, and the median of the following item: “Not all students are able

to evaluate their peers” was “Agree”. When analysing each item separately, the largest group of teachers (17.8% Strongly agree, 64.4% Agree) believed there were different levels of ability between students in assessing peers, which is what the students also thought. Therefore, a qualitative method was required to investigate this item in more detail. The greatest disagreement of teachers (15.2% Strongly disagree, 41.3% Disagree) relates to the item on teachers’ authority, which suggests that peer assessment does not affect the role of teachers.



**Figure 4.4. Frequencies in challenges of peer assessment-teachers’ responses**

Teachers who responded to the questionnaire understood the benefits of peer assessment, although few had experience in using them as the demographic data showed. And there was no indication of significant issues with peer assessments that might hinder using the activity with programming students. Considering “benefits” and “challenges” enabled the researcher to take a decision of using peer assessment in programming courses objectively, and it helped to reach a balanced and informed decision. However, since peer assessment was not popular with this group, these results

indicate the need to discuss reasons that prevent teachers to use peer assessments within programming curricula.

#### **4.2.3 Correlation between awareness of benefits and fear of challenges**

The correlation was used to describe the relationship (if any) between awareness of benefits and fear of challenges when using peer assessment in introductory programming courses. The relationship was investigated using the Pearson product-moment correlation coefficient. Preliminary analyses were performed to ensure no violation of normality, linearity, and homoscedasticity assumptions. There was a small correlation in students' data between two variables,  $r=.237$ ,  $n=238$ ,  $p<0.05$ , with high awareness of peer assessment benefits and high-level fear of peer assessment challenges. However, there is no correlation for teachers' results because the *p-value* was larger than 0.05. Hence, there was no significant difference in the teachers' results.

The next step entailed calculating the regression analysis; variables that qualified to be included in the regression analysis had at least a moderate relationship (e.g., greater/less than 0.4/-0.4). The correlation was  $r=.237$ , which indicates a very weak relationship. Thus, there was no need to calculate the linear regression for the given samples.

#### **4.2.4 Comparison between students and teachers in the benefits and challenges**

Welch's t-test was conducted to compare the mean of awareness of the benefits of peer assessments between students and teachers. Although it has been argued that taking the mean of a Likert-scale variable might not provide useful answers (Barry, 2017), the researcher found that it was meaningful to calculate the means of all the benefits items, and the means of all the challenges items in order to compare between the belief of the two groups, students and teachers. According to Sullivan and Artino (2013), parametric statistics with the Likert scale can be used to raise the quality of research. Thus, Welch's t-test was used with two unequal groups: students ( $n=238$ ) and teachers ( $n=46$ ). The result indicated no statistically significant difference between the two groups in the awareness of the benefits:  $F(1, 284)=.047$ ,  $p=.785$ . The students' group's mean score ( $M=4.9$ ,  $Sd=.71$ ) was not significantly different from the teachers' group ( $M=4.07$ ,  $Sd=.50$ ).



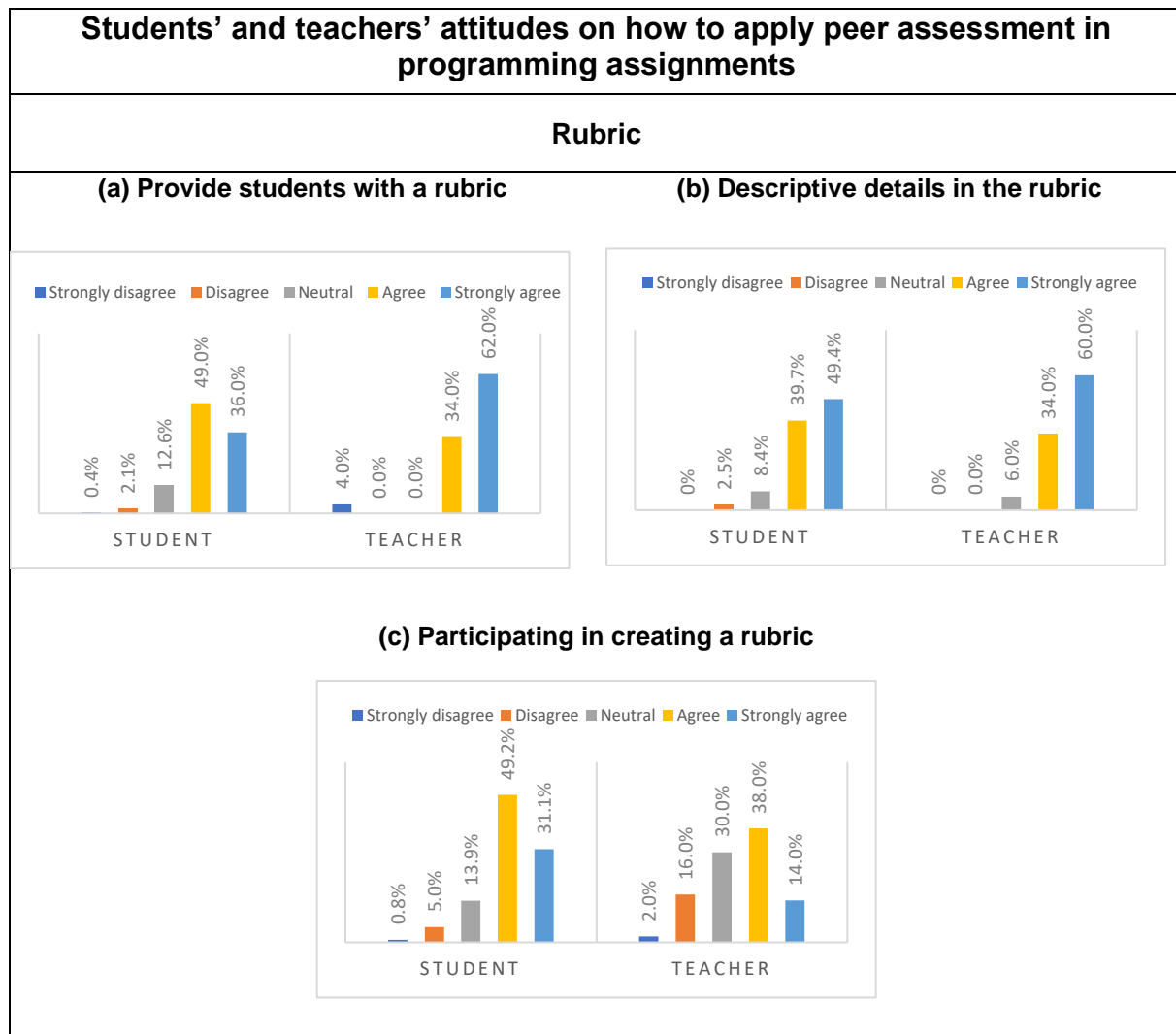
In summary, there was no significant difference between groups regarding the awareness of benefits.

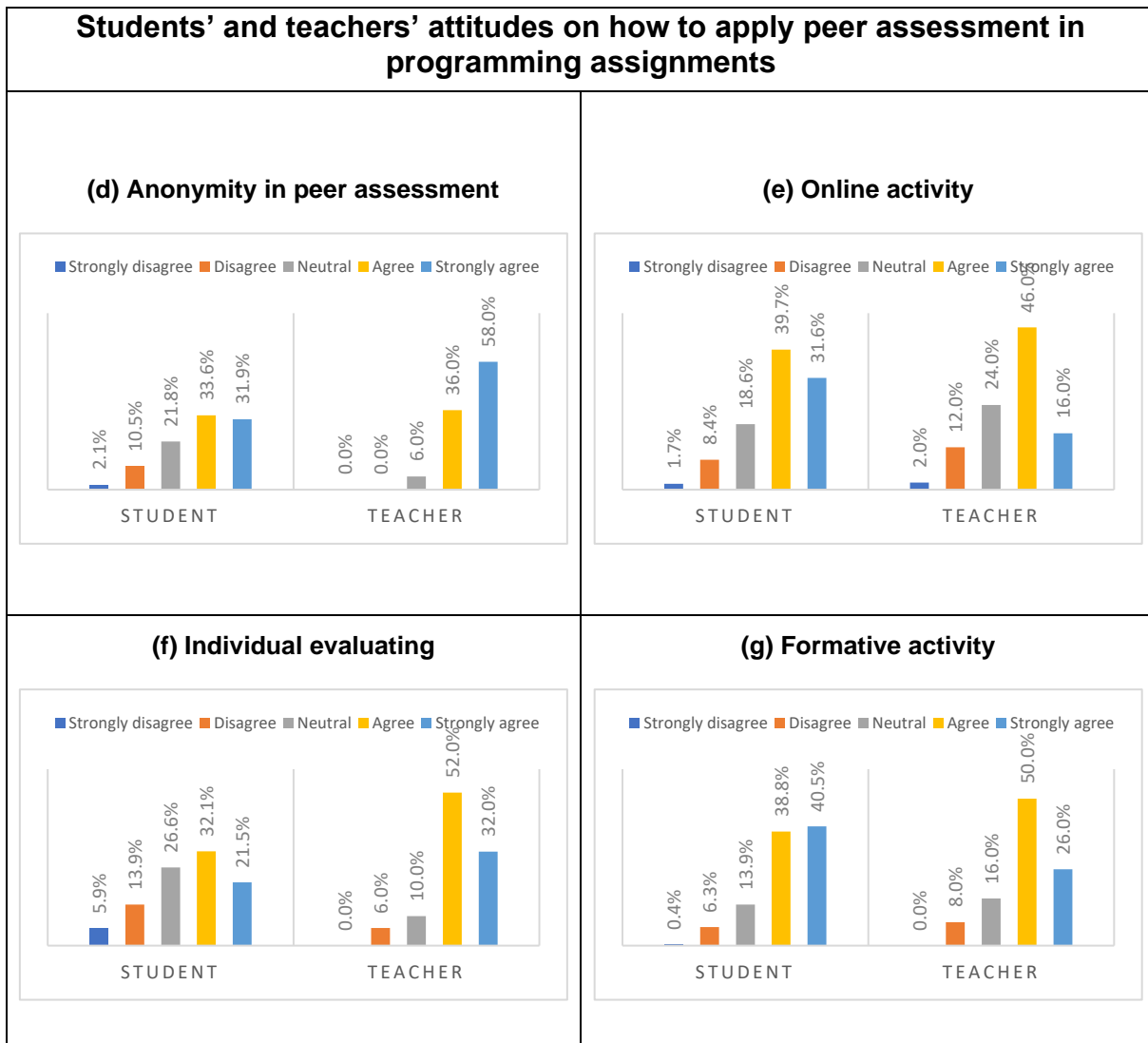
However, Welch's t-test was conducted to compare the mean of fear of challenges between students and teachers. The result indicated a statistically significant difference at  $p < .05$  level for two groups in regard to fear of challenges:  $F(1, 284)=8.9, p<.001$ . Despite reaching statistical significance, the actual difference in the mean scores between groups was a small effect. The effect size, calculated using eta squared, was 0.03. The students' group's mean score ( $M=3.32, Sd=.838$ ) was significantly different from that of the teachers ( $M=2.93, Sd=.585$ ). As a result, the fear of challenges from the students' perspectives is higher than the fear of challenges from the teachers' perspectives. This may be because such activities concern students more than teachers; therefore, students fear obstacles more.

#### **4.2.5 Descriptive data: How to apply peer assessment**

One of the aims of the data gathered through the questionnaire was to devise a suitable method for peer assessment in programming courses, which was informed by students' and teachers' opinions. Therefore, the third block in the questionnaire (a suitable method for applying peer assessments) was divided into the following five sections: marking criteria, privacy, online assessment, teamwork, and grading. Figure 4.5 shows a number of graphs that indicate the percentages of participants' responses to each item related to the use of peer assessment. The median for all student items (providing a rubric, providing descriptive rubric, participating in creating a rubric, online activity, individual evaluation, and no grading) was "Agree". While the median of teachers in three elements - providing rubric (Figure 4.5, a), descriptive details rubric (Figure 4.5, b), and anonymity (Figure 4.5, d) - was "Strongly agree". As for the rest of the elements (participating in creating a rubric, online activity, individual evaluation, and no grading) (Figure 4.5, c, e, f, g), the median was "Agree". Thus, all the elements have a general agreement orientation for both students and teachers. When analysing each item separately, the greatest item agreed by students was offering a descriptive details rubric (Figure 4.5, b) as 49.4% Strongly agree, and 39.7% Agree. While the greatest item agreed by teachers was providing first-

year students with rubrics (Figure 4.5, a) as 62% Strongly agree, 34% Agree. These results indicate that teachers must provide students with a rubric when assessing peers as the students' and teachers' frequencies indicate. "Disagree" for all elements, on the other hand, did not exceed 16%, as some teachers did not agree with students' participation in creating a rubric (Figure 4.5, c). In contrast, some students (13.9%) did not agree on the individual evaluation (Figure 4.5, f).





**Figure 4.5. Students' and teachers' opinions about elements of peer assessment**

Based on the information gathered in the form of students' and teachers' responses, a general method for applying peer assessment was defined. Firstly, the rubric is a critical element, and it should contain descriptive details to accurately assess peers. Students can participate in creating the rubric. However, more information from teachers is needed regarding students' participation in creating the rubric. Further, a lot of students and teachers support anonymity in the peer assessment activity. Also, many participants agreed with the idea of online peer assessment and using peer assessment as a feedback activity rather than a summative assessment. Evaluating peers' work individually in

programming assignments was strongly supported by teachers, but students had distributed opinions. The following section outlines a comparison between students' and teachers' perspectives in applying peer assessment using the Mann-Whitney U test.

#### **4.2.6 Comparison between students and teachers in methods of applying peer assessment**

A Mann-Whitney U test revealed a significant difference in all aspects of applying peer assessment, except in two items; there were no significant differences, which were in “The scale of rubric must contain descriptive details to each criterion” and “Peer assessment should be used as a feedback activity rather than a graded practice”. The distribution of two groups (students and teachers) were the same in these two dependent variables. The Mann-Whitney U test for item No.2 indicated no significant difference between students and teachers  $U=5252.0, Z=-1.5, p=.134$ . Also, the Mann-Whitney U test in item No.7 was  $U=5126.0, Z=-1.6, p=.109$ . Table 4.3 results of Mann-Whitney U test compare students' and teachers' perspectives in some of the critical elements of peer assessment.

When analysing other significant items, Eta squared was calculated for all of the elements; they had a small effect size. When analysing the frequencies, students and teachers differed only in the intensity of the agreement; but all of them agreed with the statement. This indicates that they have the same tendency in applying the peer assessment method.

	No	Item	Md for student	N for student	Md for teachers	N for teacher	U	z	p
<b>Marking Scheme</b>	1	Provide first-year students with a rubric	4	238	5	48	4266.5	-3.5	.000
	2	The scale of rubric must contain descriptive details to each criterion	4	239	5	45	5252.0	-1.5	.134
	3	Students should participate in creating a rubric	4	236	4	44	3934.0	-4.0	.000

	No	Item	Md for student	N for student	Md for teachers	N for teacher	U	z	p
Anonymity	4	The assessee and assessor should be anonymous.	4	233	5	46	3736.0	-4.3	.000
Online	5	Online peer assessment is an effective way to assess programming assignments.	4	233	4	45	4879.0	-2.1	.039
Teamwork	6	Evaluating peers' work individually is better than a collective discussion between assessors.	4	223	4	45	4134.0	-3.5	.000
Grading	7	Peer assessment should be used as a feedback activity rather than a graded practice.	4	236	4	46	5126.0	-1.6	.109

**Table 4.3. A Mann-Whitney U test results**

The data analysed so far allows us to build an initial peer assessment activity with students. However, some aspects remain to be clarified by programming teachers (e.g., details on rubric form) to develop and conduct an experiment of a peer assessment activity with first-year students.

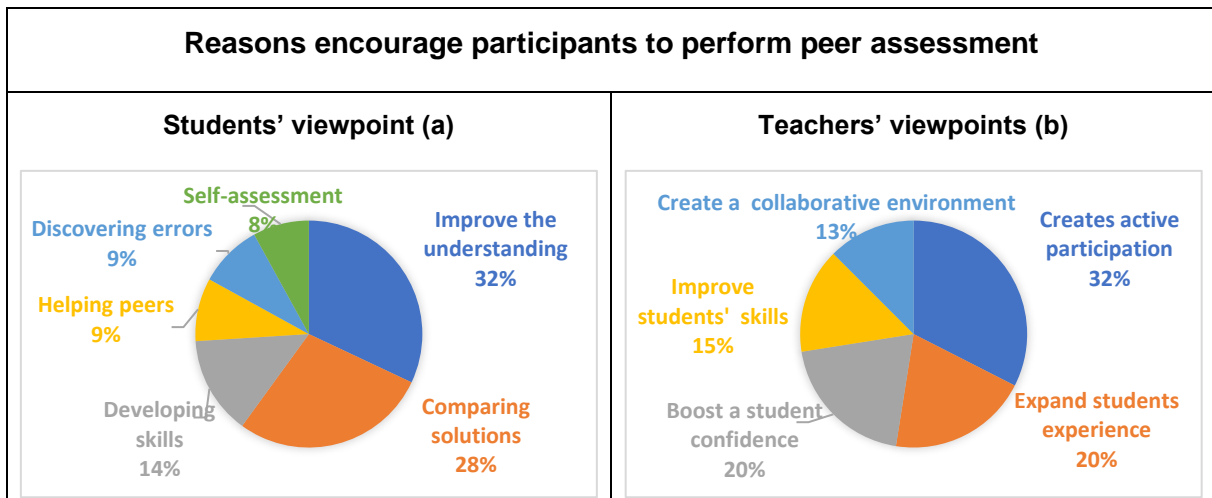
#### **4.2.7 Open-ended questions**

##### **What would encourage you to perform peer assessment with programming assignments?**

This question explored participants' motivations to perform a peer assessment for the programming assignment. Out of the 244 responses to the questionnaire, 138 answered

this question regarding students' responses. Qualitative data were analysed based on the affinity diagram, and many groups and subgroups were generated. Figure 4.6, (a) shows the main groups and their percentages extracted from the data. It is clear that 32% of students emphasised that peer assessment activity could enhance their understanding of the assignment and broaden their knowledge in programming. In addition, 28% of students want to do a peer assessment to see how a problem has been solved differently and to look at other approaches. Moreover, 14% aspire to improve their skills through this type of assessment – their programming skills, critical and judgement skills, and soft skills. Surprisingly, 9% of students think that helping their peers would be the primary motivation for peer assessment, with the same number believing that students would be excited about finding code errors. Finally, 8% of students think that assessing peers would lead to them self-assessing. These answers show that most of the students focused on the motivations of being a reviewer, rather than being an author.

The teachers asked the same question; from the 48 responses to the questionnaire, 40 answered this particular question. Figure 4.6, (b) shows that 32% of teachers thought that peer assessment could create active participation in the learning process; 20% expected that peer assessment would expand students' experience by seeing and evaluating others' code; and 20% of teachers believed that peer assessment could boost students' confidence by helping each other. Furthermore, 15% thought peer assessment could improve students' programming and critical and soft skills. Finally, 13% expected that it would create a collaborative and comfortable environment among the students if they applied such an activity.



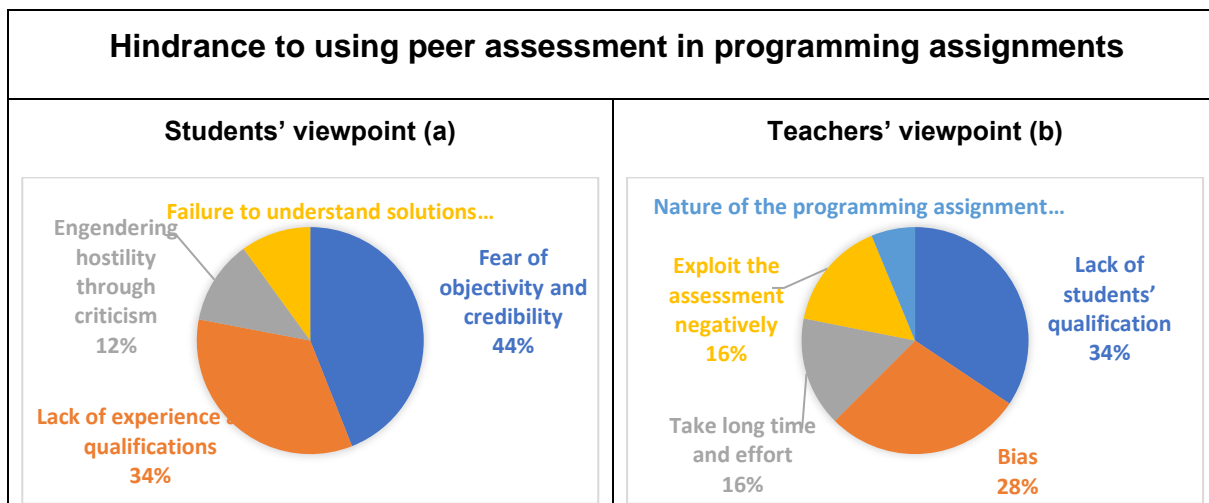
**Figure 4.6. Reasons encourage participants to perform peer assessment**

Students in this question were excited to assess their peers for many reasons; the most frequent reason was to improve understanding of programming assignments. Also, the most frequent reason from the teachers' viewpoints was to create an active learning strategy. This indicates that the selected sample believe in the importance of active learning for first-year students.

### **What elements would prevent participants from using peer assessment in programming courses?**

Out of 244 student responses to the questionnaire, 147 answered this question. From Figure 4.7, (a) it is apparent that the students' first concern was a lack of objectivity and credibility in the peer assessment (44%). The subjectivity in peer assessment is potentially either due to an intentional manner (e.g., relationship) or an unintentional (e.g., too harsh or lenient) bias among students. The second concern is lack of experience and qualification (34%); students feel they would not be qualified to judge their peers' work in programming assignments because they are novices. Furthermore, 12% of the students fear that their criticism might create hostility towards their peers. Some students (10%) were afraid of failure in understanding peers' solutions.

Regarding teachers' perspectives, from the 48 responses to the questionnaire, 33 answered this question. As Figure 4.7, (b) shows, the most significant response by teachers is the lack of students' qualifications as 34% did not like the idea of "non-experts" assessing their peers' code; they thought first-year students were not qualified to assess their peers yet. Also, 28% of teachers were concerned about bias (e.g., friendship, peers not accepting each other's views, being unfair), while 16% thought that students might exploit the assessment negatively (e.g., by ridiculing colleagues). Moreover, 16% of teachers believed that the process takes a long time and effort to prepare and manage. The smallest percentage (6%) thought that because of the nature of programming assignments, the questions often do not have a single correct answer, so it is difficult for first-year students who do not yet have much knowledge to assess the solutions.



**Figure 4.7. Hindrance to using peer assessment from participants' perspectives**

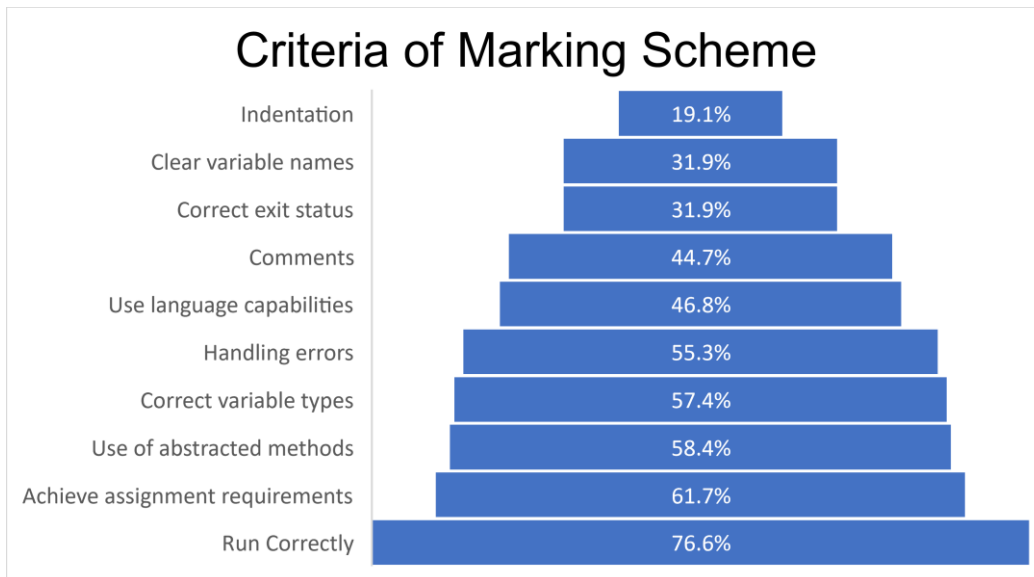
Open-ended questions allowed collecting qualitative answers from students and teachers that were, for the most part, full of information about their motivations to use peer assessment in programming courses, and barriers that could hinder this activity. Hearing about the benefits and challenges from stakeholders themselves was very important in order to make sure that the activity is used properly, to enhance motivation, and to find solutions to problems that were identified. By asking these questions, participants had the opportunity to answer whatever they liked, without limiting or influencing them with



predefined answers. Thus, some new information came to light in these open-ended questions that led to consider new ideas in peer assessment (e.g., abilities of reviewers).

### **What are the criteria that must be considered in the marking scheme?**

Teachers' opinions on the marking scheme criteria were considered. To this end, possible elements that could be included in a marking scheme were shown to teachers to ascertain which they considered the most important. These were multiple selection questions so that teachers could select more than one element. The elements were: run correctly, handling errors, assignment requirements, well structured, variable names, use of abstracted methods, easy to follow, comments, indentation, and the correct exit status. These elements were decided after reviewing literature that considers marking schemes. Figure 4.8 displays the essential elements based on the teachers' choices. It appears that the code running correctly is the most important element (76.6%). Next, they considered achieving the assessment's requirements necessary (61.7%); the students should fit a particular code to the assignment's needs. The next element in the rankings was using abstracted methods in the code (58.4%), followed by handling errors (57.4%). In contrast, indentation seemed unimportant to the teachers (19%). The marking scheme criteria were decided from these responses, but further details about building the marking scheme, its scales, and the level of details must be collected using the interview method to create a suitable marking guide for students' reviewers in peer assessment.



**Figure 4.8. Teachers' perceptions of essential criteria in the marking scheme**

In summary, there are no criteria that must be excluded when designing the rubric for a peer assessment. However, some criteria are more important than others, and criteria can be categorised for better comprehension and retention.

#### **4.2.8 Reliability and validity of scale**

Reliability is typically measured in quantitative research through the use of statistical techniques. Using SPSS, Cronbach's Alpha has been calculated to measure the scale's internal consistency. The reliability of the students' questionnaire was 0.824, meaning that the student questionnaire was statistically acceptable and reliable, and the scale had very good internal consistency. Further, the reliability of the teachers' questionnaire was 0.704, meaning that it had good reliability, and the scale had good internal consistency. Moreover, the test-retest method was conducted over a period with a different group of participants. The questionnaire was distributed at PNU; then, after a few months, it was distributed at Newcastle University. Regarding validity, two external experts from the Computer Science and Information System College at PNU recorded their perspectives, either agreement or disagreement on the items of the questionnaire. Then, Cohen's Kappa index was calculated to measure the agreement between two experts regarding the questionnaire's items,  $kappa (\kappa) = .571$ ,  $p < .05$  represents a moderate strength of

agreement. Therefore, the current method was deemed to be satisfactorily reliable and valid.

In summary, the questionnaire method helped answer the first research question: How do programming students and teachers perceive peer assessment in introductory programming courses? Although peer assessment was a new form of practice and a new pedagogical experience for the majority of participants in this study who were used to studying in summative assessment settings, a large proportion of participants were convinced of the benefits of peer assessment when learning programming. Furthermore, the obstacles facing students and teachers in peer assessment are general obstacles (e.g., injustice, different levels of abilities, inexperience in a subject). This indicates that the obstacles to peer assessment in programming for this sample were similar to those faced in other disciplines, and are not insurmountable. Moreover, the main points of using peer assessment with programming students were made clear. For example, the participants needed a detailed marking scheme, and both teachers and students felt that the assessment should be anonymous, an individual activity, and a formative assessment. These findings suggest that a marking scheme must be carefully prepared to support students when assessing their peers. It also became clear that there was a need to talk to teachers to better understand their views before developing a peer assessment exercise. The following section discusses the results of the interviews conducted with programming teachers toward peer assessment in introductory programming courses.

### **4.3 Results from the interviews**

Semi-structured interviews were also used to address the first research question: How do programming students and teachers perceive peer assessment in introductory programming courses? This method was chosen for two reasons: first, the demographic results of the teachers' questionnaire showed teachers' reluctance to use peer assessments in introductory programming courses. Second, since the second research objective focuses on measuring first-year students' performance accuracy in a peer assessment, the experiment needs a marking scheme guide. The marking scheme should be created based on teachers' perspectives. The data collected from the questionnaire

regarding the marking scheme was insufficient. Thus, interviews have been conducted to ascertain teachers' perspectives, which the questionnaires cannot measure. The interviews questions were organised around four blocks (see 3.4.2, Table 3.3): (1) personal questions; (2) benefits and challenges of peer assessments in programming courses; (3) suitable method to apply peer assessments in programming courses; and (4) questions to take advantage of data collected from peer assessments.

#### **4.3.1 Demographic data**

The interviews were conducted on dates between 30 July and 20 October 2018. A total of 11 participants took part in the interviews; (n=5, 45%) came from different Saudi universities (PNU, King Saud University, and Imam Mohammad Ibn Saud Islamic University), and (n=6, 55%) participants from Newcastle University, UK. All Saudi participants were female, and only one female participant was from Newcastle University, so the total of female participants was (n=6, 55%). The total of male participants was (n=5, 45%), all of whom were from Newcastle University. Participants occupied different positions: professor (n=1, 9%), senior lecturer (n=2, 19%), lecturer (n=4, 36%), and assistant teacher (n=4, 36%). A total of (n=5, 45%) participants from Newcastle University had used peer assessments in different courses, and only one of them had used a peer assessment in a programming course. In contrast, none of the Saudi participants had used a peer assessment before. Table 4.4 shows the participants' university, gender, experience, position, and peer assessment experience.

<b>Variable</b>	<b>Item</b>	<b>Frequency</b>	<b>Percentage</b>
<b>University</b>	Saudi universities	5	45%
	Newcastle	6	55%
<b>Gender</b>	Male	5	45%
	Female	6	55%
<b>Experience</b>	0–3 yrs	2	18%
	3–6 yrs	2	18%
	>5 yrs	7	64%

<b>Position</b>	Professor	1	9%
	Senior Lecturer	2	19%
	Lecturer	4	36%
	Assistant teacher	4	36%
<b>Peer assessment experience</b>	Yes	5	45%
	No	6	55%

**Table 4.4. Interviewees' demographic variables**

The data show that peer assessment is used at Newcastle University in some computer science subjects, but it is rarely used in introductory programming courses. Saudi universities are not accustomed to using peer assessments in programming courses at all at this point.

#### **4.3.2 Presentation of the key themes**

Thematic analysis was used to identify patterns in the interviews' data regarding teachers' viewpoints. Four key themes related to using peer assessment in programming assignments emerged: 'Difficulties of novice programmers', 'Teachers' attitudes towards peer assessment', 'Strategies preferred to carry out the peer assessment' and 'Utilising the data collected from peer assessment'. For clear data display, tables were created to summarise the coding information in detail (e.g., Table 4.5). The first and second columns list the content of each theme and the sub-categories' themes. The third column presents an example of quotes from the interviews for each sub-theme. The final column displays the frequency of this sub-theme.

#### **Theme 1: Difficulties of first-year programmers**

This theme demonstrates some of the students' difficulties - from the perspective of the teachers - during the first year of studying introductory programming courses. It includes the following sub-themes, summarised in Table 4.5:

- Difficulties at lecture time
- Difficulties at practice time

Theme	Subcategory theme	Supporting quote	Frequency
<b>At lecture time</b>	Understanding concepts	<i>"It is a new subject; and difficult to understand its fundamentals and to be familiar with the concepts."</i>	4
	Time consuming	<i>"Students do not grasp the concept quickly, and the lecturer will continue teaching without waiting for all the students to fully understand the topic."</i>	3
	Pay attention in lectures	<i>"The main problem of students keeps full attention in lectures."</i>	2
<b>At practice time</b>	Lack of practice	<i>"Some students struggle in labs even if they are getting the concepts because they should put effort into extra practice."</i>	4
	Lack of confidence	<i>"Some first-year students do not trust their capability of performing programming assignments."</i>	3
	Lack of diversity in practice types	<i>"Our questions are often writing programs. We do not do enough of looking at other programs and analysing other programs."</i>	2

**Table 4.5. Theme 1: Some difficulties of first-year students**

First-year computing students face a diversity of challenges. Not only must they cope with the pressures of starting higher education, which means adapting to university study, but they are also challenged with immersing themselves into a discipline in which they may not have had any previous knowledge and for which they must basically learn a new language. Teachers mentioned many issues; the theme divided these into two parts: lectures and lab sessions. During lectures, students often struggle to understand the basic concepts of programming and may have difficulties completing a particular assignment. One participant said: *"It is a new subject; and difficult to understand its fundamentals and be familiar with the concepts"*. Furthermore, teaching can be time-consuming, particularly if the subject is difficult, and students' abilities to understand are different. Moreover, some teachers struggle to maintain everyone's full attention throughout the lecture. Therefore,

taking an active role in class could help to keep students' minds busy so they do not have time to get distracted.

In lab sessions, some students struggle during practice time even if they understand the concepts. This is because programming concepts require a significant amount of practice, and some students appear to not spend enough time on studying or practical exercises. In addition, first-year students often lack self-confidence due to a lack of experience and self-efficacy. One participant mentioned that providing support to other peers could increase a student's confidence, self-awareness and enthusiasm for learning. Some teachers have also indicated that a lack of diversity in lab questions is one of the causes of students' problems. One participant said: "*Our questions are often about getting students to write programs; we do not do enough of looking at other programs and analysing other programs.*" This theme summarised most of the difficulties in introductory programming courses; however, difficulties are often an unavoidable but important part of learning. So, tailoring different learning practices could support students and improve their learning experience.

## **Theme 2: Teachers' attitudes towards peer assessment**

Teachers described their personal experience using peer assessment in their classrooms and their positive or negative viewpoints regarding incorporating peer assessment in programming courses. They also determined ways to overcome the challenges of using peer assessment in programming courses. This topic includes the following sub-themes, summarised in Table 4.6:

- Personal experiences of using peer assessment
- Benefits and challenges of programming assignments
- Ways to overcome challenges

Theme	Subcategory theme	Supporting quote	Frequency
<b>Usage</b>	In programming	<i>"I did peer assessment with Python, but some feedbacks are useless, and some of them are valuable."</i>	1
	Other courses	<i>"I apply it but not in programming courses."</i>	4
	Never	<i>"I have never been used it. But I imagine it would be effective if we make students assess each other's programs."</i>	6
<b>Benefits and challenges</b>	Benefits	<i>"It helps you to understand what happens in the class."</i>	7
	challenges	<i>"The worrying thing is if it doesn't work, so you put people together. They don't understand the problem well; then they see really that as a group."</i>	4
<b>Ways to overcome challenges</b>	Marking scheme	<i>"Providing students with a marking scheme might help them and make them understand the criteria."</i>	7
	Anonymity	<i>"I suppose if not anonymous, you will increase the risk of being unfair and ineffective."</i>	3
	Formative assignment	<i>"If it's not a formative assessment, I have to assess all of the students' assessments; that is an extra effort."</i>	3
	Advanced stages	<i>"It's better getting second or third-year students rather than the first year, students at advanced levels have a bit more experience, and the activity could be more effective."</i>	3
	Practice	<i>"Students can assess a piece of work that does not belong to any member, and then the class discuss the feedback and their reasons for their assessment."</i>	1

**Table 4.6. Theme 2: Teachers' attitudes towards peer assessment**

From this sample, only one teacher from Newcastle University had used peer assessment in a programming course for a long time. Other teachers from Newcastle University (n=4,



36%) had used peer assessments in other advanced courses, such as in team projects modules and in Software Engineering module. Teachers from Saudi universities (n=5, 45%) had never used peer assessments before. Many teachers, either at Newcastle University and Saudi universities, had a positive opinion of using peer assessment in introductory programming courses (n=6, 55%); they emphasised that peer assessment would engage students in the learning process, that it would make students responsible for their learning, and that it would make them active learners. Some teachers were more negative about using peer assessments in introductory programming courses (n=5, 45%): the biggest challenge that prevents teachers from using peer assessment is the lack of students' programming knowledge and experience in assessment. They thought that peer assessment could be effective in advanced programming courses or at postgraduate levels, with students who have at least basic knowledge, who are able to assess peer more accurately, as they thought that students at advanced levels are better in assessment than first-year students.

Teachers mentioned many ways to succeed in producing the desired result in peer assessment with first-year students. Firstly, to counter the lack of experience with assessment, a clear marking scheme can reduce incorrect assessments and increase the acceptance of peer assessment as the students can see the criteria that teachers use to assess their work. One participant said: *"maybe give students a clear marking scheme, and it might help them understand the requirements and how to assess peers accurately"*. Secondly, to reduce hostility and bias, assessments can be done anonymously. One participant said: *"I suppose if it is not anonymous, you will increase the risk of being unfair and ineffective"*. Thirdly, peer assessment can be formative so that students can be less worried about wronging others or about giving an incorrect assessment. Besides, students can practice peer assessments so they can calibrate their scores with their peers. It is one way to get them used to providing feedback and listen to others. Since two teachers stated that they would have to make extra effort when applying such exercises with *"first-year programmers who were unqualified"*, the experimental method examines first-year

students' accuracy in assessing peers' work. This theme showcases teachers' experiences with peer assessment and teachers' perspectives.

### **Theme 3: Strategies preferred to carry out the peer assessment**

This theme illustrates teachers' opinions on appropriate ways to implement peer assessment in introductory programming courses. It includes the following sub-themes, summarised in Table 4.7:

- Type of peer assessment activity
- Teacher's role
- Marking scheme and its aspects

<b>Theme</b>	<b>Subcategory theme</b>	<b>Supporting quote</b>	<b>Frequency</b>
<b>Type of peer assessment</b>	Formative	<i>"I would be nervous about the module if I made it part of the grade. So, I will avoid that until I am sure it is working well and accurate."</i>	9
	Summative	<i>"I could allocate 5% until students were familiar with assessing, then I can put 10%, but I have to review their assessment."</i>	2
<b>Teacher Role</b>	Moderator	<i>"Preparing marking scheme, setting up, observing, making sure students understand the criteria and dealing with queries."</i>	7
	Review the assessment	<i>"I will evaluate and review the evaluation process."</i>	2
<b>Marking scheme</b>	With criteria	<i>"Marking schemes for students like for demonstrators, they need clear criteria."</i>	10
	Without criteria	<i>"To find students creativity in the assessment, I can encourage students to assess their peers based on their own standards."</i>	1

Theme	Subcategory theme	Supporting quote	Frequency
	Detailed rubric	<i>"The detailed rubric helps us to make student knows how they can improve to get a full mark, but it would be very long."</i>	5
	Concepts rubric	<i>"Criteria with category help the student understand the context of a good program and what is not."</i>	5

**Table 4.7. Theme 3: Strategies preferred to carry out the peer assessment**

Most teachers emphasised that if peer assessment was used in programming courses, it should be used as a formative assessment strategy (n=9, 82%) to encourage students to comment on the work of their peers without the pressure of losing marks. One participant said: *"I would be nervous about the module if I make it as a part of the grade. I will avoid that until I am sure it is working well and accurate."* Few teachers (n=2, 18%) suggested making it summative, but said they would review the assessment of peers and decide to grant the peer efforts 10% weight or lower from the official score. Teachers determined their role during applying peer assessment. Many teachers (n=7, 64%) mentioned managing peer assessment and providing support during the activity, and they expressed that they do not want to put in an extra effort when using this activity. However, few teachers mentioned that they would evaluate and review the evaluation process (n=2, 18%) because they do not trust first-year students' assessments.

In order to establish a suitable marking guide for new students in a peer assessment activity, two forms of the marking guide were prepared (see Table 3.4 and Table 3.5 in section 3.4.3). The detailed form is named "rubric", and the general form is named "marking scheme". These forms were similar in criteria but differed in the scales and the descriptive details. Ten teachers agreed on the importance of using a marking scheme with first-year students, and they agreed that all the criteria are important, but some teachers edited some phrases to avoid subjective criteria. The interviewer asked their opinions about the most suitable form for first-year students. Teachers were divided equally over which one was better for students. Some teachers preferred to display

descriptive details in each scale in the marking scheme. One participant said: *“For assessing it is better to give detailed rubric, the detailed one help us to make student know how they can improve to get a full mark, but it would be very long.”* The other half of the teachers preferred the marking scheme form, as criteria that have main concepts might be helpful and better for first-year students as they are simple. One participant said: *“Criteria with category helps the student understand the context of a good program and what is not”*. Only one teacher suggested encouraging students to assess without criteria to *“Find students thought about the code and find the creativity in the assessment.”* Therefore, the experimental method was used to examine both forms by employing a pilot experimental method to determine the differences between the effectiveness of the forms.

#### **Theme 4: Utilising the data collected from the peer assessment**

Under this theme, the teachers talked about the possibilities of using the data collected from peer assessments in a helpful way. When data are collected, tracked and analysed over time, students’ progress can be measured. Table 4.8 shows the content of this theme:

- Overall performance
- Pointer to learning objectives
- Invalid data

<b>Theme</b>	<b>Subcategory theme</b>	<b>Supporting quote</b>	<b>Frequency</b>
<b>Overall performance</b>	Struggling with a particular topic	<i>“It could pick up struggling in a specific topic, if I found low marks in peer assessment.”</i>	1
	Struggling with marking scheme	<i>“If all students are struggling with a specific criterion, I can clarify the marking criteria more in-depth.”</i>	1
<b>Pointer to achieve learning objectives</b>		<i>“Maybe if I put in each marking scheme relation with the learning objectives.”</i>	1

Theme	Subcategory theme	Supporting quote	Frequency
Invalid data		<i>“Data from peer assessment, especially from first-year students, are invalid and unreliable; it is difficult to use it as a pointer to students’ progress.”</i>	8

**Table 4.8. Theme 4: Utilizing the data collected from the peer assessment**

Most of the participants were not enthusiastic about utilising such data (n=8, 73%); they thought peer assessment data were not reliable data. They thought it was difficult to use peer assessment data as a pointer to students’ progress because students at this stage are not qualified, and their data are inaccurate. In contrast, two participants (n=2, 18%) thought that peer assessment would help them see if the assessed students were struggling with a specific topic by looking at the agreement of reviewers’ comments, or if the assessors struggled with understanding the marking scheme’s criterion if they did not assess the peers’ work accurately. However, they conclude that these data can be gathered through traditional assessments. Although one participant commented that it could be used as a pointer to illustrate whether a student has achieved learning objectives, the participant said, *“Maybe if I put in each marking scheme relation with learning objectives”*. There may be data in the peer assessment that could be a good indicator of students’ performance or a predictor for failure, but this sample did not clarify any of this. Because many teachers believe that the data collected from students are invalid, the interviewer decided to ask students how this study could utilise peer assessment data in their favour in the second phase of the research.

These two methods, questionnaires and interviews, partly achieved the first research objective; they helped to ascertain students’ and teachers’ perspectives towards peer assessment in introductory programming courses. Teachers and students acknowledged the benefits of peer assessment for first-year students; they also discussed barriers to peer assessment. However, these obstacles did not outweigh the benefits of this tool, so teachers suggested some solutions. Moreover, students and teachers identified their

attitudes towards the main elements when applying peer assessment, such as type of the assessment, privacy, grades, teacher's role, and marking scheme. Although the marking scheme was a key factor in peer assessment, results from interviews found that half of the teachers preferred the marking scheme form, and half of them preferred the rubric form. Therefore, experimental studies were conducted to find a better form of marking scheme for first-year programmers. Some teachers questioned the students' abilities to do such activities, so the experimental method was aimed to measure the students' accuracy in assessing their peers' code. The following section answers the second research question and presents the result of the experimental methods used in this study.

#### **4.4 Results from experimental method**

The experimental method was employed to answer the second research question: Are first-year students who participate in peer assessment more likely to perform significantly better on programming skills than those who do not? This question finds the correlation between students' assessment and teachers' assessment, to measure the accuracy of students' assessment, and to ensure their abilities in assessment. If the result of the correlation is medium or strong, the researcher can measure the students' programming performance after applying peer assessment. This method is divided into the pilot-experiment method and the pseudo-experiment method.

A convenience sampling technique was employed in this method because it was "simply available to the researcher by virtue of its accessibility" (Bryman, 2008, p. 190). Participants in this method were chosen from PNU because the researcher is a member of the academic staff at the Deanship of Community Service and Continuing Education so access to participants at the College of Computer Science and Information System was easier. Further, it was also possible to repeat the method if the participants did not participate. The same held for Newcastle University, as the researcher was studying there, and there were academic staff who volunteered to help the researcher with the distribution of two forms of marking guide and to conduct the experiment method, which made it easier to access the samples. In addition, there was a large and widely dispersed population of

CS students studying introductory programming courses in these universities, especially at undergraduate level.

#### **4.4.1 Demographic data: Pilot-experiment method**

The pilot-experiment method considers two points: (a) which method gives the best correlation between first-year student and teacher assessment, a rubric form or a marking scheme form? And (b) how long does it take to complete a task or experiment? The pilot-experiment method was applied in two different universities with undergraduate students who were studying introduction in Java programming language. At PNU, the experiment was conducted on the 27 February 2019, and at Newcastle University, it was conducted on the 29 April 2019. A total of 42 participants participated in the pilot study; (n=25, 60%) were recruited from the College of Computer and Information Sciences of PNU, KSA, and (n=17, 40%) came from the School of Computing at Newcastle University, UK. All of the participants from PNU were female, and only three participants from Newcastle University were female, so the total number of female students was (n=28, 67%). The other students were male (n=13, 31%). Students were asked about their programming experience; although all of the students were studying the first stage of programming, (n=22, 53%) considered themselves to be novices, (n=9, 21%) considered themselves to be competent, (n=1, 2%) considered themselves to be proficient, and (n=10, 24%) did not decide. Moreover, (n=32, 76%) of the participants had never used peer assessment before, (n=4, 10%) of the participants had used it, and (n=6, 14%) of data are missing. Table 4.9 shows demographic data of the pilot-experiment method.

<b>Variable</b>	<b>Item</b>	<b>Frequency</b>	<b>Percentage</b>
<b>University</b>	PNU	25	60%
	Newcastle	17	40%
<b>Gender</b>	Male	13	31%
	Female	28	67%
	Missing	1	2%
<b>Programming experience</b>	Novice	22	53%
	Competence	9	21%
	Proficiency	1	2%

Variable	Item	Frequency	Percentage
	Missing	10	24%
Peer assessment experience	Yes	4	10%
	No	32	76%
	Missing	6	14%

**Table 4.9. Demographic data of pilot-experiment method**

Before the experiment, the researcher ensured that all of the participants in the two universities were at the same level of their education and had almost the same curriculum. During the experiment, the first 15 minutes were spent defining the experiment's aim, explaining the content of the marking guide, and distributing consent forms, marking schemes and rubric forms (see section 3.4.3). Next, the students were asked to use the marking scheme to assess one sample answer. Students took around 12 minutes to read the code and assess the answer using the marking scheme. They were then asked to use the rubric to assess another sample answer to the same question. Students took around 17 minutes to read and assess another answer using the rubric. Each student who participated in this experiment assessed two answers; the first was assessed using the marking scheme, and the second with the rubric. It is noteworthy that the peer assessment activity was not part of the students' official coursework in both studies. It was an optional external activity. The actual student marks have not specified any score for this activity.

For data pre-processing, missing data were filtered out; for example, students' forms that only contained personal information without completing both marking guide forms were excluded from the pilot study analysis.

#### **4.4.2 Determining the optimal marking guide form**

A paired-sample t-test was conducted to determine which marking scheme best correlates the students' assessment and the teacher assessment – rubric or the marking scheme. Table 4.10 shows that no statistically significant difference was observed between the students' scores based on the rubric ( $M = 5.05$  out of 10,  $Sd = 1.8$ ) and those based on the marking scheme ( $M = 5.52$  out of 10,  $Sd = 2.03$ ),  $t[41] = 1.27$ ,  $p > 0.05$  [two-tailed]. The mean increase in the marking scheme scores was 0.47 with a 95% confidence



interval, ranging from -.28 to 1.24. Due to these results, the marking scheme form was selected to use in the empirical study. Moreover, during the experiments, the rubric took longer to read and prolonged the decision-making process by about 5 minutes. Furthermore, many students expressed their preference for the marking scheme, perhaps because they were unfamiliar with the use of the rubric form in programming courses.

Paired-Samples Test					
Test	M	SD	t	df	Sig.
Scores—marking scheme	5.52	2.03	1.27	41	.213
Scores—rubric	5.05	1.79			(N. S.)

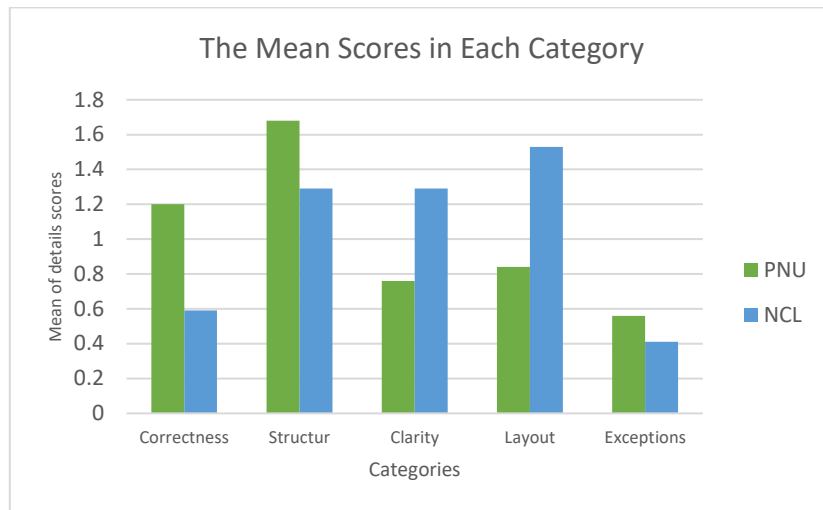
**Table 4.10. Comparison between the means of the rubric and marking scheme**

However, there were some differences when the detailed scores for each category were considered. A paired-sample t-test was conducted between the mean of each category in the marking scheme and the rubric. A statistically significant difference was observed between the students' mean scores in two categories: correctness and structure (Table 4.11). Concerning correctness, the mean of the students' scores with the marking scheme was  $M = .95$  out of 3,  $Sd = .71$ , and with the rubric, it was  $M = 1.83$  out of 3,  $Sd = .87$ ,  $t[41] = -5.49$ ,  $p < 0.001$  [two-tailed]. Thus, the mean was higher in the rubric by 0.88 with a 95% confidence interval, ranging from -1.20 to -.56. Concerning the structure category, the mean with the marking scheme was  $M = 1.52$  out of 2,  $Sd = .77$ , and with the rubric it was  $M = .90$  out of 2,  $Sd = .66$ ,  $t[41] = 3.90$ ,  $p < 0.001$  [two-tailed]. The mean was higher in the marking scheme at 0.61 with a 95% confidence interval, ranging from .30 to 3.90. Other categories did not have significant differences. This may indicate that if teachers are concerned to teach their students to measure the correctness of the code, they may need a detailed scale (e.g., rubric form) to specify how the program adheres to requirements. While code structure may not need to be elaborated in scale, as it can be measured by using, for example, yes, partly, and no scales. More quantitative data are needed to verify such interpretation.

Paired-Samples Test						
	Test	M	SD	t	df	Sig.
<b>Correctness</b>	<b>marking scheme</b>	.95	.71	5.49	41	.000 (0.01)
	<b>rubric</b>	1.83	.87			
<b>Structure</b>	<b>marking scheme</b>	1.52	.77	3.90	41	.000 (0.01)
	<b>rubric</b>	.90	.66			
<b>Clarity</b>	<b>marking scheme</b>	.98	.68	1.08	41	.285 (N. S.)
	<b>rubric</b>	1.14	.59			
<b>Layout</b>	<b>marking scheme</b>	1.12	.81	1.94	41	.059 (N. S.)
	<b>rubric</b>	.82	.58			
<b>Exception</b>	<b>marking scheme</b>	.53	.51	0.23	41	.822 (N. S.)
	<b>rubric</b>	.50	.51			

**Table 4.11. Comparison between the means of categories**

As the marking scheme form was chosen in the experiment, the researcher found different scores between the two universities' students for each category. Figure 4.9 illustrates the differences between the two universities in the mean of the detailed scores for each category. From the data it can be observed that neither university is distinct from the other in all of the categories, but there is an equal distinction by category between the two universities. For instance, the Newcastle University students had higher mean scores for clarity and layout, whereas PNU students had a higher mean score for correctness and structure. Thus, results must be interpreted by their teachers. The scale in the chart is between 0 and 2 because the mean of the detailed scores for each category was displayed rather than the mean of the total score.



**Figure 4.9. Comparison between PNU and Newcastle University scores in each category**

In summary, the marking scheme form was selected to be used in the peer assessment activity with first-year students because these students need to focus on concepts of criteria rather than on scale quality; the marking scheme form is also simple to use and easy to follow. However, teachers need to decide what they want to achieve with peer assessments - is it for learning or for making a judgment - and then select a suitable form accordingly. The differences in details scores may ultimately be the result of the differences between teaching methods. This finding must be compared with other research to find the appropriate interpretation.

#### **4.4.3 Marking guide development**

The marking scheme and rubric were imperfect when the pilot study was run; these forms have gone through several stages of development. Part of the initial marking scheme form is shown in Figure 4.10. The purpose of displaying the initial one is to help teachers avoid repeating mistakes while designing a students' marking guide form. For instance, under each category, there were three open-ended questions. However, many students only filled in the questions in the first two categories (correctness and structure), and they left the other categories empty. It may be that they did not complete the open-ended questions because they were bored or because the following categories (clarity, layout, and

exceptions) did not need open-ended feedback. Thus, open-ended questions were put at the end of the assessment instead of having some for each category.

Furthermore, a student should determine the score for each category to justify their choices on the scale. Although the researcher noticed that many students were able to critique and comment, they assigned a high score despite identifying errors. Thus, the decision to remove the score section was taken. Instead, the system can decide the score based on the student reviewer's choices. If the reviewer's choice corresponded to the teacher's choice, the student was awarded one score on this criterion. In some situations, the tutor gave half a point for the nearest choice to the tutor's choice.

Each piece of work was assessed by two assessors; for example, there was a column for the second assessor to agree/disagree with the first assessor assessment and write the reason. However, the second assessor could read the comments of the first assessor and either agree or disagree with their evaluation. After experimenting, the researcher noticed that the second assessor was sometimes affected by the opinion of the first assessor, so the decision to cancel the presentation of the opinion of the first assessor was taken. Additionally, the researcher made minor changes in the correctness criteria formulation to make it clearer for the students in the empirical study. As a result, the marking scheme (see section 3.4.3, Table 3.4) was the final form used in the pseudo-experiment method.

Assessor	Category	Criterion	Yes	Partly	No	Not applicable	I do not know
First Assessor	Correctness	Program runs correctly					
		The program produces correct output as specified in problem description.					
		The code handle errors.					
	Suggestion	What the programmer did right? ----- What the programmer did wrong? ----- How to improve for the future? -----				Total	4
Second Assessor	Agreement	<input type="checkbox"/> Yes <input type="checkbox"/> No Why: .....			Total		

Figure 4.10. Sample of initial marking scheme form

#### 4.4.4 Demographic data: Pseudo-experiment method

The pseudo-experiment method assessed how close the first-year programming student assessments were to the teachers' assessment across a large sample. It also investigated the influence of peer assessments on first-year programmers' performances in programming courses. In the pseudo-experiment study, the experimental group had 170 participants, and the control group had 162. All participants from the College of Computer and Information Sciences at PNU were studying the introductory Java programming language in the first year and were novice programmers when they participated in the study on the 14, 15 and 16 May 2019. Students themselves determined the programming experience of the experimental group; (n=99, 58%) of participants categorised themselves as novices and (n= 40, 24%) thought that they were competent when they participated in the peer assessment activity. None of the students selected the proficient level in this experiment, and (n=31, 18%) of the responses were missing these data. Students were asked about their peer assessment experience; (n=137, 81%) of students had not used peer assessment before, only (n=6, 3%) had used it, and (n=27, 16%) had missing data. Table 4.12 shows demographic variables of participants in the experiment method.

Variable	Item	Frequency	Percentage
<b>Group</b>	Experimental	170	51%
	Control	162	49%
<b>Programming experience</b>	Novice	99	58%
	Competence	40	24%
	Proficiency	0	-
	Missing	31	18%
<b>Peer assessment experience</b>	Yes	6	3%
	No	137	81%
	Missing	27	16%

Table 4.12. Demographic data of pseudo-experiment method

The experiment took half an hour to conduct. The researcher obtained students' consent to collect their two mid-term exam scores for this research because this experiment took place between two mid-term exams conducted by the teacher. The researcher asked the

participants to assess a random selection of quiz answers using the marking scheme. The students took 10 minutes to read the code and assess the answers using the marking scheme (see an example of students' assessment in Appendix C). In contrast, the students in the control group completed the lab as usual. Thus, the students took part voluntarily and did not have an advantage over those who did not take part. Also, there was no overlap between students who took the quiz and students who participated in the experiment. The participants received a thank you certificate, and two hours of voluntary participation were added to their skills record.

Some students were excluded from data pre-processing due to missing data; for example, some participants provided only personal information without completing the marking scheme form. Students for whom the researcher could not obtain their mid-term scores were excluded as well (see an example of data in SPSS in Appendix D).

#### **4.4.5 Correlation between students' assessment and teacher assessment**

An empirical study with a large sample was conducted ( $n = 170$ ) to determine how close the computer programming students' assessments of their peers' assignments were to the teacher's assessment. Although the experiment was a formative peer assessment, the accuracy of the assessment for participants was calculated by comparing the peers' choices with the tutors' choices in each criterion. Then, the total scores in the peer assessment for each student assessor was determined. Following this, the mean of the accuracy scores for all of the students in the peer assessment activity was calculated ( $M = 5.65$ ,  $Sd = 2.12$ ). Further, the result found that the Person correlation coefficient,  $r = .451$ ,  $n = 170$ ,  $p < .001$ , thus, the student assessors and teachers were similar, at a moderately medium level. Interpretation of the correlation coefficient is small if  $r = .10$  to  $.29$ , medium if  $r = .30$  to  $.49$ , and large if  $r = .50$  to  $1.0$  (Pallant, 2001). The researcher tested the marking scheme with a large sample size to decrease the standard error of the mean. This is because with a bigger sample size, the mean of the sample becomes a more accurate estimate of the parametric mean; thus, the standard error of the mean was ( $SE = 0.16$ ), indicating the accuracy of the dataset. Thus, the students' assessments were similar to the teachers' assessments, at a moderately medium level.

#### 4.4.6 Impact of peer assessment activity on students' performance

An ANCOVA test was used to assess the effect of the peer assessment activity on the programming course scores. The peers' first mid-term scores were used to see if peers had equivalent skills, and the second mid-term scores were used to evaluate their programming skills after the peer assessment activity. The mid-term exams were selected because these are an official way of evaluating students' outcomes and can be used instead of pre-/post-tests. The independent variable was the peers' scores in the first mid-term exam, and the dependent variable was their scores in the second mid-term exam. The students were divided into two groups: students who had participated in the peer assessment experiment (experimental group) and those who had not (control group). Preliminary checks were conducted to ensure that there was no violation of assumptions of normality, linearity, homogeneity of variances, homogeneity of regression slopes, and the reliable measurement of the covariate. Table 4.13 shows the ANCOVA result, which indicated a significant difference between groups on the second mid-term scores:  $F(1,345) = 13.33, p < 0.001, \text{partial eta-squared} = 0.037$ . A significant relationship was observed between the first and second mid-term scores, as indicated by the large effect size of the partial eta-squared value of 0.458. Therefore, the result indicates that the peer assessment had a positive effect on the students' performance in the mid-term exam.

ANCOVA						
Source	SS	df	MS	F	p	PES
Midterm 1	483.894	1	483.894	248.020	.000	.418
Group	26.015	1	26.015	13.334	.000	.037
Error	673.105	345	1.951			
Total	18665.088	348				

Table 4.13. The effect of peer assessment on scores

This suggests that peer assessment may significantly impact student performance, as statistics have demonstrated in this study, but further work is needed to explore this. Additional qualitative data from first-year programmers about the possible impact of peer assessment would, for example, be beneficial.

#### 4.4.7 Students' preferences regarding peer assessment feedback

The researcher compared students' assessments with the teacher's assessment, and then prepared two types of feedback for each participant: written feedback and visual feedback. The aim was to ascertain which form is suitable to represent the feedback in peer assessments. There were four results in each participant's feedback: two visual results and two written results (see an example of visual results in Appendix E). After one week, on the 21, 22, and 23 May 2019, all of the participants in the experimental group (students' reviewers) received their feedback. The students took 15 minutes to read the results and fill in the questionnaires. Visual results that represent peer feedback was preferred by students ( $n=86$ , 66%) over written feedback ( $n=44$ , 34%). Table 4.14 shows students' preferences regarding peer assessment feedback.

Students Preference in Feedback			
		Frequency	Valid Percent
Valid	Visual	86	66.2
	Written	44	33.8
	Total	130	100.0
Missing	System	20	

Table 4.14. Students' preferences regarding peer feedback

The next week, on the 28, 29, and 30 May 2019, the researcher repeated the peer assessment activity with the same participants who received the feedback to determine its impact. Some participants were absent, so only 131 participants repeated the peer assessment activity. A paired sample t-test was conducted to evaluate the impact of written and visual feedback on students' reviewers. There was a statistically significant difference in the mean total peer assessment grade from the first time ( $M = 5.76$ ,  $Sd = 2.1$ ) to the second time ( $M = 7.19$ ,  $Sd = 1.8$ );  $t(131) = 7.31$ ,  $p < .001$  (two-tailed) as is shown in Table 4.15. The mean increased by 1.43 with a 95% confidence interval ranging from 1.4 to 1.8. The eta squared statistic (.29) indicated a large effect size.



Paired-Samples Test					
Test	M	SD	t	df	Sig.
First Peer Assessment	5.76	2.15	7.31	131	.000
Second Peer Assessment	7.19	1.8			

**Table 4.15. Impact of feedback on peer assessment activity**

The statistical evaluation suggested that feedback positively affected reviewers' assessment performance, but this result is not definitive. More work on qualitative evaluation from students and more testing of the ideas of peer feedback to enhance students experience in peer assessment feedback is needed.

#### **4.4.8 Open-ended questions**

There were three open-ended questions in the marking scheme form with free-text boxes in which students could write their opinions of the pros and cons of the code and how it could be improved. Answers to the open-ended questions were coded based on the criteria in the marking scheme. There were ten criteria divided into the following categories: Correctness, Structure, Clarity, Layout, and Exceptions. The researcher categorised students' answers according to the marking scheme's criteria and then determined how similar the answer was to the model answer. Whereas the students' responses to each question were compared with the model answer to find the level of similarity, the level of similarity between the student's opinion and the model answer was divided into three levels (low, medium, and high).

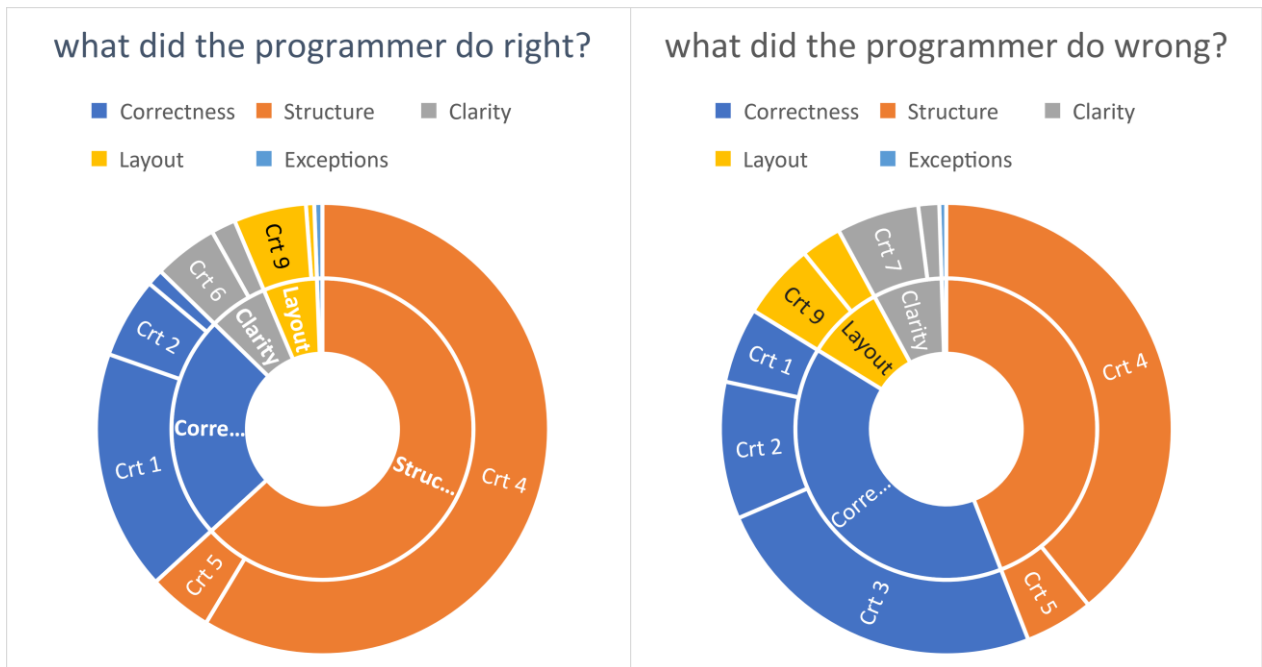
Most of the responses to the first question, "What did the programmer do right?" fell into two categories: the first category was structure, whereby 60% of the students chose the following criterion: correct choice of variable types and data structure. The second highest choice was correctness; 'Program runs correctly' was selected by 17% of the students. Other criteria selected by the students were not chosen by more than 6% (Figure 4.12). The highest level of similarity with the model answers was medium, whereby 55% of the students' opinions achieved medium similarity with the model answer.

The choices for the second question, "What did the programmer do wrong?", also fell into two categories: the first was structure, whereby 47% of the students chose the following

criterion: wrong choice of variable types or data structure. This indicated that it was easy for students' reviewers to judge the program's structure, especially to make a judgement on choosing variable types. The second highest choice was correctness: the code does not handle syntax and logic errors (27%), followed by the program does not produce the correct output as specified in the problem description (12%). Other criteria selected by the students were not chosen by more than 7% (Figure 4.11). Concerning the similarity with the model answers, around two-thirds of students' opinions had a medium similarity level with the model answer, and 27% had a high similarity.

Most of the responses to the third question, "How could it be improved for the future?" also fell into two categories: structure and correctness. In structure, 18% of students chose the following criterion: choice of correct variable types and data structure. The second highest category was correctness: the programmer should produce correct output as specified in the problem description (12%). Selections for the other criteria such as layout, clarity, and exceptions did not exceed 7.8%. However, around 41% of students wrote general recommendations such as practice, reading more codes, etc. As a result of this, the researcher rephrased the question so that it was more accurate by adding all categories under the question to direct students to select any of these categories then write their comments. Concerning the similarity with the model answers, 39.4% of the students' opinions were highly similar to the model answer.

To sum up, the students' responses focused mainly on the structure and correctness categories in the open-ended questions. Some students selected criteria from the clarity, layout, and exception categories, although most of the sample answers that had been assessed clearly ignored layout criteria (such as using code comments in different parts, and the indentation of the code). Despite this, many students did not mention that. This was maybe because their teacher did not require criteria considering clarity, layout, or exception categories from their students' during the assessment.



**Figure 4.12. Students' positive comments in open-ended questions**

**Figure 4.11. Students' negative comments in open-ended questions**

The analysis and findings from the experimental method can be summarised as follows: Firstly, the mean score achieved when using the marking scheme was higher than when using the rubric, although there was no significant difference. This is despite the fact that, during the experiments, the rubric took longer to read and prolonged the decision-making process, and many students expressed their preference for the marking scheme. Secondly, student assessors and teachers were similar when they were using marking schemes, at a moderately medium level. This means first-year reviewers' assessments were close to the teacher's assessment which is adequate to reach to desired benefits of peer assessment. Thirdly, students who participated in the peer assessment performed better in their mid-term exams than those who did not. Consequently, peer assessment significantly impacts student performance, as statistics have shown. The students' reviewers mainly focused on the structure and correctness categories when they assess their peers' work. Fourthly, students were more satisfied with visual feedback than written feedback that represents their performance as reviewers in peer assessment. The statistical results suggested that peer feedback positively affected reviewers' assessment

performance in peer assessment. However, these results were not definitive; more valid qualitative data are needed to support this claim.

#### ***4.4.9 Reliability and validity of scale***

To attain a high level of sampling accuracy in this experimental method, correct data have been carefully included by conducting data pre-process to minimise sampling errors. Data were cleaned to delete student records that did not include complete records, and outlier data that caused typographical errors were checked. The sample size was large as the data consisted of 170 records who conducted peer assessment. The standard error was calculated in the correlation between students' assessment and teachers' assessment, and it was small. The standard error is inversely proportional to the sample size, which means the larger the sample size, the smaller the standard error because the statistic will approach the actual value. In addition, the correlations were tested twice, in the pilot-experiment phase and in the pseudo-experiment phase, to ensure accurate results. They were then compared to previous studies in which correlations were statistically significant. Reliability was ensured using the stability method as experiments were repeated with similar data. Pilot studies were conducted twice - both with PNU students and with Newcastle University students – and were performed before the primary study to make sure the same results were measured whenever the method was used. Results were consistent when the experiments were repeated with similar data.

#### **4.5 Summary**

This chapter outlined the statistical results that were used to answer the first two research questions outlined in Chapter 1. The statistical analyses of questionnaires helped to explore the students' and teachers' perspectives regarding the awareness of benefits and challenges of peer assessment practice, and how they want to apply peer assessment in programming assignments. Number of factors were determined, such as: the type of the method, guiding tool, grading, privacy, and teachers' role. As the teachers' perspective remained unclear in some respects in the questionnaire, interviews were used for further clarification. The main results of the interviews with programming teachers helped to

ascertain how to build a marking scheme to guide students during peer assessment. The experimental method then measured the correlation between students' assessment and teachers' assessment to determine how accurate first-year students are in the assessment. Experiment results then indicated the suitable marking scheme form and measured the impact of peer assessment on students' performance. It also gave insight into what form of feedback was preferred by students' reviewers, i.e., visual feedback. However, none of the tests used determined the "cause" of selecting visual feedback in peer assessment. Furthermore, none of the methods allowed us to "understand" students' personal needs in peer assessments. With this aim in mind, the decision to build an initial prototype for a peer assessment that contains the main factors was taken, before gathering more data from students to understand programming students' needs when applying peer assessment, keeping in mind the teachers' opinions gathered from the interviews. The following chapter outlines the second phase of the study, which focuses on collecting qualitative data from first-year students.

## **Chapter 5. Second phase of the study**

### **5.1 Introduction**

The second phase of the study seeks to achieve two objectives: to determine students' needs and potential problems in terms of peer assessment and develop a prototype website with suggestions for peer assessment activities that meet programming students' and teachers' requirements. Therefore, the second phase started with an initial design of a peer assessment prototype that represents the main elements gained from the first phase. This chapter describes the development of the initial prototype. It will be argued that the initial prototyping needs to be continually developed according to the users' views which is why a user-centred design approach was followed. This approach includes users throughout the design and development of the peer assessment process. Focus groups and interviews methods were selected to understand users' requirements and their concerns regarding peer assessment, as well as to evaluate and release the prototype website. This chapter provides details on how these methods can be structured and how data can be analysed using thematic analysis. Finally, the chapter describes validity and reliability of the second phase to support the study's credibility, and it outlines the limitations of the approach used in this study.

### **5.2 Second phase of the research**

The second phase concentrated on building an initial prototype incorporating all of the findings concluded from the previous phase by following a user-centred design approach that adopts iterative qualitative methods to obtain information about students' expectations and critical issues and to evaluate to what extent the prototype represents users' expectations. The final version of the prototype, the "Peer Programmer prototype", was then the result of this phase. It includes all functional requirements set by the students and teachers who were part of this study and whose responses were assessed through qualitative and quantitative data gathered in the two phases.

The design of a prototype goes through many stages (Canziba, 2018). Firstly, an overall vision of the product is defined, and all objectives are identified. At this point, sketches are made and verbal descriptions during discussions with stakeholders to discover and explore their initial needs are collected. The product's main features are sufficient to represent the final product at this stage; there is no need to make the prototype identical to the end product. This allows the developer to discuss the key features with the stakeholders. The designer can then analyse the users' perceptions which leads to building the prototype phase; it is the longest part of the process as the designer must consider all the various options involved. Following this, the designer tests and refines the prototype based on the stakeholders' perceptions to enhance the overall design. Testing and refining should happen a number of times to ensure the prototype is ready to be revealed to the stakeholders. This study started with an initial design prototype, which was then improved multiple times to produce the final design that represents the users' requirements. The following section describes the development of the initial prototype.

### ***5.2.1 Initial design of the prototype***

The initial prototype includes the main features of the peer assessment that were identified in the first phase. It includes the following features:

1. Formative assessment: Students do not provide scores for their peers in the peer assessment activity because the activity is for learning and motivating students to comment on the work of their peers without the pressure of losing scores.
2. Marking scheme: The prototype contains the marking scheme form that was evaluated and recommended in the first phase (see section 3.4.3, Table 3.4). Results of the first phase emphasised the significance of the marking scheme to guide students in their assessment.
3. Grading: Peer grading in the prototype is not included as part of the student's final grade; it is used as a feedback activity only. The marking scheme form does not ask for grades at all. However, the system calculates the student's grade according to the reviewers' choices in the marking scheme.

4. Anonymity: The prototype hides the identity of each author and reviewer to increase the credibility of the peer assessment; this is according to the preferences voiced by students and teachers.
5. Online activity: The prototype is a website. Students highlighted in the first phase of this study that they would prefer an online activity since it is easier to guarantee anonymity by using usernames.
6. Individual activity: The prototype allows students' reviewers to evaluate peers' work individually without collaboration in assessment to increase the chance of understanding the peers' work and evaluate it without external influence.
7. Teacher's role: Many teachers concluded that they can upload the assignment, assign the assessment grades, and manage the process.
8. Students' role: In the prototype, students can solve an assignment, assess their peers' work, and get feedback through the assessment process. Figure 5.1 shows an example of feedback in the initial design of the prototype.

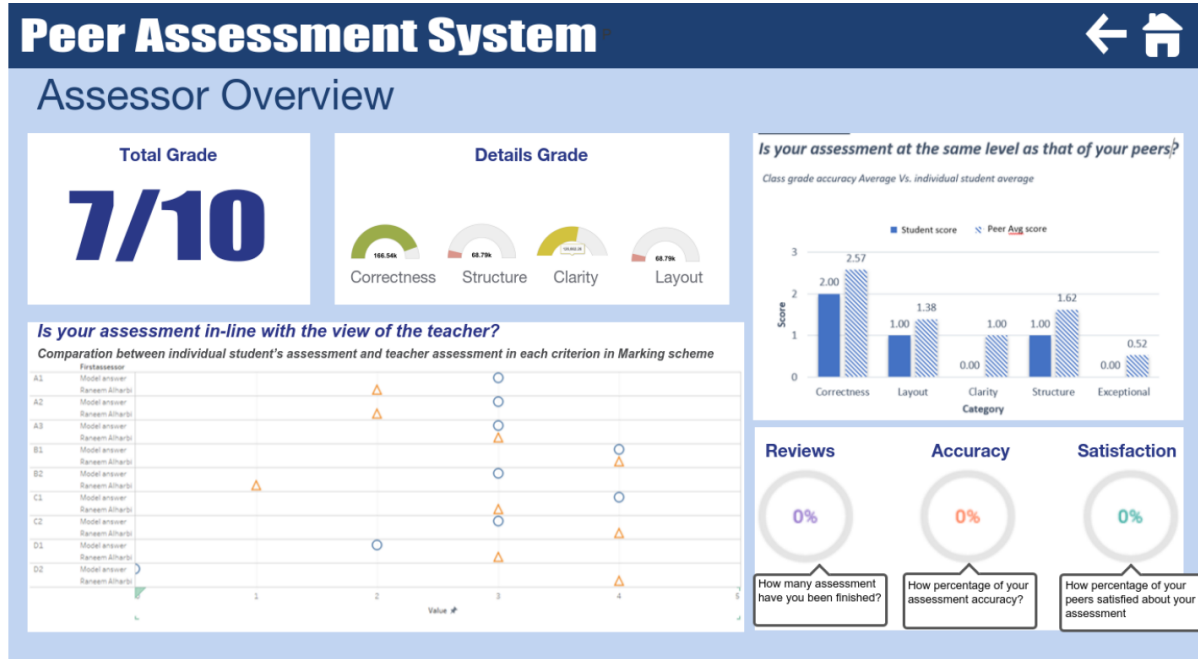


Figure 5.1. Initial design of the prototype



### **5.2.2 User-centred design**

The importance of users' involvement in the design and development processes of any user-driven system can no longer be ignored because of their contribution to the usage, effectiveness, efficiency, and safe product (Kahraman, 2010; Quintana *et al.*, 2013). User-Centred Design (UCD) is increasingly accepted as particularly useful in promoting a positive user experience (Schmidt *et al.*, 2020). UCD can be defined as an iterative design approach in which the needs and limitations of end-users are considered at every phase of the design process to enhance the understanding of the researcher (Galer *et al.*, 1992). This is accomplished by employing techniques, procedures, and methods throughout the product life cycle that focuses on the user (Baxter, Courage and Caine, 2015). The UCD stresses understanding users' needs and expectations during all design phases to build more user-friendly systems (Rogers, Sharp and Preece, 2002). As this study adapts an educational model (peer assessment) for which students are the primary users, the UCD was followed throughout the second phase of the project.

To produce a UCD, information needs to be gathered first; the researcher identifies the users' needs which is known as requirements gathering. This process includes gathering, analysing and assessing user data from different resources (e.g., surveys, focus groups, observations, and interviews). Having identified users' needs, a set of features are then created to define system capabilities that meet those needs. The final design should simulate the real design – in this study, of the peer assessment activity – before any resources needed for implementation are allocated; the design also needs to be tested for correctness before it comes into production. The UCD approach was used in this study to provide the programming students with the chance to express their opinions and express their expectations and concerns on what is considered appropriate when applying a peer assessment activity in programming assignments. This approach also helped to evaluate the prototype designed for this activity.

UCD involves several evaluation methodologies commonly used in a learning design context; one of these methodologies is the focus group method (Schmidt *et al.*, 2020). This approach was used to incrementally develop the prototype and assess the final

version of the peer assessment prototype specialised for programming students. The following section discusses in more detail the focus group method.

### **5.2.3 Focus group method**

A focus group is an interview where five to ten – ideally, six to eight – people are brought together to give their opinions or experiences around specific topics introduced by a proficient moderator in an open, non-judgmental environment (Baxter, Courage and Caine, 2015). The main benefits of a focus group are that the group dynamic raises issues that the researcher may otherwise have never thought of. The synergy of a group discussion may activate new ideas or stimulate participants, making them more willing and able to discuss their experiences openly and use their preferred language style with peers than they would with an interviewer (Baxter, Courage and Caine, 2015). Focus group discussions were selected in this study to collect student expectations and critical issues around implementing peer assessment in introductory programming courses. This relates to the third research question this thesis aims to address. This method was also selected to evaluate a prototype designed for the peer assessment activity. Therefore, focus groups were conducted iteratively to allow for incremental development of the prototype model; a form of cyclic system development process. Each prototype was developed from a previous version by editing the weak points or adding new features. By applying prototyping design within iterative design processes, the design of the Peer Programmer prototype is improved in a way that the researcher concentrates not only on intended learning outcomes but also on the user experience and usability of their designs.

As part of this study, three focus group iterations were conducted. Since participants from two countries (UK and KSA) were involved in this study, some discussions were conducted face-to-face, and some were conducted online. Each group consisted of 6-8 participants, and each focus group iteration had 3-4 groups. This means 30 users in total participated in each iteration. In some focus group sessions, the researcher facilitated and moderated the discussion; e.g., the sessions that occurred in the KSA. In other sessions that occurred in the UK, an external moderator was employed because the researcher was not able to reach the students at that moment, and to make students feel more at

ease when stating their opinions with a faculty staff. The following sections describe each iteration.

### **Study setting of the first focus group**

The first focus group iteration aimed to explore the perspectives of programming students toward peer assessment in programming assignments and to gather information on tasks students want to do in the peer assessment. The discussions began with a presentation outlining the session's goals and introducing the initial prototype of the website; a set of questions was then posed. This focus group was asked to talk about (1) general perspectives of peer assessment in programming courses; (2) the characteristics of an ideal peer assessment; (3) the tasks that students would like to perform with the ideal system; (4) students' opinions about the suggested ideas in peer feedback; and (5) the desired outcomes from a peer assessment system. The questions are set out in Table 5.1. Students were given the focus groups' main questions form (Appendix F) together with the participant consent form so that they were able to outline their ideas prior to the discussion. As samples were from two different languages, a translator translated the English version into Arabic; the translation was equivalent for each item. While creating questions for the focus group, the guidance published in Baxter, Courage and Caine (2015) was followed.

<b>Topic</b>	<b>Question</b>	<b>Duration</b>	<b>Goal</b>
<b>Warm-up</b>	What is your definition of peer assessment in programming courses?	~5 min	Transition to the topic, gauge participants' knowledge of the topic
<b>Key topic1- characteristics</b>	What characteristics of an ideal system that lets you do peer assessment?	~10 min	Get answers to find the required system characteristics
<b>Key topic2- needs</b>	What kind of information do you need to represent as feedback from the ideal peer assessment system?	~10 min	Get answers to find the students' needs from feedback.
<b>Prototype</b>	Display the initial prototype	~5 min	

<b>Key topic3-attitudes</b>	What are your attitudes toward these charts as feedback? (like/dislike)	~10 min	Find the students' opinion about the suggested feedback
<b>Key topic4-tasks</b>	What tasks would you like to perform with the system?	~10 min	Get answers to find the demanded tasks from the system
<b>Key topic5-outcomes</b>	Describe the desired outcomes from the peer assessment system.	~ 5 min	Get answers to find the desired outcomes.
<b>Warm-up</b>	Of all things that have been discussed, which one is most important to you?	~5 min	Reflect on discussion; bring closer to discussion
<b>Summary</b>	Does this summary capture what was said?	~5 min	Let participant validate/refute key findings

**Table 5.1. Discussion guide for the first focus group**

### **Study setting of the second focus group**

The second focus group iteration aimed to explore student expectations and critical issues around the second development of the prototype website. After analysing students' discussions from the first focus group, the prototype was edited. Another focus group discussion was then prepared to examine the new prototype's content. A website link was distributed to all participants to allow them to explore the prototype before the session. The focus group discussion contained four sections: prototype features, obstacles, improvement, and overall usage of the Peer Programmer prototype. Table 5.2 shows the discussion guide of the second focus group. Asking about features provided an insight into the areas that should be expanded upon to produce a good peer assessment system. It also helps the researcher to avoid making any changes to features that user truly enjoy. To make the right changes, users must be asked about what they were unhappy with regarding the prototype. After gaining general satisfaction knowledge from users, the researcher hears exactly what work well (or does not) in this prototype. The researcher wants to hear how users would do things differently with these features if they were in charge by asking them about areas for improvement. Asking about potential usage led to knowledge about the actual foreseen consumption of the activity; a close-ended question

forces the whole group to answer. Two forms were prepared for students prior to the focus group discussion: the consent form and the focus group questions form (it contains the same questions in the discussion guide of the focus group, to make each participant summarise their thought). Further, Arabic versions of all forms were created following revisions from an external translator.

Topic	Question	Duration	Goal
<b>Warm-up</b>	What words or phrases come to mind when you think about peer assessment for first-year programmers?	~5 min	Transition to the topic, gauge participants' knowledge of the topic.
-	Explore the prototype	~10 min	-
<b>Key topic 1- Features</b>	In what aspects was this prototype successful? What features most caught your attention?	~7 min	This question let users think about the features of this prototype.
	What features would make you more inclined to use this prototype regularly in the future?	~3 min	This would provide solid ideas for future features that could not be changed.
<b>Subtopic – pairing in assessment process</b>	Would you like to determine the optimal reviewers for a specific author during a peer assessment procedure? What are the preferred properties of reviewers?	~5 min	Get answers to find properties of reviewers that could support the author.
<b>Subtopic – visualization</b>	What is your opinion or attitude toward these data visualization? (like/dislike)	~10 min	Displaying some suggested visualizations to find their opinion.
<b>Key topic 2- Obstacles</b>	What are this prototype's key weaknesses? What are specific issues/concerns?	~7 min	Identify the specific issues that users face to figure out the prototype issue.
	How significant is the issue or concern you have with this prototype?	~3 min	How immediately the issue needs to be addressed.

Topic	Question	Duration	Goal
<b>Key topic3 - Improvement</b>	If you could choose a task of this prototype to eliminate, what would you choose?	~5 min	Narrow and streamline the prototype to cut out any aspects users see as unnecessary.
	If you could choose a task of this prototype to develop further, what would you choose?	~5 min	Aspects of a prototype that might need revamping.
	If you could add any task to our prototype, what would it be?	~5 min	Adding what the users feel is necessary.
<b>Key topic4- Overall Usage</b>	Do you expect to use this system again in programming courses? Why or why not?	~5 min	Learn about the actual foreseen consumption and use of the prototype.
<b>Warm-up</b>	Of all things that have been discussed, which one is most important to you?	~5 min	Bring closer to discussion.
<b>Summary</b>	Does this summary capture what was said?	~3 min	Let participant validate/refute key findings.

**Table 5.2. Discussion guide for the second focus group**

### **Study setting of the third focus group**

The third focus group questions were similar to those of the second focus group; they were based on the Peer Programmer prototype. The same four topics made up the discussion: prototype features, obstacles, improvement, and overall usage of the peer assessment activity. However, additional questions focused on some concerns that students had raised in previous focus groups. In the previous focus groups, students were concerned about the programming proficiency level of reviewers who would evaluate the authors' work, thus, more questions were asked about the pairing author and reviewers. Besides, students in the previous focus groups were interested in the visualisation feature; thus, more questions were asked about the charts. Table 5.3 shows the discussion guide of the third focus group. Again, two forms were prepared for the third focus group discussions: the consent form and the focus group questions form (Appendix G). The website link of the prototype was again distributed prior to the discussion to allow participants to explore the prototype. Furthermore, focus group discussion questions were

converted into an online survey. This allowed students who felt uncomfortable participating in the discussion to explore the prototype and fill in the survey in their own time. A Jisc website was used to create the online survey (Jisc, 2021). The English version of all forms was translated into Arabic and then reviewed by a professional translator.

<b>Topic</b>	<b>Question</b>	<b>Duration</b>
<b>Warm-up</b>	Do you think there is any value for the assessor in peer assessment in programming courses? Do you think there is any value for the first-year programmers being assessed by his/her peers?	~5 min
-	Explore the prototype	~10 min
<b>Key topic 1- Features</b>	In what aspects was this prototype successful? What features most caught your attention?	~5 min
<b>Sub-topic – Matching</b>	As an author, would you like to determine optimal reviewers to assess your task? What are the preferred properties of the reviewer? Would you like to use the task's difficulty level to assign the optimal reviewer? Who will assign this knowledge level? Is it the author himself/herself, reviewers, or all of them?	~8 min
	Would you like the following output of matching: The matching process depends on an author's need; the author needs at least two proficient reviewers, if he/she is not proficient, and at least one proficient reviewer if he/she is already proficient. Therefore, there are two proficient students and two non-proficient students? Are there any changes you would suggest to the matching process?	~8 min
<b>Sub-topic – visualization</b>	What is your opinion or attitude toward these figures? How do important you find these figures? How useful? And how comfortable are you with these figures? Are there any changes you would suggest to any figure?	~8 min
<b>Key topic 2- Obstacles</b>	What are this prototype's key issues/concerns? How significant is the issue you have with this prototype?	~5 min
<b>Key topic3 - Improvement</b>	If you could choose a task of this prototype to eliminate, what would you choose?	~4 min

Topic	Question	Duration
	If you could choose a task of this prototype to develop further, what would you choose?	~4 min
	If you could add any task to our prototype, what would it be?	~4 min
<b>Key topic4- Overall Usage</b>	Do you expect first-year programmers will use peer assessment in programming courses? Why or why not?	~5 min
<b>Warm-up</b>	Of all things that have been discussed, which one is most important to you?	~3 min
<b>Summary</b>	Does this summary capture what was said?	~3 min

**Table 5.3. Discussion guide for the third focus group**

### **Data analysis procedure**

The following section describes the thematic analysis method and the coding procedure that were used in this study.

#### ***Thematic analysis***

Thematic analysis is a method for analysing qualitative data; it makes patterns detectable, thus the researcher discovers themes through the data that have been collected (Clarke and Braun, 2014). A thematic analysis includes a critical review of responses to define suitable codes that are relevant to the research question and to create themes from these codes. It is an accessible and flexible approach as it allows for many different ways to interpret meaning from the dataset. According to Braun and Clarke (2014), a thematic analysis involves six phases; these are also present in this study: 1) Familiarisation with the data by reading and rereading the data; 2) Systematic coding of the data that captures aspects relevant to the research question; 3) Generating a set of candidate themes by considering the semantic meaning and underlying concepts to form themes; 4) Reviewing the candidate themes on two levels; first, an initial review of the themes against the coded data to examine their fit; secondly, an advanced review of the themes against the full dataset; 5) Defining and naming themes that describe the scope and boundaries of each theme; 6) The final phase is writing up an analysis of the data. In the analysis, each theme



must be described; its number of occurrences must be stated; and examples from the data must be included as evidence.

### ***Coding procedure***

Coding is an analytical process that describes both semantic meaning within the data and underlying meaning (Braun and Clarke, 2012). According to Saldaña (2016), there are three cycles for coding; the first cycle, the second cycle, and a combination cycle. 32 coding methods make up these three cycles. Saldaña identifies the first cycle as coding strategies that occur during the initial data, when they are raw data. The second cycle is reorganising and condensing the large array of initial analytic details; it also requires coding strategies, but the coding is the result of, for instance, abstracting, classifying, conceptualising, prioritising, synthesising, integrating, and theory building. The hybrid method is positioned between these two cycles, and produces codes as a result of a combination of two or more coding methods. In the first cycle, an initial coding method was selected for this study. The *initial 'open' coding* method is a qualitative method that breaks down data into discrete parts, closely examines them, codes, and sorts them into categories or concepts and compares them for similarities and differences (Saldaña, 2016). A list of concepts was generated as part of the initial coding method; all participant data were coded.

The next cycle coding method used in this study was an *axial coding*; it extended the analytic work from initial coding. This coding was reconstructed divided data during the initial coding method. Besides, it was decided which codes were important ones and which are the less dominated ones, and rearranged the dataset. Redundant codes were removed, synonyms were grouped, and the best representative codes were chosen, in accordance with (Saldaña, 2016). Also, this coding method combined categories with subcategories and considered how they are related and how to specify the properties and dimensions of the categories.

The third phase of the coding in this study was *theoretical 'selective' coding*. The main themes of the study were identified, and the sub-themes that include all internal codes

were developed. In theoretical coding, all categories become systematically integrated and synthesised around the core category and represent the main theme of the study (Saldaña, 2016). The category list was clustered into higher order “key categories”. Key categories were examined if they were “saturated”- which means constant reading of the data failed to provide new information and that the category was well represented among participants. When saturation was completed, the key category was accepted, and the main themes were built. The frequencies of the categories’ appearances in the transcripts were used to give credibility to these categories. Thus, the analysis procedures went through several types of coding, open, axial, and theoretical, to determine the key themes of the dataset. Microsoft Word was used to code the data and extract the comments into a table as NVivo does not support the Arabic script.

#### ***5.2.4 Interview method***

Interviews were aimed to evaluate the prototype that was designed to represent a peer assessment activity; particularly, to hear the voices of the programming teachers. Interviews with teachers were valuable for better understanding their views on the final version of the prototype. They were also important to explore pedagogical practices because teachers are the ones that give shape and meaning to such practices in specific courses. Since programming teachers are aware of first-year students’ issues and their needs in programming courses, they can reshape the activity in accordance with these needs. Four topics were addressed in the interview discussion: prototype features, obstacles, improvement, and overall usage of the peer assessment website. There were also questions about the matching author and reviewers feature and the visualisation feature. Interviews with teachers followed the iterative discussions with students and took place only once because the prototype was refined multiple times based on the students’ feedback. Interviews were conducted online using the Zoom application due to the Covid-19 pandemic.

## Study setting

The semi-structured interviews included 12 questions to obtain teachers' perspectives on the peer assessment prototype. The interview questions were organised around five blocks: (1) personal questions; (2) prototype features; focusing on visualization and matching features; (3) prototype issues and how significant these issues are; (4) suggestions for improvement of prototype; and (5) questions about how teachers might use peer assessment in programming courses. Table 5.4 shows the discussion guide for the interviews.

Topic	Question	Duration
<b>Worm-up</b>	Personal information and individual peer assessment experience.	~5 min
-	Explore the prototype	~10 min
<b>Key topic 1- Features</b>	In what aspects was this prototype successful? What features most caught your attention?	~5 min
<b>Sub-topic – visualization</b>	What is your opinion or attitude toward these figures? How important you find these figures? How usefulness? And how comfort these figures? Are there any changes you would suggest to any figure?	~10 min
<b>Sub-topic- matching</b>	When matching authors and reviewers in peer assessment, what criteria do you recommend for pairing students? Would you like the following output of matching: The matching process depends on an author's need; the author needs at least two proficient reviewers, if he/she is not proficient, and at least one proficient reviewer if he/she is already proficient. Therefore, there are two proficient students and two non-proficient students? Are there any changes you would suggest to the matching process?	~5 min
<b>Key topic 2- Obstacles</b>	What are this prototype's key issues/concerns? How significant is the issue you have with this prototype?	~5 min

Topic	Question	Duration
<b>Key topic 3 - Improvement</b>	If you could choose a task of this prototype to eliminate, what would you choose?	~5 min
	If you could choose a task of this prototype to develop further, what would you choose?	~5 min
	If you could add any task to our prototype, what would it be?	~5 min
<b>Key topic 4- Overall Usage</b>	Do you expect to use peer assessment in programming courses? Why or why not?	~5 min

**Table 5.4. Interview questions to evaluate the prototype**

### **Data analysis tool**

Thematic analysis was used to identify patterns in the data. Each interview took approximately 40-60 minutes, and audio recordings were transcribed for analysis. Due to the sampling from two different countries, transcripts were produced in English and Arabic, based on the native language of the participants. After the transcription process, the transcriptions were reviewed for accuracy, then reread to create codes in English to unify the analysis process. An initial, open, axial, and selective coding was employed in this method. Microsoft Word was used to code the data and extract themes.

### **5.3 Validity and reliability of the methods used in the second phase**

The validity in qualitative data “might be addressed through the honesty, depth, richness and scope of data achieved, the participants approached, the extent of triangulation” (Cohen, Manion and Morrison, 2012, p. 133). This study attempted to ensure the validity of the data by triangulating the data through questionnaires, interviews with teachers, and focus groups with students. In this way, the qualitative data provided by teachers and students was checked against that from the questionnaires. Moreover, qualitative research can be tested for transferability, which considers unique cases as being valid examples in a wider setting (Cohen, Manion and Morrison, 2012), though these needs providing sufficient contextual information about the fieldwork (Lincoln and Guba., 1985), and detailed reporting of the study process. In accordance with Geertz’s (1973)

suggestion of thick description, the explanation of all details of the study design, of the contexts of the different modules, and of the processes for data collection and analysis can reduce the limitation of the mixed-method approach. For this reason, reflecting on all the phases of the research process was a useful step; for instance, extensive background information about the context of this study was provided, including details of how samples were selected; the policy procedure; the data collection procedure; the data analysis instruments; the transcripts; the data analysis decisions; and possible values of collected data. The aim was to provide a comprehensive understanding of the context, because it must be possible for another researcher to come along and do exactly the same study that has been conducted in this thesis and then get the same results to verify the conclusions, or conduct comparison of the findings to other situations. Thus, other researchers can reflect on the comparability of the research, and relate the research to their own contexts.

Regarding inter-rater reliability, there is a statistical agreement (e.g., Cohen's Kappa) and consensus agreement (e.g., all members agree on all codes) (Olson et al., 2016). In this phase, consensus coding was employed; a bilingual external Arabic researcher volunteered to analyse half of the discussions to confirm the codes and categories. The consensus agreement passed many steps in accordance with Olson *et al.*, (2016), 1) Each researcher performed open coding separately. The main researcher sent transcripts of half of the discussions to the external researcher to perform open coding. However, the main researcher and the external researcher firstly decided to code based on sentences as a unit, in which a complete sentence would be coded. Besides, they discussed how many codes to apply per sentence and researchers chose to strive for the one or two most important codes and not apply more than three codes in the one sentence. 2) Researchers exchanged open coding by email, so they reviewed each other's code before talking. They then met and discussed each code and its definition to unify codes. During the discussion, they merged, renamed, deleted, and unified codes. 3) Each researcher independently re-coded these transcripts using unified codes. Each researcher avoided creating new codes during this step, except, if there was a significant theme that had been missed previously.

In this case, the researcher defined it and added it to the unified coding. 4) Researchers met again to identify the number of times the code was applied by each researcher and to identify the areas where there was no agreement. They noticed similar frequency but in some situations each researcher applied the code to different sentences. Such situations were the focus of the discussion. This step resulted in an understanding and improvement of the codes. 5) Finally, the main researcher coded all transcripts and produced the themes, before meeting with the external researcher to examine these themes. The researcher and the volunteer researcher then discussed and agreed upon the themes. Thus, validity and reliability of qualitative methods were considered.

#### **5.4 Summary**

This chapter presented the second phase of the research. It first discussed the building of the initial prototype and its incremental developed leading up to the final version. The chapter described the user-centred design approach that was employed in this study. It discussed qualitative methods that were used to gather and investigate information on the participants' requirements, as well as to develop and evaluate the prototype website. It highlighted how iterative focus group discussions with students and interviews with teachers were used to produce an effective prototype. Each method and its study setting and data analysis process were outlined in detail. Lastly, the validity and reliability of the qualitative data were discussed. The next chapter presents the results obtained through the qualitative methods as well as the final design of the prototype. These are significant research findings of this study.

## **Chapter 6. Results of the second phase**

### **6.1 Introduction**

After exploring students' attitudes towards and their experiences of peer assessment in programming courses using quantitative scales (see Chapter 4), describing the participants' lived experiences in their own words was the next logical step. This chapter presents the results of the second phase of the study. The following questions were investigated:

1. What are student expectations and critical issues related to implementing peer assessment in introductory programming courses?
2. How can peer assessment, as a learning process, be integrated into introductory programming courses?

This phase mainly used qualitative data collection methods in the form of focus groups and interviews. Relevant demographic data are described for each method used; the procedures used to collect and analyse data, as well as the key themes that emerged from the data relevant to the research questions are discussed. The final version of the Peer Programmer prototype website is presented in this chapter. It includes requirements analysis, all relevant diagrams, and shows the users' journeys in the prototype. In the end, the chapter discusses one of the significant findings in this phase which is visualising peer feedback.

### **6.2 Results from focus groups**

The focus group method was chosen to answer the third research question: What are student expectations and critical issues related to implementing peer assessment in introductory programming courses? Hence, it was used to identify students' requirements - the elements they require and the problems they may encounter - regarding peer assessments in programming courses. Further, the iteration technique in this method was performed to answer the fourth research question: How can peer assessment, as a learning process, be integrated into introductory programming courses? This question

could be answered through developing and evaluating the Peer Programmer prototype website that contains all important elements of peer assessment. The following section describes the demographic data of the focus group.

### 6.2.1 Demographic data

The focus group discussions were iterated three times and conducted in the academic years July 2019, February 2020, and September 2020, respectively. The total number of participants that took part in all the iterations was 89. Of these, (n=87, 98%) were female. They were sampled from several Saudi female universities (PNU, Imam Mohammad Ibn Saud Islamic University, and King Saud University). There were two male participants (n=2, 2%) from Newcastle University. All of the respondents were attending or had attended computer programming courses at their respective universities in the departments of Computer Science (n=37, 41%), Information Technology (n=23, 26%), Information Systems (n=22, 25%), or Software Engineering (n=7, 8%). However, the participants were distributed across many knowledge levels. Some were studying their university’s advanced undergraduate computer program (levels 7 or 8), and they ranked themselves as having proficient programming skills (n=28, 31%). Others (levels 1, 2 or 3) ranked themselves as novices (n=21, 24%), and the largest group (n=39, 44%) classified themselves as competent (levels 4, 5, or 6), with 1% missing data. Some participants (n=14, 16%) reported that they had informally applied peer assessment with their friends and that they had assessed each other’s work without teacher intervention. However, most of the participants had never used peer assessment before. Table 6.1 shows the distribution of focus group participants’ demographic variables.

Variable	Item	Frequency	Percentage
<b>University</b>	PNU	70	79%
	Imam Univ.	11	12%
	King Saud Univ.	6	7%
	Newcastle Univ.	2	2%
<b>Gender</b>	Male	2	2%
	Female	87	98%



Variable	Item	Frequency	Percentage
<b>Department</b>	CS	37	41%
	IT	23	26%
	IS	22	25%
	SE	7	8%
<b>Programming Experience</b>	Novice	21	24%
	Competent	39	44%
	Proficient	28	31%
	Missing	1	1%
<b>Peer Assessment experience</b>	Yes	14	16%
	No	75	84%

**Table 6.1. Distribution of focus group participants' demographic variables**

The Covid-19 pandemic affected the data collection as it restricted the sample selection and reduced participation, particularly, participants from Newcastle University. This led to repeat focus groups with PNU students. Most of the participants were female because PNU is a female-only university.

### **6.2.2 Procedures of collecting and analysing data**

**First iteration:** The first focus groups took place in the period from the 9 to the 11 July 2019. Three focus group discussions were conducted with 28 undergraduate students from PNU, and one participant from Newcastle University - although the focus group was conducted by a volunteer from the faculty in Newcastle University who invited students to participate in the focus group discussion. This iteration aimed to explore the students' requirements in peer assessment. The questions were organised around five topics (see section 5.2.3, Table 5.1): (1) general perspectives of peer assessment in programming courses; (2) the characteristics of an ideal peer assessment; (3) the tasks that students would like to perform with the ideal system; (4) students' opinions about the suggested ideas in peer feedback; and (5) the desired outcomes from a peer assessment system.

**Second iteration:** The second iteration was performed three times between the 11 and the 13 February 2020, with 30 new students from PNU University who had not participated in the previous iteration. The second focus group iteration aimed to evaluate the Peer

Programmer prototype that was created based on students' feedback in the first iteration. Accordingly, questions were divided into four categories: (1) features of the prototype that caught their attention; (2) the prototype's key weaknesses; (3) how users could improve the prototype; and (4) potential students' usage of the peer assessment (see section 5.2.3, Table 5.2). Nobody from Newcastle University participated, although one member of staff who moderated the discussion asked their students to participate in the focus group voluntarily. Before discussions began, participants were provided with a link to the prototype, so they were able to explore its contents at their own convenience.

**Third iteration:** The third focus group iteration covered the period from the 30 September to the 16 October 2020. Six online discussions were conducted with 29 students from different Saudi Universities (PNU, Imam Mohammad Ibn Saud Islamic University, and King Saud University). Online discussions were held due to the COVID-19 pandemic. For Newcastle students, a video was prepared that described the prototype and an online questionnaire that contained discussion questions was provided; however, only one participant filled in the questionnaire. This iteration aimed to evaluate the Peer Programmer prototype, refine it, and release it. The discussion questions were similar to those of the second iteration; features, weakness, improvement, and potential usage (see section 5.2.3, Table 5.3). However, additional detail about the matching author and reviewers feature and the visualisation feature was included, as students in previous iterations had raised these aspects. Before the focus groups began, students were also provided with a link to the prototype.

In all iterations, students were given the participant consent form with the main questions form for the session to outline their ideas for the discussion (see Appendix B, Appendix F, Appendix G as examples). At the start of each session, the informed consent procedures were explained. The researcher explored the prototype, but without assigning any tasks to students, the discussion then took place. Each discussion lasted approximately one hour and fifteen minutes. After the focus groups concluded, the students returned the forms and were given an online certificate of thanks for their voluntary participation.

An analysis was performed after each iteration. Focus group discussions were held in Arabic and English and were recorded with a recording device or through Zoom. Subsequently, these recordings were transcribed based on their language: Arabic text via the Speechnotes application that supports the Arabic language, or Zoom for English transcripts. The transcriptions were then reviewed for accuracy. Because there was more than one focus group discussion, all answers to one question were combined under that question. Since participants had received a form that contained questions of the discussion so that they could outline their answers in advance, the participants were asked to return these forms at the end of the session. They were then used in the transcription process when answers were unclear or short so that the participants' written responses were added to the transcription for each question along with the recorded data. Microsoft Word was used to code the data and extract open, axial, and selective coding (see Appendix H as an example). Approximately 321 concepts were identified during the first cycle coding method (e.g., the suggestion of adding an assessment rating; adding chat rooms; displaying the level of assessor; anonymity etc.). Then, the concepts were compared against one another for similarities, and grouped and abstracted into categories - this resulted in 75 categories (e.g., prototype features; design issues; students' perspectives etc.). During the second cycle coding method, the relationships between the categories were explored; this resulted in 16 separate key categories. Finally, four key themes emerged from the data related to the students' perspectives of peer assessment: 'Perceptions of peer assessment', 'Credibility of the peer assessment', 'Structure of the prototype' and 'Appearance of the prototype'. The following sections describe each theme and its content.

### 6.2.3 Presentation of the main key themes

#### Theme 1: Perceptions of peer assessment

This theme demonstrates students' differing views regarding what peer assessment for programming courses is, how to conduct it, and its main aspects. It includes the following subthemes:

- Understanding the meaning of peer assessment.
- The value of being a reviewer and an author.
- Considering the aspects and data collected.
- Choosing when and where to use peer assessment.

Students mentioned that being assessed by or assessing peers without scoring was an interesting idea. One participant said: *“We follow the same peer assessing process informally; we correct each other’s codes and see how our colleagues solve problems in some assignment”*. Another participant stated, *“I usually search on the Internet to ask for help, but I have difficulties understanding expert suggestions. But, in this activity, we all learned in the same learning environment, so peers’ comments were not complicated”*.

Students were also asked to consider representing results after peer assessment data, even without teacher intervention. One participant suggested displaying feedback for both author and reviewers. Concerning the use of peer assessment in programming courses, most students (80%) agreed that it should be implemented, but with some specified conditions. They said that peer assessment should be employed until becoming proficient at programming or if the assignment had more than one solution. They also stated that peer assessment should be used for more difficult assignments. One participant said: *“I will use it if I face difficulty in the assignment to get feedback, but if I become a proficient programmer, I do not need such activity”*. However, some students (20%) said they would never use the peer assessment process because they thought peer assessment made them waste time. The data showed that students considered peer assessment and their roles in it successfully; they also understood when and how it could be applied.

Table 6.2 shows the sub-themes for the theme 'perception of peer assessment'. The first and second columns list the name of the theme and the sub-themes. The third column provides an example of a quotation from the interviews for each sub-theme. The final column displays the frequency of this sub-theme (i.e., how many times it was mentioned).

Theme	Subcategory theme	Supporting quote	Frequency
<b>Perceptions of peer assessment</b>	Understanding the meaning of peer assessment.	<i>"We follow the same peer assessing process informally; we correct each other's codes and see how our colleagues solve problems in some assignments."</i>	46
	The value of being a reviewer and an author.	<i>"As an assessor, peer assessment can increase my confidence in the ability to evaluate, and as an assessee, it could improve the quality of my code."</i>	17
	Considering the aspects and data collected.	<i>"A chart can be produced showing the student's progress in peer assessment during a period time."</i>	27
	Choosing when and where to use it.	<i>"I will use it if I face difficulty in the assignment to get feedback, but if I become a proficient programmer, I do not need such activity."</i>	59

**Table 6.2. Theme 1: Perceptions of peer assessment**

## **Theme 2: Credibility of the peer assessment**

The second theme relates to trust in peer assessment, which depends on the credibility of the peer feedback. This topic includes the following subthemes:

- Implementing an anonymity process.
- Peer assessment validation issues.
- The skills and abilities of reviewers.
- Using the matching technique.

Particularly the need for anonymity caught some students' attention (28%). Students felt anonymity was important to avoid or reduce bias and improve peer feedback credibility.

One participant said: *“Since the names are unknown, my assignment will be evaluated by students of the same class with all honesty.”* Also, one critical concern was the validation of the peer assessment process. Some students discussed their concerns of validation of receiving incorrect feedback and some ideas to validate peer assessment – e.g., they suggested multiple reviewers should assess the same assignment, and reviews should be compared between each other. One participant said: *“It could be accurate assessment if someone else assesses the assignment and has a similar view”*. This comment led to thinking about presenting the assessment of the reviewers in a way that makes it easier for the author to compare between their feedback.

The students also believed that the credibility of peer assessment feedback depends on the reviewer’s ability. Therefore, the interviewer asked students what skills a reviewer should have. The consensus was that they preferred students with high programming skills to review their work. Students commented that proficient students’ reviewers could have more knowledge in programming, thus they can identify more issues in a piece of code and do reviewing more efficiently. While weak students often focus on handling syntax errors that make very few changes in the code. The students also discussed how these programming skills could be determined if students are in the first year, and most agreed that determining the task difficulty level should be set as a value for choosing which reviewer should be assigned to the task. Building on this, they talked about how the task difficulty levels could be assigned, and most agreed that both self- and peer-assessments should be used to determine this value. However, they all agreed that self-assessment should account for no more than 20% of the difficulty level. One participant said, *“...I will allocate the lowest percentage to myself as sometimes, I do not recognise my mistake”*. Another one said, *“The reviewer’s ability must be determined before starting the peer assessment by a pre-test”*. Altogether, the students considered the quality of their reviewers’ feedback; they confirmed the importance of anonymity and multiple reviewers with the need for a proficient reviewer in each matching between author and reviewers. Table 6.3 shows the sub-themes for the theme ‘credibility of peer assessment’.

Theme	Subcategory theme	Supporting quote	Frequency
<b>Credibility of the peer assessment</b>	Implementing an anonymity process.	<i>“Since the names are unknown, my assignment will be evaluated by students of the same class with all honesty.”</i>	24
	Validation issues.	<i>“It could be accurate assessment if someone else assesses the assignment and has the similar view.”</i>	11
	The skills and ability of reviewers.	<i>“The reviewer’s ability must be determined before starting the peer assessment using pre-test.”</i>	50
	Using the matching technique.	<i>“I expect that the reviewer will be chosen based on his strengths, which are my weaknesses, to complete each other.”</i>	50

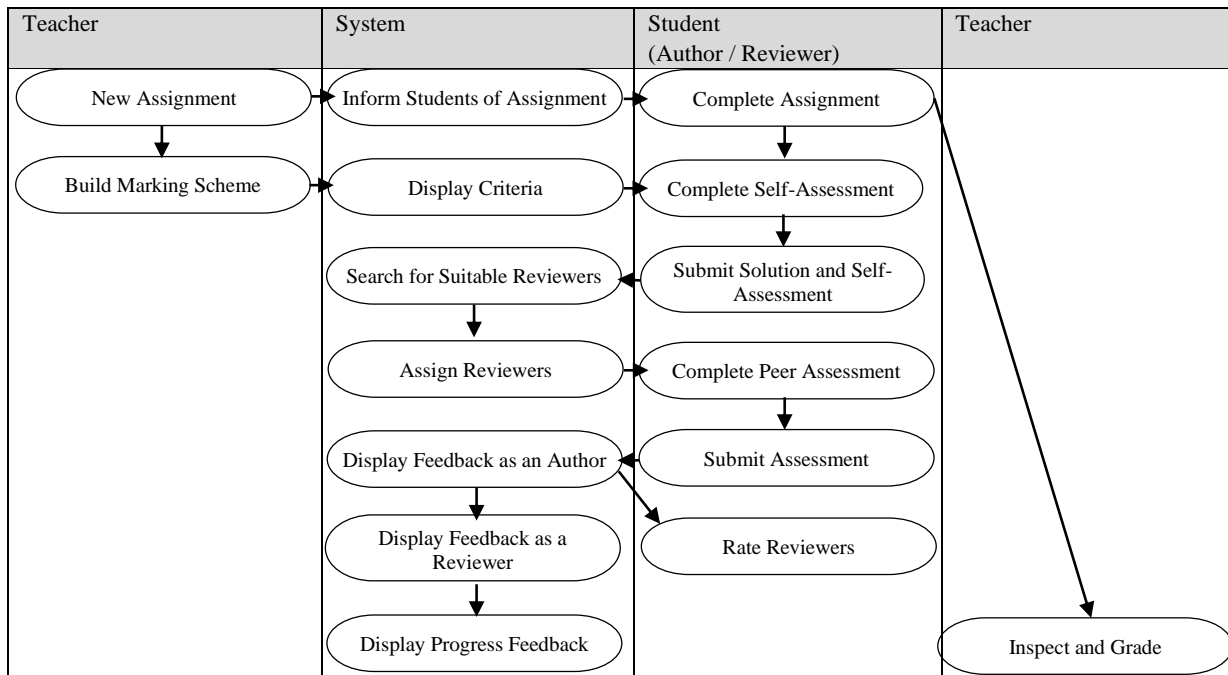
**Table 6.3. Theme 2: Credibility of peer assessment**

### **Theme 3: Structure of the prototype**

This theme concerns the composition of all the components of the peer assessment and tasks in the prototype. It contains:

- Organising the procedures of the peer assessment activity.
- Inclusion, elimination, or modification of tasks.
- Task processes of an ideal peer assessment.
- Navigation within the prototype.

Students determined the series of steps followed to conduct a peer assessment. An activity diagram in Figure 6.1 shows the peer assessment procedures agreed by students. The activity diagram describes the behaviour of the peer assessment system. It portrays the control flow from a start point to a finish point showing the different decision paths that exist while the peer assessment activity is being executed.



**Figure 6.1. Activity diagram of Peer Programmer prototype**

Students also discussed the tasks of the activity. For instance, when discussing adding more tasks, students most often mentioned adding rewards to motivate students to conduct such an activity (36%). One participant said, *“Is it possible to get a bonus for students who have been assessed well – or display his/her nickname in a dashboard – to keep them from feeling that their efforts are invisible? When the effort appears [in front of other people], he/she will continue [being diligent]”*. In a similar vein, students suggested adding self-assessment tasks to comprehend the criteria and make the reviewers more serious when assessing peers because they have assessed their own work before. Therefore, self-assessment was set as a precondition, which students were required to meet before they could conduct peer assessments, using the same criteria they had employed in their peer assessments.

Concerning the elimination of certain tasks, many participants wanted to eliminate rating reviewers after peer assessments. For example, one said: *“I assessed my code from my viewpoint, and they assessed my code from their viewpoint; there is no need to rate their assessment in return”*. As for modifying tasks, one student, for example, suggested



reducing the criteria in self-assessment. Some students also suggested to modify the navigation in some tasks; one participant, for example, suggested to: “*add navigation menu to the feedback page instead of displaying all charts on the feedback page.*” These results indicated that students considered various aspects of the structure in detail; they organised activities and suggested additions to, changes to, and deletions of some tasks Table 6.4 shows the sub-themes for the theme ‘structure of the prototype’.

Theme	Subcategory theme	Supporting quote	Frequency
<b>Structure of the prototype</b>	Organising the procedures of the activity.	<i>“Prevent the evaluation before solving the assignment and doing self-assessment, to avoid cheating and consider to all criteria.”</i>	20
	Inclusion, elimination, or modification of tasks.	<i>“I want to eliminate rating the reviewers after peer assessment, they have assessed me from their point of view, why do I return rate them.”</i>	90
	Task processes of ideal peer assessment.	<i>“Is it possible to get a bonus for students who have been assessed well – or display his/her nickname in a dashboard – to keep them from feeling that their efforts are invisible? When the effort appears [in front of other people], he/she will continue [being diligent].”</i>	24
	Navigation within the prototype.	<i>“Add navigation menu to the Feedback page instead of displaying all charts on one page.”</i>	17

**Table 6.4. Theme 3: Structure of the prototype**

#### **Theme 4: Appearance of the prototype**

This theme, which was raised in all the discussions of the second and third iterations, focuses on how the Peer Programmer prototype appeared to its users. It includes the following subthemes:

- Prototype design.
- Design errors.

- Usability.
- Opinions/attitudes toward figures.

Most of the participants' comments included statements about the prototype's appearance (positive or negative) and suggestions for adding or eliminating elements of the design. For instance, the participants stated that the most interesting and noteworthy feature were the charts (44%). The most often mentioned figure was the progress chart (Figure 6.2). One participant said: *"The progress chart helps to better understanding myself; it shows my strengths and weaknesses in each category, then I can find ways to improve"*. In contrast, other participants criticised some charts; they suggested deleting some of them. One participant said: *"The detailed grades can be found in the bar chart, thus we can omit the details grade chart"*. Besides, some participants made suggestions for arranging the charts. One participant said: *"no need to display all details in the chart, you can only show the first three elements and for more details the user can click on it"*. Some participants praised the usability of the prototype, one of them said: *"The prototype and its task is easy to use"*. Altogether, students were very interested in the prototype's appearance and commented that charts were a good way to represent peer feedback as charts captured their attention. Table 6.5 shows the sub-themes for the theme 'appearance of the prototype'.

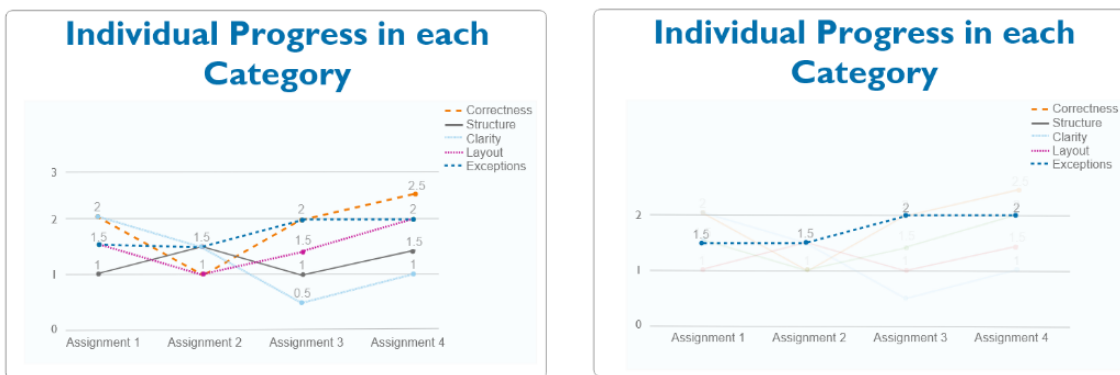


Figure 6.2. Progress chart

Theme	Subcategory theme	Supporting quote	Frequency
<b>Appearance of the prototype</b>	Prototype design.	<i>"The logo of the prototype needs to be modified; it is like a button."</i>	23
	Design errors.	<i>"The font and icons in the toolbar should be considered."</i>	18
	Usability.	<i>"The prototype and its task are easy to use."</i>	16
	Opinions/ attitudes towards figures.	<i>"Create a table containing criteria as items, then assign an icon to each reviewer. Then, select all the choices for each reviewer. So that the author can compare the opinions of the reviewers."</i>	59

**Table 6.5. Theme 4: Appearance of the prototype**

The students who participated in the focus groups offered positive opinions about the prototype and peer assessment. They suggested that it would function very well in programming courses when used at suitable times. They thought it best to decide when to use peer assessment. Participants suggested this type of assessment should be used when students are novices and want to develop their knowledge by connecting with more skilled peers. The participants also discussed the features that students would like to see in the peer assessment process. For example, the focus groups discussed the value of presenting the feedback as a visual method and how this attracts their attention; they suggested the benefits of self-assessment as a pre-condition to increase the credibility of the process, and indicated the need for rewards, either tangible or intangible to encourage them to continue peer assessment. Furthermore, the students raised a concern about the validity of the process and the abilities of the reviewers, which led them to suggest ideas for matching authors and reviewers during peer assessment instead of randomly selecting a reviewer. The structure and appearance of the prototype were improved due to the focus group discussions. However, the opinions of programming teachers were deemed necessary at this stage in order to adapt the prototype further. The next section presents the programming teachers' attitudes towards the Peer Programmer prototype website.

### 6.3 Result from interviews

After examining students' perspectives towards the prototype many times, allowing the teachers to describe their opinions was the final step before releasing the prototype. To this end, semi-structured interviews were conducted to evaluate teachers' perspectives of the prototype. This method was chosen to address the fourth research question: How can peer assessment, as a learning process, be integrated into introductory programming courses? The following section describes the demographic data of the participants.

#### 6.3.1 Demographic data

The teachers' interviews were conducted between the 15 October and the 5 November 2021. The total number of participants was six programming teachers: (n=3, 50%) from PNU and (n=3, 50%) from Newcastle University. Five of these participants were female (n=5, 83%), and one was male (n=1, 17%). All participants had different positions - assistant teacher (n=2, 33%), lecturer (n=3, 50%), and senior lecturer (n=1, 17%). Half of the participants (n=3, 50%) had used peer assessment before, two of them in programming courses. Table 6.6 shows the participants' university, gender, experience, position, and peer assessment experience.

Variable	Item	Frequency	Percentage
University	PNU	3	50%
	Newcastle	3	50%
Gender	Male	1	17%
	Female	5	83%
Experience	0–3 yrs.	1	17%
	3–6 yrs.	1	17%
	>5 yrs.	4	66%
Position	Senior Lecturer	1	17%
	Lecturer	3	50%
	Assistant teacher	2	33%
Peer assessment experience	Yes	3	50%
	No	3	50%

Table 6.6. Interviewees' demographic variables

This study aimed to interview teachers who had experience with peer assessment and those who did not have experience of using peer assessment. The study also aimed to interview teachers from both cultural backgrounds to gain a broader perspective.

### ***6.3.2 Procedures of collecting and analysing data***

Since the interviews aimed to evaluate the Peer Programmer prototype website, interviewees were asked about their opinion of the prototype. Questions were divided into four categories: (1) features of the prototype that caught their attention; (2) the prototype's key weaknesses; (3) how users could improve the prototype; and (4) potential usage of the system (see section 5.2.4, Table 5.4). At the start of each session, teachers were given the participant consent form as well as the main questions for the session to give them the opportunity to outline ideas for the discussion. Participants were also provided with a link to the prototype so they could test its contents. Next, the prototype was explored but without assigning any tasks to teachers. Each discussion lasted approximately one hour. All interviews were completed voluntarily and online using Zoom software.

The procedures used to analyse the interviews discussions were the same as those used for the focus group discussions. Thematic analysis was used to create key themes. Firstly, interviews with Saudi participants were conducted and transcribed in Arabic to allow them to fully express their experiences, while interviews with English participants were conducted and transcribed in English. All codes were developed in English to unify transcripts and identify the main themes. Next, open coding, axial and selective coding was used. Four key themes emerged from the data related to the teachers' perspectives of peer assessment: 'Perceptions of peer assessment', 'Concerns with peer assessment', 'Structure of the prototype', and 'Appearance of the prototype'. The themes were similar to the focus group themes as the questions were almost identical. However, there were differences in the subthemes. The following section describes the themes and their content.

### 6.3.3 Presentation of the main key themes

#### Theme 1: Perceptions of peer assessment

This theme relates to the teachers' views regarding their individual experience of peer assessment, the value of the activity for first-year students, and its possible use in introductory programming courses. It includes the following subthemes:

- Individual experience of peer assessment.
- The value of the activity on students.
- Using the activity with students.

When discussing teachers' previous experiences of peer assessments, three out of the six participants reported that they had conducted a peer assessment activity as part of their teaching. They described their experiences, and one of the teachers commented that: *"if students engaged it would be success, but the problem was students didn't engage by the time at the end of the module came and they had other deadlines"*. Another participant commented that the first year that contains CS1 does not count in students' final degree in some universities, so many students did not care to conduct such an activity. Teachers who had not used a peer assessment before considered the value of peer assessments with programming students. One teacher expressed some critical issues in regard to the first year of an undergraduate program and the role the peer assessment could play in communication between students. One participant said: *"Social activities could raise persistence in student learning"*. Another teacher mentioned the role of generating feedback to improve understanding of the assignment.

In regard to the use of the Peer Programmer website, half of the teachers ( $n=3$ , 50%) agreed that it should be implemented in introductory programming courses. One participant said: *"I could use it; I think it's useful in terms of building confidence for some students, and I think this system would help"*. On the other hand, one participant who did not agree said: *"I don't think we could anymore, as we've changed the structure of the module, now we teach in block mode, it means that every four weeks, they have another summative assignment, they don't have time, it's not worth"*. As a result, it was clear that

half of teachers had positive perspectives and half of them had negative views. Table 6.7 includes all sub-themes and shows the frequencies of each sub-theme.

Theme	Subcategory theme	Supporting quote	Frequency
<b>Perceptions of peer assessment</b>	Individual experience.	<i>“If students engaged, but the problem was students didn't engage by the time at the end of the module came, and they had other deadlines.”</i>	3
	The value of the activity.	<i>“Social activities could rise persistence in student learning.”</i>	6
	Usage experience.	<i>“I could use it I think it's useful in terms of building confidence for some students, and I think this system would help.”</i>	6

**Table 6.7. Theme 1: Teachers' perceptions of peer assessment**

## **Theme 2: Teachers' concerns about peer assessment**

This theme relates to teachers' fears when applying a peer assessment activity in programming courses as well as any content issues with the prototype. It includes the following subthemes:

- Peer assessment weakness
- The prototype's weakness

Regarding the concerns with peer assessment, results show no negative views of the peer assessment with two of the participants (n=2, 34%). However, one participant mentioned the lack of time to conduct such an activity. Students are usually preoccupied with other modules' assignments which ultimately hinders using the activity: *“they don't have time; it's not worth”*. Another participant mentioned that peer assessment could be biased, so the participant suggested adding some pop-up messages to avoid unconscious bias. The participant said: *“you have to inform students that the assessment they're doing about the code is not about the person”*. Two participants thought this type of activity should not be used in the early weeks of the introductory programming module as it requires knowledge, and students do not have enough knowledge yet.

In regard to perceived weakness of the prototype, half of the teachers (n=3, 50%) did not have any significant issues with the prototype. One participant said: *“I think it's perfectly fine, there's definitely no key issues”*. However, one participant suggested avoiding red colour in the feedback charts. The participant said: *“if you were a student and you got all red colour in your dashboard, I'm sure that wouldn't make you happy, a red colour is usually a negative number”*. Therefore, a different colour might make a student more comfortable. Two participants noted that the matching technique needs more input data such as previous knowledge or preferences to become more accurate when pairing authors with reviewers. Table 6.8 includes all sub-themes and shows the frequencies of each sub-theme.

Theme	Subcategory theme	Supporting quote	Frequency
<b>Teachers' concerns about peer assessment</b>	Peer assessment weakness	<i>“You have to inform students that the assessment they're doing about the code they're not about the person.”</i>	4
	Prototype weakness	<i>“If you were a student and you got all red colour on your dashboard, I'm sure that wouldn't make you happy, red colour is usually a negative number.”</i>	3

**Table 6.8. Theme 2: Teachers' concerns about peer assessment**

### **Theme 3: Structure of the prototype**

This theme was also raised with the students in this study; it involves the composition of all of the components of the Peer Programmer prototype website. It includes:

- Reorganising the tasks of the activity.
- Inclusion to, elimination, or modification of tasks.

Teachers considered the entire setup, the way the information flowed, and the different tasks in the Peer Programmer prototype. Some teachers reorganised some tasks in the peer assessment (n=3, 50%). One participant suggested making a self-assessment after having done a peer assessment: *“The reason students can do a more extensive self-*



*assessment after the actual review stage once they've seen many solutions.*" Moreover, teachers discussed tasks of this prototype to add, eliminate, or develop. For example, when talking about adding more tasks, one participant suggested adding a model answer before assessing peers to help students with accuracy in their peer assessment. The participant said: *"you know they're learning, so you need to give them a model solution to be able to help with the reviewing, but you have to remind the students that this is not the only way that they could do it"*. Concerning the elimination of specific tasks, many teachers (n=4, 66%) did not think anything could be eliminated. However, one participant suggested eliminating the Satisfaction chart that displays the authors' satisfaction with the reviewers' feedback; the participant said: *"It does not really add to students' any significant feedback"*. Regarding developing specific tasks, one participant suggested using a tutorial element that guides students; the participant said: *"So that the first time you use it, you get a little pop-up message in all tasks explains the contents."* This shows that teachers considered various aspects of the prototype structure extensively; they organised activities and suggested adding, editing, and deleting some tasks. Table 6.9 includes all sub-themes and shows the frequencies of each sub-theme.

Theme	Subcategory theme	Supporting quote	Frequency
<b>Structure of the prototype</b>	Reorganising the tasks of the activity.	<i>"Students can do a more extensive self-assessment after the actual review stage once they've seen many solutions."</i>	3
	Inclusion, elimination, or modification of tasks.	<i>"You know they're learning, so you need to give them a model solution...."</i>	15

**Table 6.9. Theme 3: Teachers' view about the structure of prototype**

#### **Theme 4: Appearance of the prototype**

This theme focuses on how the Peer Programmer prototype website appears to users. It includes the following subthemes:

- Prototype design.

- Usability.
- Opinions/attitudes toward figures.

Half of the teachers (n=3, 50%) stated that the feature that caught their attention was the charts. The grade comparison chart (Figure 6.3) and the reviewers' choices table (Figure 6.4) were charts that they liked the best. As for the reviewers' choices table, two teachers suggested that there was no need to display all components of the reviewers' choices table because displaying all parts could confuse students. One participant said: *"a little interface is enough; those who want more details can then find it by clicking on the details link and go through all criteria"*. Furthermore, one participant praised the use of colours in the charts (see, for example, Figure 6.5); the participant said: *"I noticed that colour used well, green for high scores, red for low score, that enhanced and clarified the presentations"*. However, as mentioned before, one of the participants commented that the red colour should not be used in order to avoid frustrating students who have weak scores. Some teachers (n=2, 34%) also praised the interface. One participant said: *"the user interface was really nice, the prototype is effective, and easy to navigate."* Overall, teachers were very interested in the prototype's appearance and agreed that figures were a good way to represent peer feedback as they captured students' attention. Table 6.10 summarises the subthemes and provides example quotations for each subtheme as well as frequencies of occurrence.

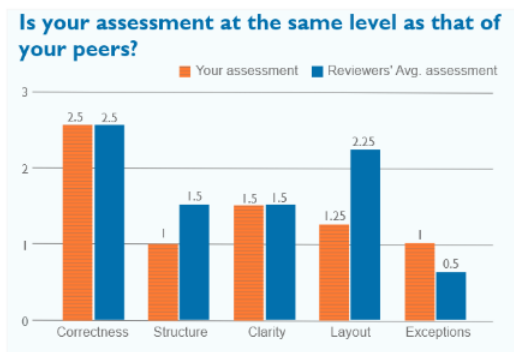


Figure 6.3. Grade details chart

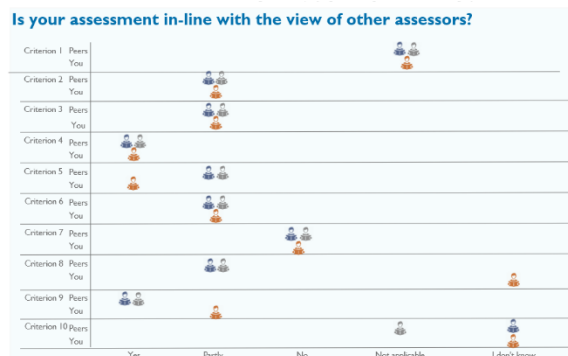


Figure 6.4. Reviewers' choices table



Figure 6.5. Colouring in details grade chart

Theme	Subcategory theme	Supporting quote	Frequency
Appearance of the prototype	Prototype Design.	<i>"I noticed that colour used well, green for high scores, red for low score, that enhanced and clarified the presentations."</i>	3
	Usability.	<i>"The prototype is effective, and easy to navigate."</i>	2
	Opinions toward figures.	<i>"a little interface is enough; those who want more details can then find it by clicking on the details link and going through criteria."</i>	6

Table 6.10. Theme 4: Teachers' view about appearance of the prototype

As a result, the teachers interviewed for this study voiced their attitudes towards the Peer Programmer prototype website. Some teachers raised their concerns about using peer assessments with first-year students; they stated the lack of time, lack of knowledge, and possibility of bias. These concerns may prevent some of them to use peer assessment in introductory programming courses. The teachers also talked about the structure of the prototype and its appearance. They discussed features that caught their attention, and they responded positively to the idea of visualising peer feedback. This finding is supported by the findings from the student focus groups, as visualisation was the feature that caught their attention most too. The next section discusses visualisation and its impact on students' learning process in an attempt to encourage other researchers to investigate the impact of visualisation in peer assessment.

## 6.4 Visualising peer feedback

Feedback in peer assessment is valuable data that can be measured, tracked, analysed, and visualised. The integration of visualisation and learning has a positive impact on the motivation to learn; it supports awareness and self-reflection, and increases critical thinking as well as communication and collaboration among students (Shatri and Buza, 2017). In this study, the goal of using visualisation in peer assessment data was to provide timely, understandable, and abstracted feedback that helps learners (both author and reviewer) deepening awareness of progress and self-reflection in their work, then motivate them to learn further. The students who participated in this study interacted with various visual charts that represented their peer assessment performance, and they helped to edit these charts to improve their appearance, and to be easy to understand. Visualising feedback was a powerful tool that caught students' attention, and helped them understand the peer feedback data. It can be assumed that visualisation of feedback data in peer assessment is a technique that can benefit learning analytics.

Learning analytics is defined as “the measurement, collection, analysis and reporting of data about learners and their contexts, for purposes of understanding and optimizing learning and the environments in which it occurs.” (Chatti *et al.*, 2012, p. 2). Learning analytics can make significant contributions and improvements in various areas, in accordance with Mullan, Sclater and Peasgood (2016), including in the following ways: 1) As a tool for quality improvement or quality assurance of teaching staff or learning institutions through enhance teachers own practice, or as a diagnostic tool for the systemic level. 2) As a tool to raise students' retention rates through identifying at-risk learners and engaging early intervention with support and advice. 3) As a tool for acting upon and assessing differential learning outcomes through the learner population; for example, monitoring low-participation learners and engaging in early intervention before assessment results are made available. And 4) as an enabler tool for adaptive learning and personalising the learning materials based on learners' previous knowledge or learner interactions. Thus, it is worth gaining insight into the potential impact of visual analytics in peer assessment data on students' performance. The concept of visual analytics is to shift

the information overload into an opportunity. The aim of visual analytics is to make the method of processing data transparent for an analytic discourse (Keim *et al.*, 2008).

Several forms of visualisation represented student data in the peer assessment prototype. This study selected charts following Kirk (2012), who summarised types of charts and appropriate usage of each type. For example, the bar chart was used to compare student levels in a specific assignment to other students in the class for the same assignment. Moreover, a visual table displayed choices in the marking scheme of all reviewers who assessed a specific assignment. Additionally, a line chart displayed a student's progress in each assignment (see section 6.5.3, Table 6.28 for a description of each chart). The aim of the diverse range of chart types that relate to the same topic and to the same assignment is to allow students to understand feedback in multiple ways. A study conducted by Ueki and Ohnishi (2016) suggested this approach; the scholars selected two types of charts to represent students' efforts in peer assessment, as Figure 6.6 shows. They found that the visualisation peer assessment data enabled users to become aware of some aspects and improve their performance. However, more data on the impact of visualisation in peer assessment are needed.

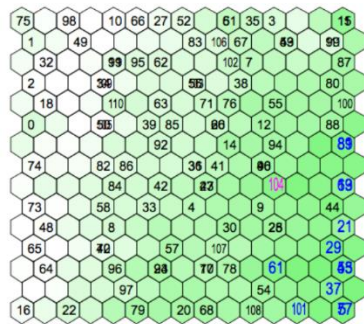


Figure 5: System for visualizing self- and peer-assessment data by SOM.

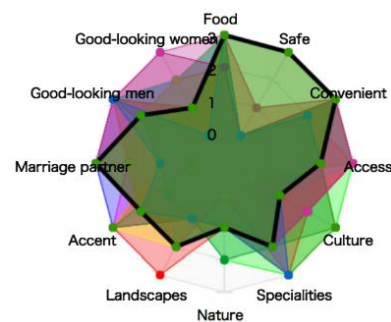


Figure 6: System for visualizing self- and peer-assessment data by a radar chart.

### Figure 6.6. Visualising peer assessment data (Ueki & Ohnishi, 2016)

There are vast sources of formative assessment data that are not used efficiently in learning analytics, and yet they can be invaluable to teachers, administrators, students, and parents. Divjak, Grabar and Maretic (2016) claim that assessment data in learning analytics environments are underdeveloped compared with other types of data analytics. Vieira, Parsons, and Byrd (2018) produced a systematic literature review of visual learning

analytics. They found that only a small number of studies have been conducted to create visual learning analytics tools into classroom settings. Moreover, although data visualisation is an effective tool of awareness and self-reflection for learners, in visual learning analytics, teachers were the main audience for the visualisation rather than students (Suzuki, 2016; Vieira, Parsons and Byrd, 2018). Teachers use these techniques to understand the use of learning resources and collaboration among participants. Of course, understanding students' situations enables instructors to inform suitable pedagogical actions, but there are also many opportunities for learners to take advantage of visualisation. Visualisations for the purpose of students displaying their own learning process may, for instance, lead to the development of cognitive and metacognitive skills; it could even improve students' retention in programming courses. Therefore, this study recommends that researchers and practitioners attempt to visualise peer assessment data to support learning analytics tools and evaluate the impact of visual tools on students' awareness, motivation to learn, and engagement in peer assessment. As these qualitative data in this study demonstrated the effectiveness of visualisation to represent peer feedback, more quantitative data are needed to measure the effect of visualisation on students' performance.

The following section presents the final form of the Peer Programmer prototype website.

## **6.5 Final design of the prototype**

The Peer Programmer prototype is the final design of the prototype; it contains a series of three phases: create, review, and feedback. In the create phase, students log into the website and are presented with a given assignment. Their teacher creates this assignment, and students can choose many forms to solve the assignment - a written response to the question, or uploading the code file. Students are then asked to assess themselves using the marking scheme prepared by their teachers. The teacher can upload the assignment and manage the setting of the assignment; for example, determine the timeframe during which students have access to the assignment. When the create phase closes, students begin the review phase. Students must assess numerous peers' work during the review phase and provide them with an evaluation and some critical

feedback. The assessment occurs with the guidance of the marking scheme prepared by the teacher. This assessment can be anonymous, and personalised based on students' needs via pairing between authors and reviewers. As soon as the review phase closes, students can log back into the Peer Programmer prototype and immediately receive their peers' assessment feedback (as an author), and students can see the feedback of other reviewers who assessed the same work (as a reviewer). The teacher can give students the opportunity to resubmit their work based on the feedback, and the teacher finally decides the official grade of this assignment. The prototype website is an interactive website that allows users to navigate from page to page and use functionalities such as interacting with figures, and rating feedback. The prototype has been modified several times according to students' and teachers' feedback gathered as part of this study. It was finalised after the second phase of the research.

This section summarises the requirements students and teachers suggested a peer assessment activity should fulfil. This can help designers and developers of peer assessment systems because all functional requirements and necessary diagrams were presented to explain the activities of the prototype. Furthermore, the journey of students and teachers are presented in the following sections as case studies.

### ***6.5.1 Requirement analysis***

A requirement analysis is the process of defining users' expectations for a proposed prototype. It helps to identify best-fit functions and reduces implementation risks. In this study, requirements were identified by analysing a variety of data; teacher and student questionnaires, focus groups, and teacher interviews. Therefore, the system requirements were divided into the functionality required by teachers, the functionality required by students, and the system functional requirements.

#### **Teacher requirements**

The teachers did not ask for specific requirements when the interviewer interviewed them but did specify conditions, e.g., teachers asked to be responsible for creating the marking scheme criteria, the assessment is formative, and they demanded that they do not want

it to cause extra effort. The results of the questionnaire and interviews set out the following functional requirements:

1. Teachers can manage assignments
  - a. The system shall allow the teachers to add new assignments that the students can solve.
  - b. The system shall allow the teachers to create marking scheme criteria.
  - c. The system shall allow the teachers to add instructions and control the setting of the assignment.
2. Teachers can assess assignments
  - a. The system shall allow the teachers to inspect and grade the students' assignments.
3. Teachers can display student's report to monitor progress
  - a. The system shall display the students' progress in self-assessment and peer assessment.
  - b. The system should summarise the progress of the assessment process.

### **Student requirements**

The results of the questionnaire and focus group discussions set out the functional requirements of the students:

1. Students can complete the assignment
  - a. The system shall allow students to solve the assignment by writing a code or uploading a file.
  - b. The system shall allow students to complete the self-assessment using marking criteria.
  - c. The system shall display instructions to support students.
2. Students can complete the peer assessment



- a. The system shall allow students to assess a set of anonymous peer work using established marking criteria.
  - b. The system shall display instructions to support students.
3. Students can display their feedback to monitor their progress
  - a. The system shall allow students to display interactive visual feedback as an author from different aspects.
  - b. The system shall allow students to display interactive visual feedback as a reviewer from different aspects.
  - c. The system shall allow students to display interactive visual progress.
4. Students can rate peers' work
  - a. The system shall allow students to rate all reviewers' work.
  - b. The system shall display the nicknames of excellent students in assessing their peers on a dashboard.

### **System requirements**

1. The system informs students of assignment
  - a. The system shall notify students of each new assignment.
  - b. The system shall display a marking scheme to students.
2. The system selects appropriate reviewers
  - a. The system shall calculate the self-assessment and the peer assessment scores, then find the average.
  - b. Based on their score, the system shall categorise students into two groups: proficient and non-proficient.
  - c. The system assigns a set of reviewers for each author based on proficiency level.
3. The system can display feedback from two sides
  - a. The system shall display the feedback as an author.
  - b. The system shall display the feedback as a reviewer.

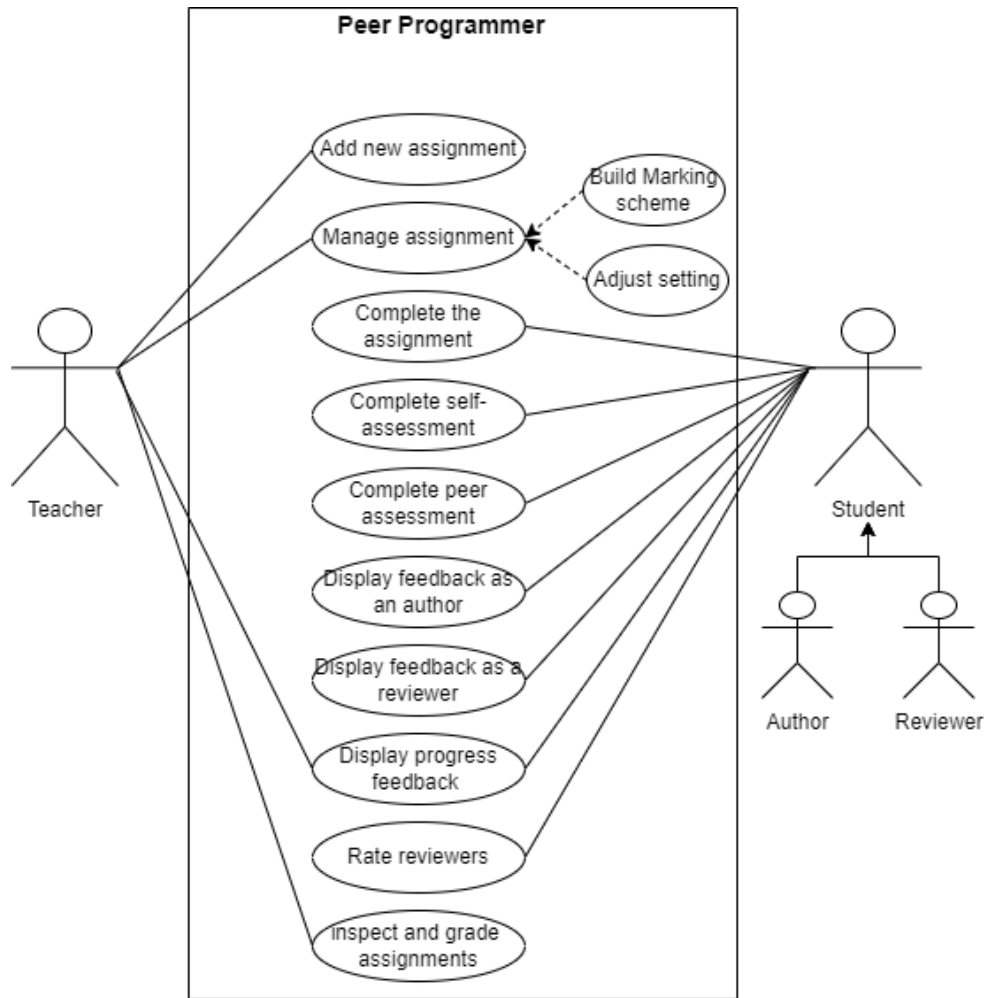
- c. The system shall display the progress feedback.
4. The system can store and access the database.
- a. The system shall store students' profiles in the database.
  - b. The system shall store assignments of courses.

### **6.5.2 Prototype diagrams**

It is valuable to visualise the functional requirements of the prototype by using diagrams. Diagrams help identify any internal or external factors that may influence the prototype which should be considered at early stages. Diagrams can also be easily translated into design choices and development priorities. This section outlines a plan for how objects inside a prototype interact. This section uses diagrams created with Unified Modelling Language (UML), a graphical language representing system components, system domains, and the relationships between elements and concepts. It provides a standard way to visualise a system's design (Booch, Rumbaugh and Jacobson, 2005). Two types of diagrams are discussed in this chapter, use case diagrams and sequence diagrams. Use case diagrams show various use cases and the system's different types of users, while sequence diagrams show object interactions arranged in a time sequence. These diagrams can help designers of peer assessment systems to understand the high-level functions and scope of a system.

#### **Use case diagram**

A use case diagram offers a comprehensive view of the entire prototype in a single diagram (Fowler, 2003). In this prototype, two actors interact: teachers and students. Each actor has an initial set of use cases. The teacher has four tasks: the teacher can add new assignments, control the assignment setting, inspect and grade, and display students' reports if needed. On the other hand, the student has five tasks: the student can complete the assignment, complete the self-assessment, complete the peer assessment, rate the assessment of peers, and display feedback. Figure 6.7 shows the case diagram of this Peer Programmer prototype.



**Figure 6.7. Use case diagram of the Peer Programmer prototype**

At the start of the peer assessment process, the website shows two types of users: teachers and students. The teacher adds a new assignment and manages the assignment by building marking scheme criteria and adjusting the assignment settings.

The students (acting as authors) then complete the work and a self-assessment using the marking scheme. After that, they submit their work and self-assessment. Shifting from the 'author' role to the 'reviewer' role, students are shown a list of anonymous solutions for the assignment they have just completed. Using the same marking scheme, reviewers assess a set of peers' works. The system then displays the feedback without the teacher's intervention. The student receives three different types of feedback based on the role they occupied, i.e., author or reviewer, and then they get overall feedback for progress over

time. Returning to the author role, the author can rate the reviewer's feedback. Finally, the teacher can display the progress and inspect and grade the assignment.

### **Sequence diagrams**

A sequence diagram is an interaction diagram that shows how functions are carried out and arranged in a time sequence (Fowler, 2003). Sequence diagrams are usually related to use case realisations in the logical view of the system that will be developed. A sequence diagram helps the designer manage the project settings to discover logic, interface, and architecture problems early in the design process. Figure 6.8, Figure 6.9, Figure 6.10 and Figure 6.11 show samples of the sequence diagrams used to express the behaviour of the prototype.

### ***Adding and managing the assignment by teachers***

A teacher is responsible for adding the assignment and building criteria to support the students when assessing themselves and their peers. The teacher can also specify the setting of the task; for example, the number of assessors required, the anonymity of the assessment, and additional instructions. Figure 6.8 shows the behaviour of the prototype when the teacher wishes to add and manage the assignment. Firstly, the teacher selects the course to which the assignment belongs. The system responds by retrieving the relevant page. The teacher then adds a new assignment to the selected course. Next, the teacher builds a marking scheme by adding criteria. After that, the teacher adjusts the assignment setting via the setting page. Finally, this assignment is added to the database and displayed to students.

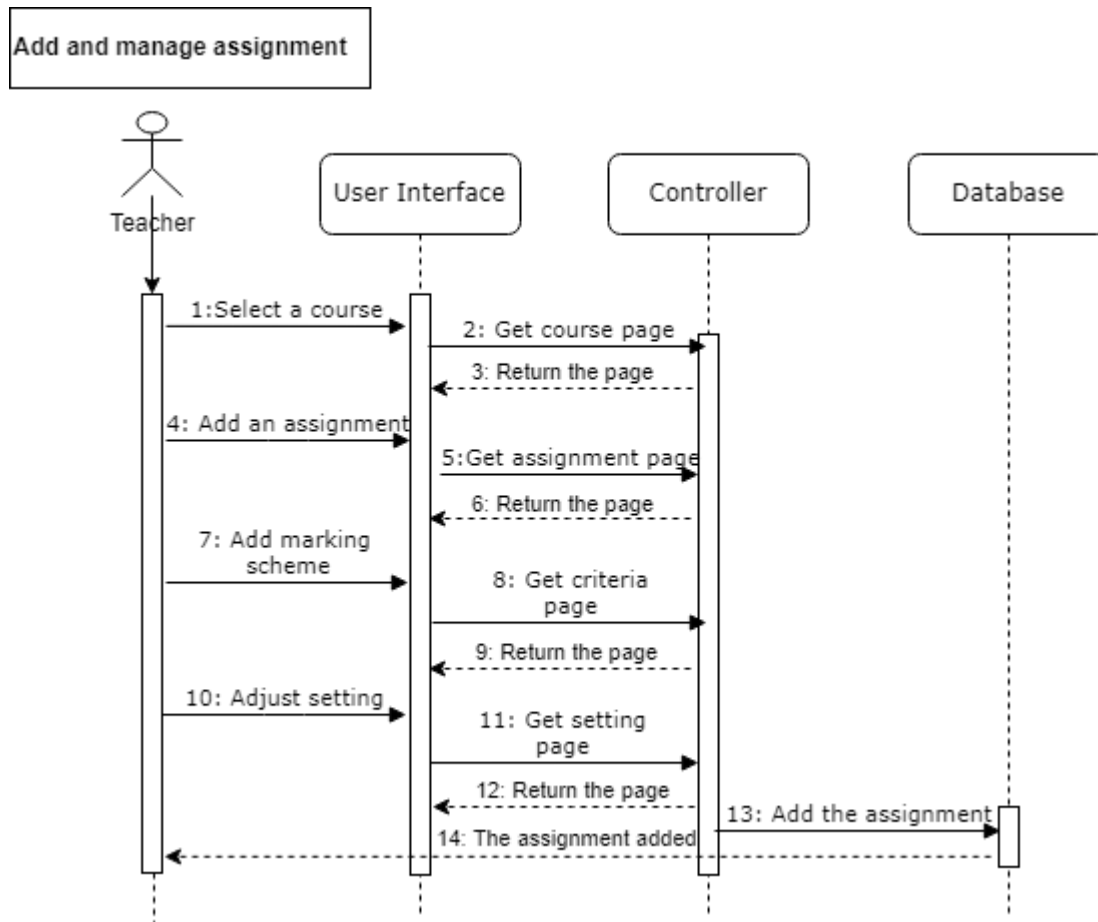


Figure 6.8. Sequence diagram to add and manage the assignment

### **Create the task phase**

For students, the prototype website consists of three phases: the create phase, the review phase, and the feedback phase. The sequences of the create phase can be seen in Figure 6.9. In the create phase, the student first selects the assignment, the system then responds by displaying the assignment question page. Then, the student uploads the solution to a given assignment and completes a self-assessment. Next, the system calculates the student's self-assessment score and categorises the student as either proficient or non-proficient. After that, the student's answer, self-assessment score and proficiency level must be saved in the database.

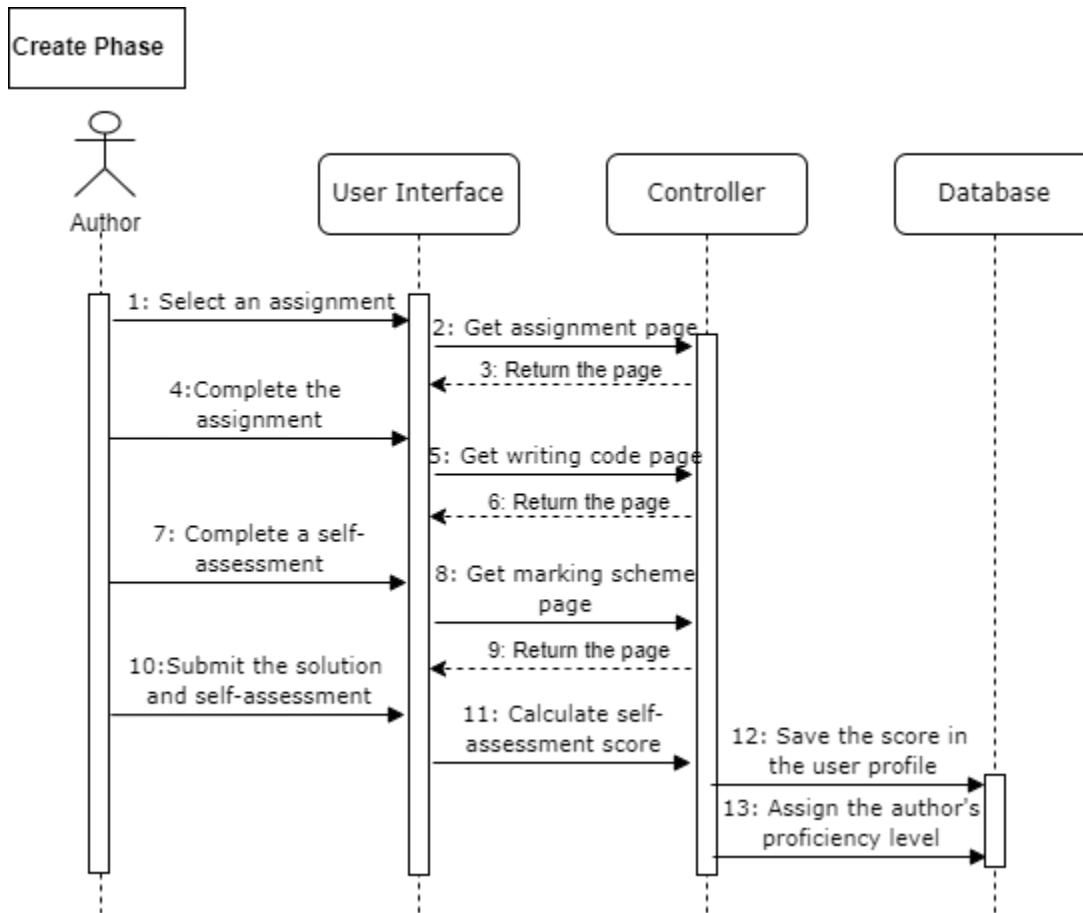


Figure 6.9. Sequence diagram of the create phase

### ***Review the task phase***

The second phase is the 'review phase', in which the students conduct three formative peer assessments using the marking scheme. When students move to the review phase, the system displays a list of anonymous peers' works. The students, as reviewers, then assess their peers' works using a marking scheme. The system then calculates the peer assessment scores and updates the user's profile and proficiency level saved in the database. Figure 6.10 shows the behaviour of the prototype when a student wishes to assess peers.

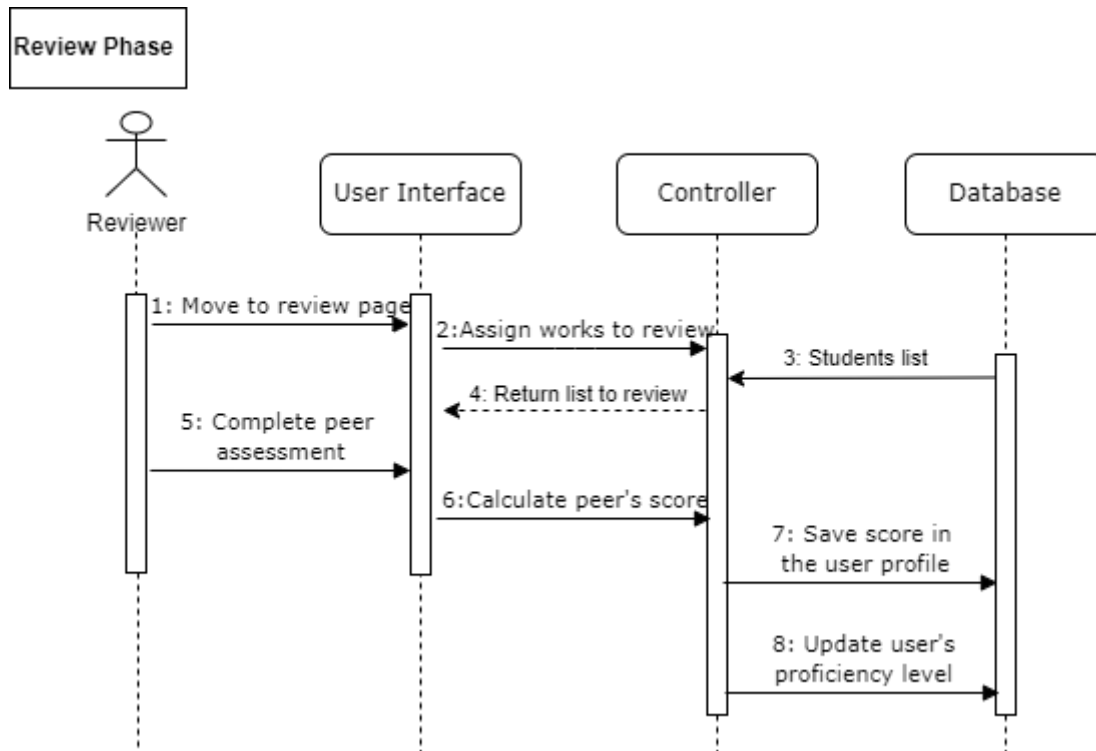
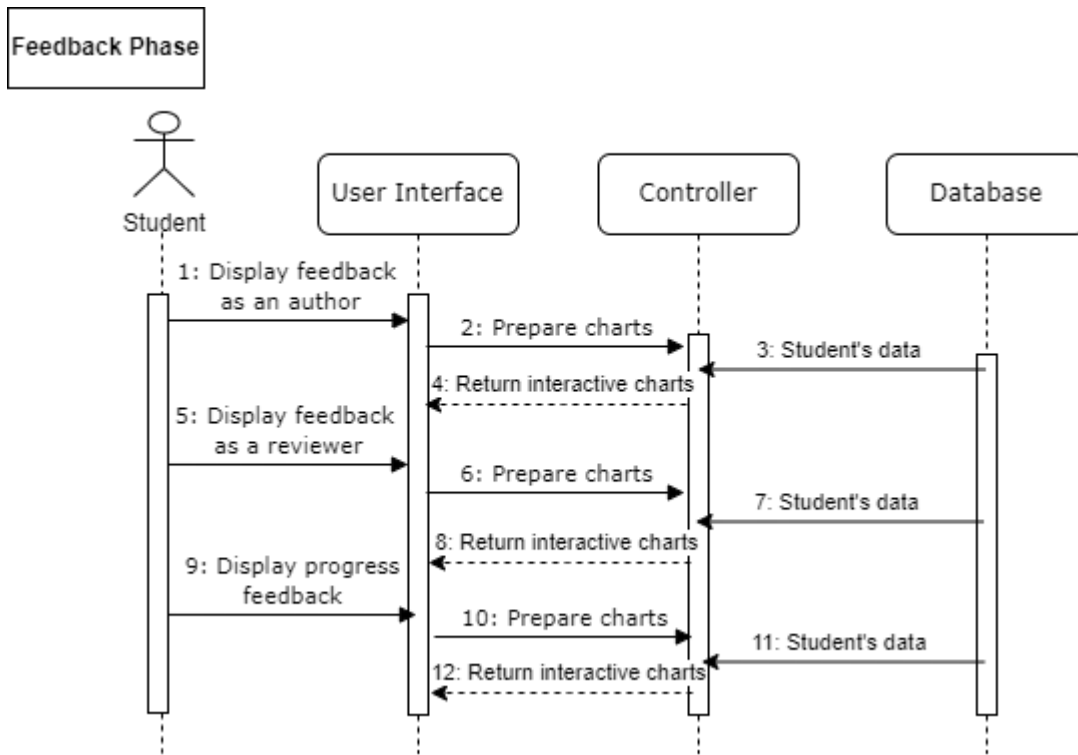


Figure 6.10. Sequence diagram of the review phase

### ***Feedback phase***

The third phase is the 'feedback phase'. The feedback appears from three viewpoints: as an author, as a reviewer, and in progress over time. All data come from the user profile that was saved in the database. The system prepares charts in three tabs. As an 'Author', the user can see the assessment of their work from all the reviewers. As a 'Reviewer', the user can compare their review grade to that of other reviewers. And in the 'Progress' tab, the user can see their progress over the span of time during which peer assessment has been implemented. When the student moves to the 'feedback page' and clicks on the 'Author' tab, the system retrieves student data from the database and displays a number of charts in the feedback page. The same happens when students click on the 'Reviewer' tab or the 'Progress' tab. Figure 6.11 shows the behaviour of the prototype when a student wishes to see their feedback.



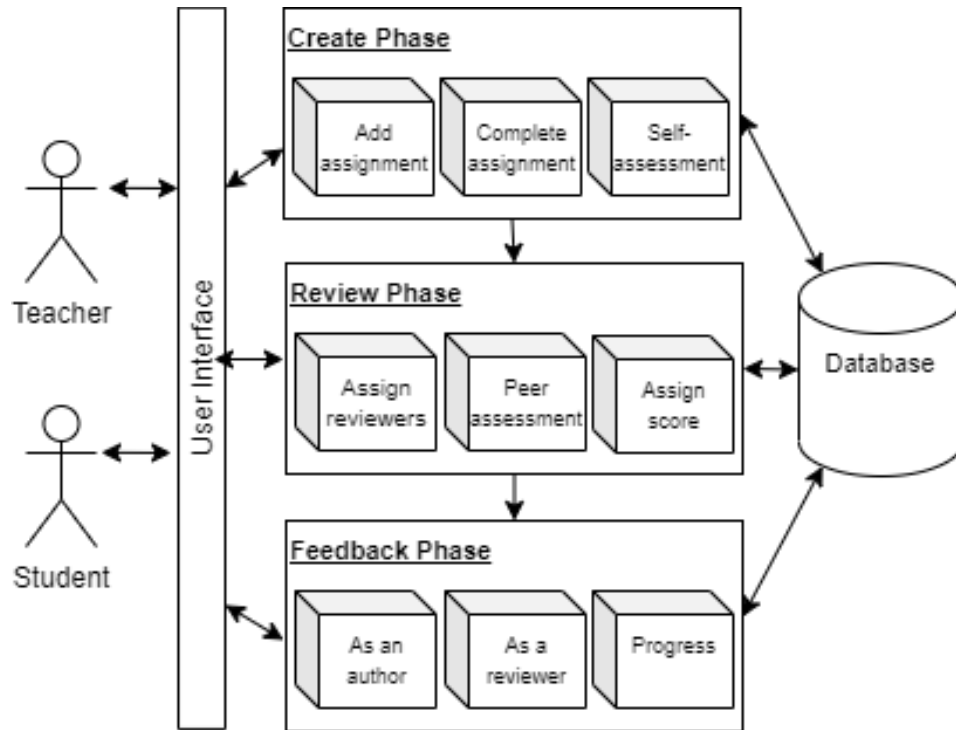
**Figure 6.11. Sequence diagram of the feedback phase**

The previous diagrams represented the most essential processes in the peer assessment and the arrangement of events in the order of their occurrence. The following section explains the description of the prototype in general.

### **Prototype description**

The general structure of this Peer Programmer prototype is shown in Figure 6.12:





**Figure 6.12. General description of Peer Programmer prototype**

As seen in Figure 6.12, the main components of the prototype are:

a. User Interface:

The user interface allows the user to interact with the website. The prototype website has two types of users: teachers and students. Therefore, two interfaces were created for the users.

b. Create Phase:

The create phase allows a teacher to add a new assignment and manage this assignment. For the students, it allows them to complete the assignment and the self-assessment.

c. Review Phase:

The review phase allows a teacher to assign a score to all of the students' work, and it enables a student to assess the peers assigned by the system.

d. Feedback phase:

The feedback phase allows the teacher to monitor the students' progress and to leave comments on peers' work. It also enables the student to view their feedback from the peer assessment process from three viewpoints: as an author, as a reviewer, and their progress over period of time.

e. Database:

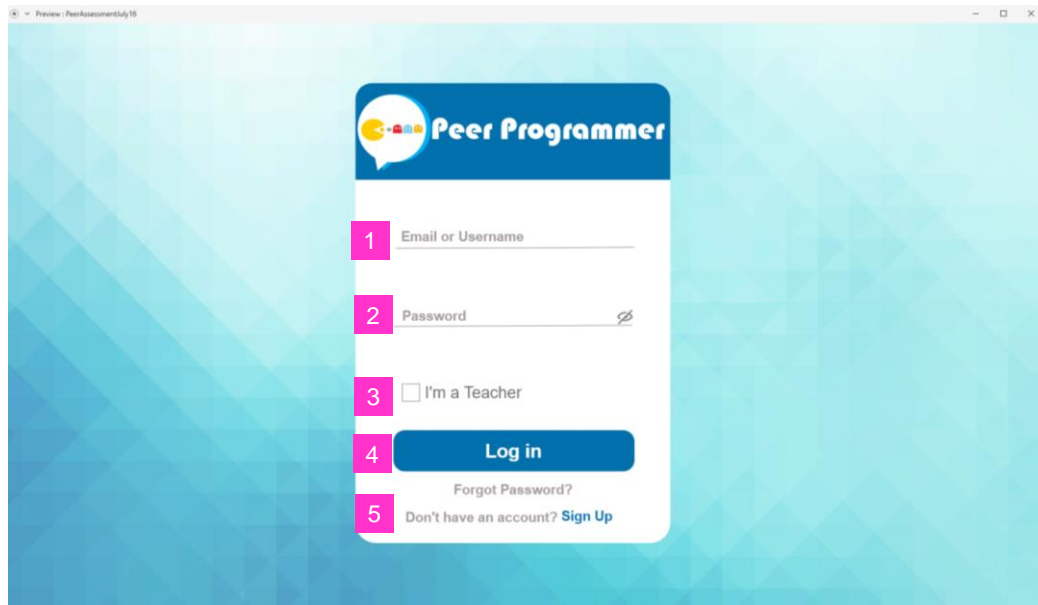
The database is a repository of user profiles and all the data pertaining to the website.

### ***6.5.3 Design of the Peer Programmer prototype***

The prototype design was illustrated with a use scenario. Scenarios are ideal for explaining how users can explore the prototype. The usage of the prototype is described from two perspectives in the following sections: one for the student and one for the teacher. Screen captures are displayed to document the user activities with a description of the page contents represented in a table under each screen capture.

#### **Case study: The story of the teacher's journey**

The login page is the initial page of the website. From there, the user can enter identifying information into the website in order to access it (Figure 6.13). The teacher can enter his/her email address and password and click on "I am a teacher", then the "Log in" button. Table 6.11 describes the content of the page.

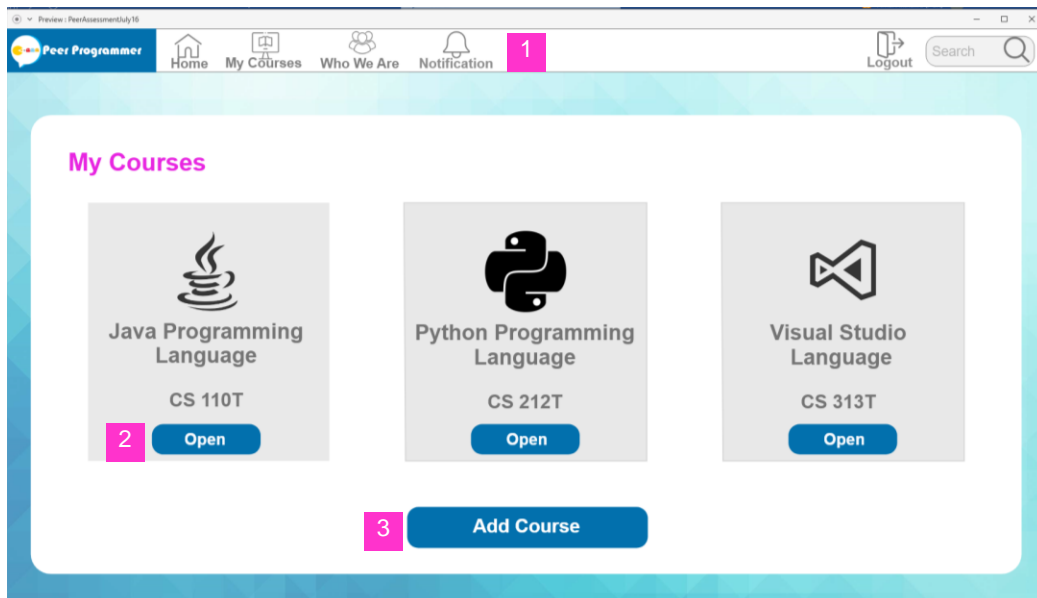


**Figure 6.13. Login page**

No	Description
1	This textbox allows a user to enter an email or username
2	This textbox allows a user to enter a password
3	This checkbox asks a user if he/she is a teacher
4	This button allows a user to login
5	This button allows a user to sign up

**Table 6.11. Description of the login page**

Next, the teacher logs onto the website. The website redirects the teacher to the Courses page. The teacher can select a course to which he/she will add a new assignment (see Figure 6.14, Table 6.12).



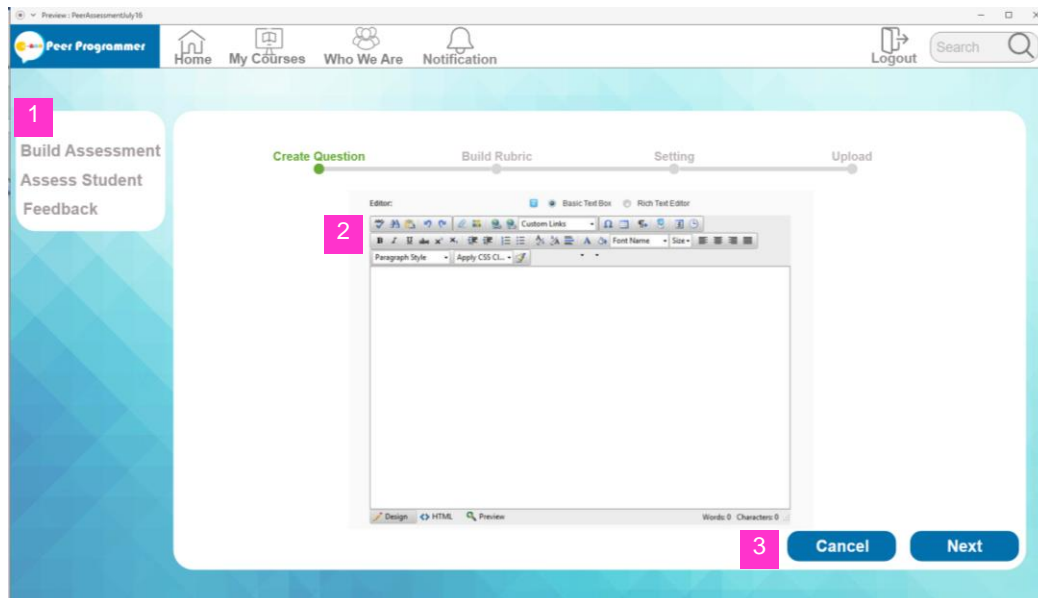
**Figure 6.14. Courses page in the teacher's account**

No	Description
1	This toolbar contains buttons that execute commands rapidly.
2	This button allows a user to select a course to add an assignment to.
3	This button allows a user to add a new course.

**Table 6.12. Description of courses page**

### **Create phase**

A teacher should not use a peer assessment at every assignment. The teacher can use a peer assessment activity if there is more than one solution to a problem, or if the task may be difficult for some students. In this phase, the teacher can build a new assignment by writing the assignment question in a rich text editor box (see Figure 6.15, Table 6.13). The teacher then writes the question and clicks on 'Next' to get the 'added successfully' pop-up message, and he/she can move on to the next phase.

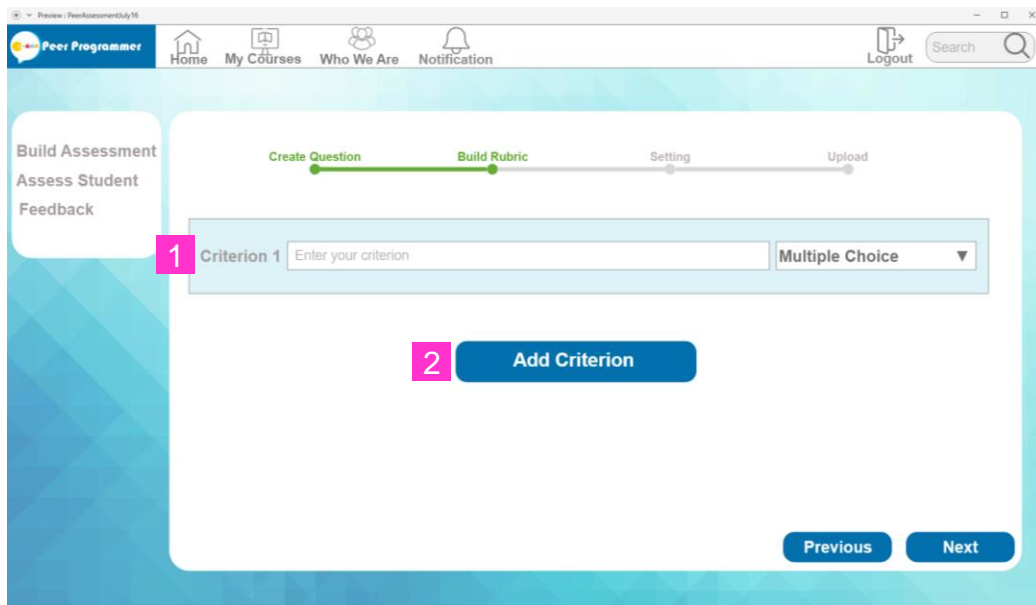


**Figure 6.15. Create question page**

No	Description
1	This navigation sidebar contains links that use responsive navigation.
2	This rich text editor allows the user to write the assignment question.
3	These buttons allow a user to move to the previous or next pages.

**Table 6.13. Description of create question page**

After that, the teacher should add the criteria of the marking scheme by clicking 'Add Criterion'. The teacher then writes the criterion and selects what type of criterion it is (multiple choice, true/false, scale, open-ended, etc.). After adding all the criteria, the teacher clicks on 'Next' to move on to the next phase (see Figure 6.16, Table 6.14).

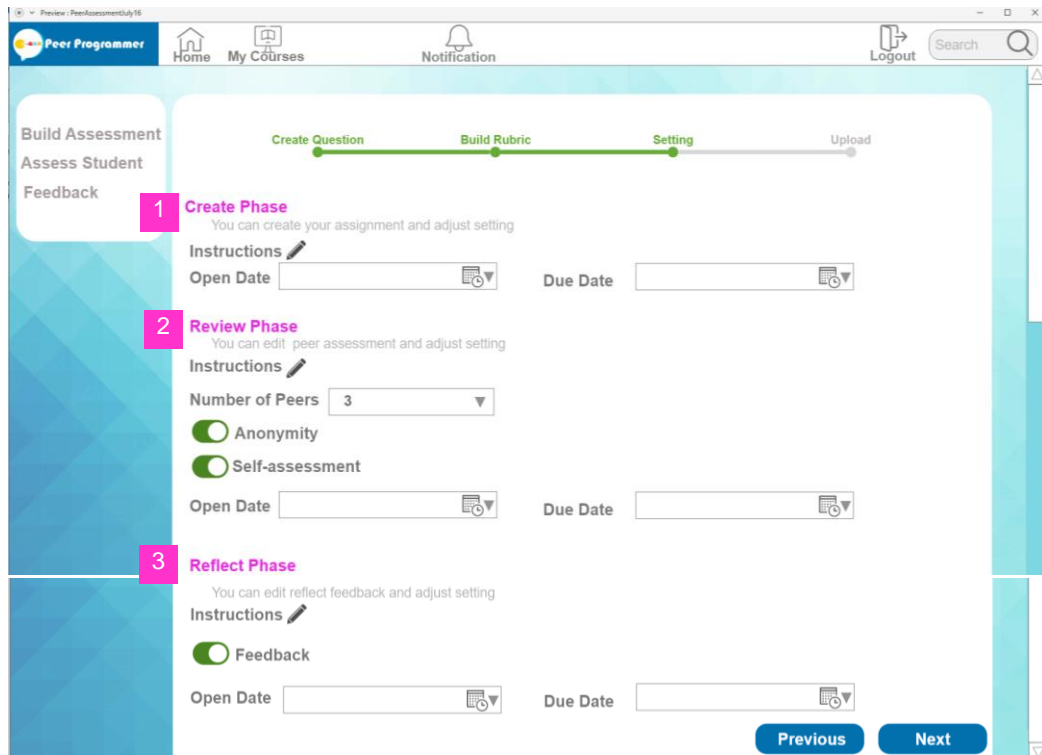


**Figure 6.16. Build a marking scheme page**

No	Description
1	This textbox allows a user to write a criterion and select the type of criterion.
2	This button allows a user to add a new criterion.

**Table 6.14. Description of build a marking scheme page**

Next, the teacher can adjust the settings of the assignment. For example, the teacher can write the instructions and determine the assignment's open date and due date in create phase. In review phase, the teacher decides the number of peers who will assess each task, turns on/off the anonymity feature so that the author's and the reviewer's identity will be unknown/known, and turns on/off the self-assessment feature – 'on' means that the authors must assess themselves before assessing any peer. In addition, the teacher can turn on/off the rating feature in the reflection phase, deciding whether or not the author can rate the assessment of each reviewer (see Figure 6.17, Table 6.15). It is worth noting that, students go through these phases sequentially. For instance, students cannot review their peers' work until they submit their assignments to avoid cheating or plagiarism.

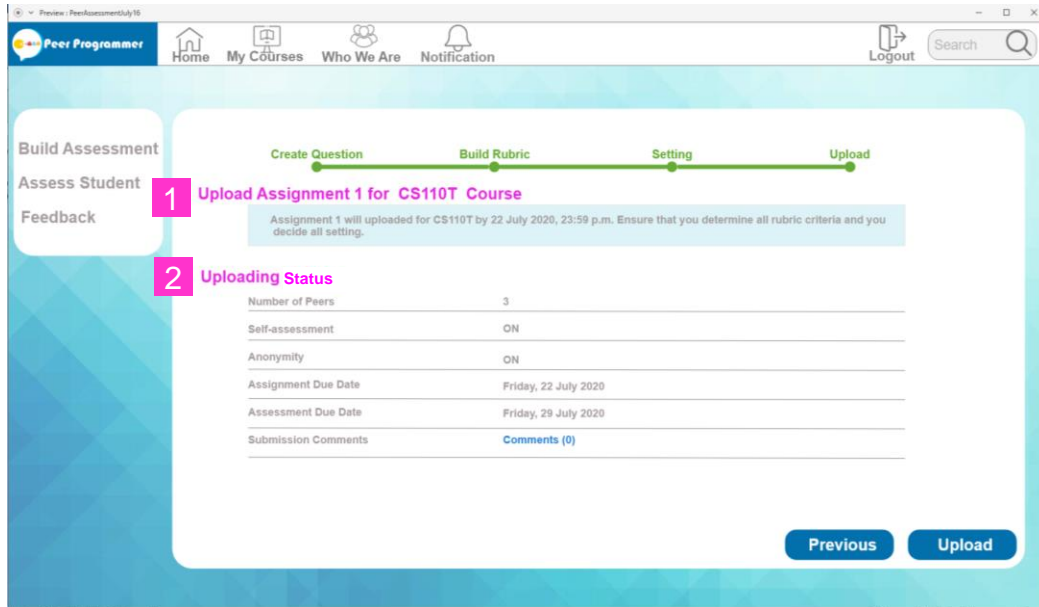


**Figure 6.17. Assignment settings page**

No	Description
1	This part allows the user to write instructions and determine the open and due dates for the assignment.
2	This part allows the user to select the number of reviewers, turn on/off the anonymity and self-assessment features, and determine the open and due dates for reviewing the assignment.
3	This part allows the user to turn on/off the rating reviewers and determine the open and due dates of rating the assessment.

**Table 6.15. Description of assignment settings page**

Finally, the teacher uploads the assignment for the selected course. This page displays a summary of the status of the uploaded assignment (see Figure 6.18, Table 6.16).



**Figure 6.18. Summary of uploading assignment page**

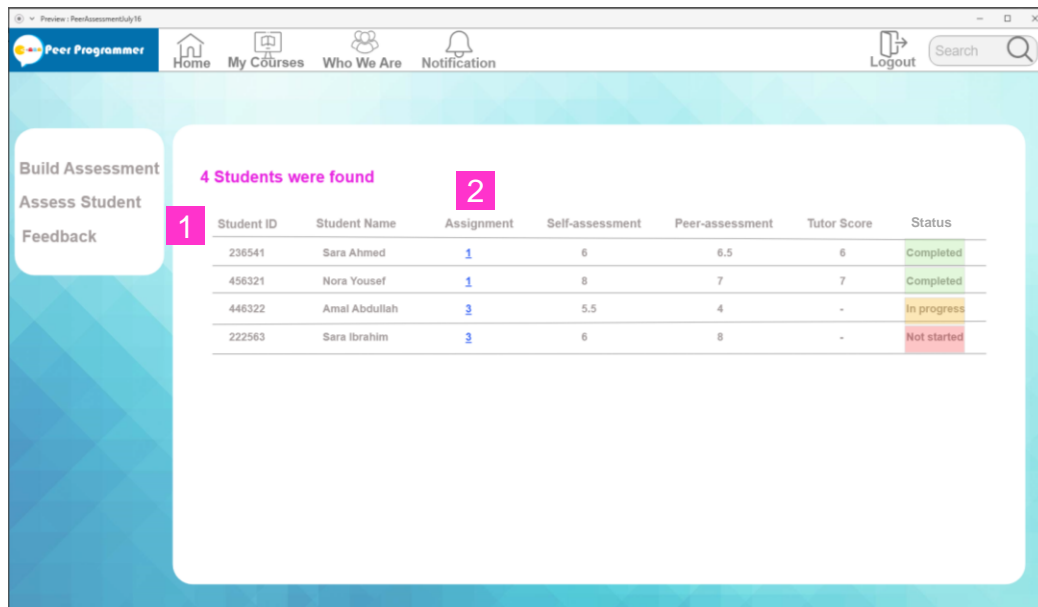
No	Description
1	This section shows the due date of the uploaded assignment.
2	This table summarizes the selected features of the uploaded assignment.

**Table 6.16. Description of uploading assessment page**

### ***Review phase***

The teacher then moves on to the next phase, which is reviewing the students' assignments. Figure 6.19 contains a table with many rows that represent the students' records. The teacher can select any record to assess the student's assignment, and they are responsible for assigning scores to the assignments. The teacher can also see the students' assessment status (see Figure 6.19, Table 6.17).





**Figure 6.19. The assessment page in the teacher account**

No	Description
1	This table displays students' records and their status for each assignment.
2	This link displays the students' work so that it can be assessed by the teacher.

**Table 6.17. Description of the assessment page in the teacher account**

### ***Feedback phase***

The teacher can then move on to the next phase, the feedback phase. The page on which the feedback is displayed is a visual page, and it contains a number of charts that summarise the results of the selected assignment (see Figure 6.20, Table 6.18).

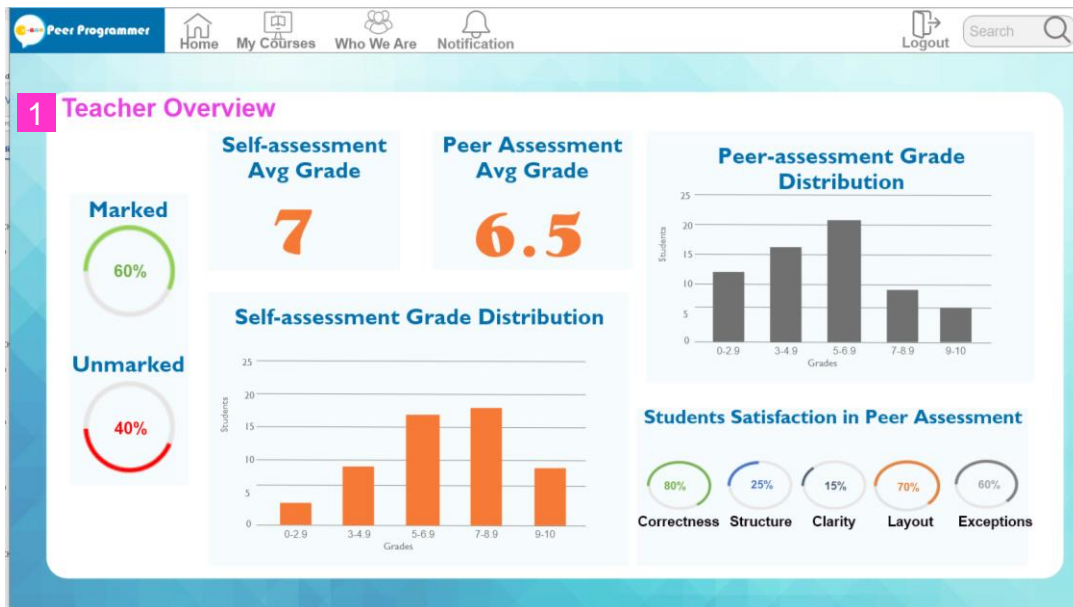


Figure 6.20. Teacher's feedback page

No	Description
1	These charts summarize the results of the selected assignment: the self-assessment, peer assessment, satisfaction with the peer assessment process, and unmarked and marked assignments.

Table 6.18. Description of teacher's feedback page

This case study showed all the tasks that teachers can do in the prototype. They can add a new assignment, build the criteria of the marking scheme, and adjust the assignment criteria. Teachers are also able to assess their students and display feedback. The following section discusses the Peer Programmer prototype from a student's point of view.

### Case study: The story of the student's journey

To begin, the students log into the system via the Login page. They then move on to the 'My Courses' page. It contains a toolbar at the top which includes the following related links: 'Home', 'My Courses', 'Who we are', 'Notification', and 'Logout'. First, the students can select the course for which they will complete a new assignment (see Figure 6.21, Table 6.19).

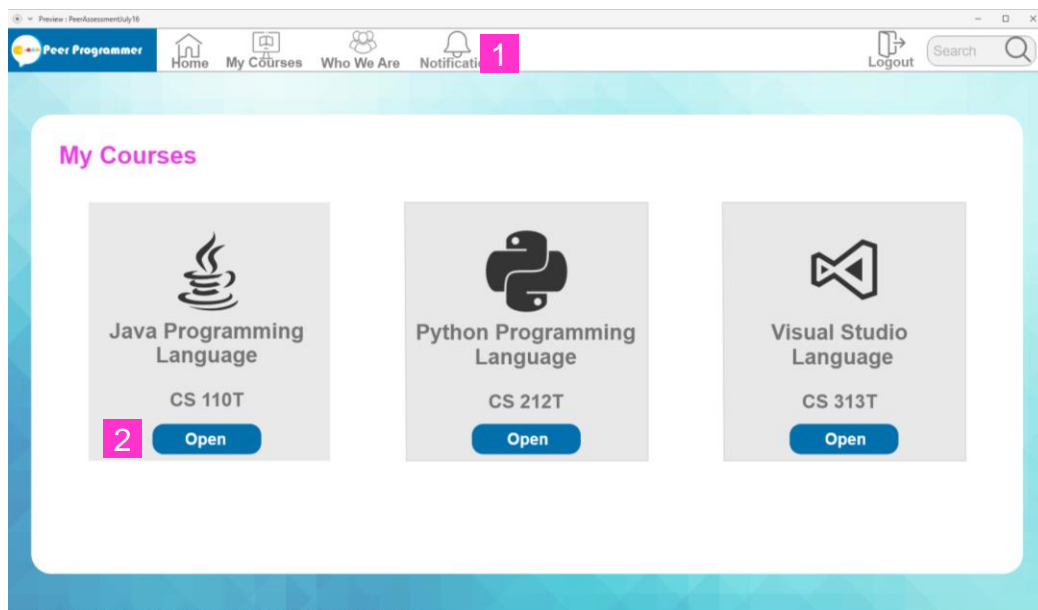
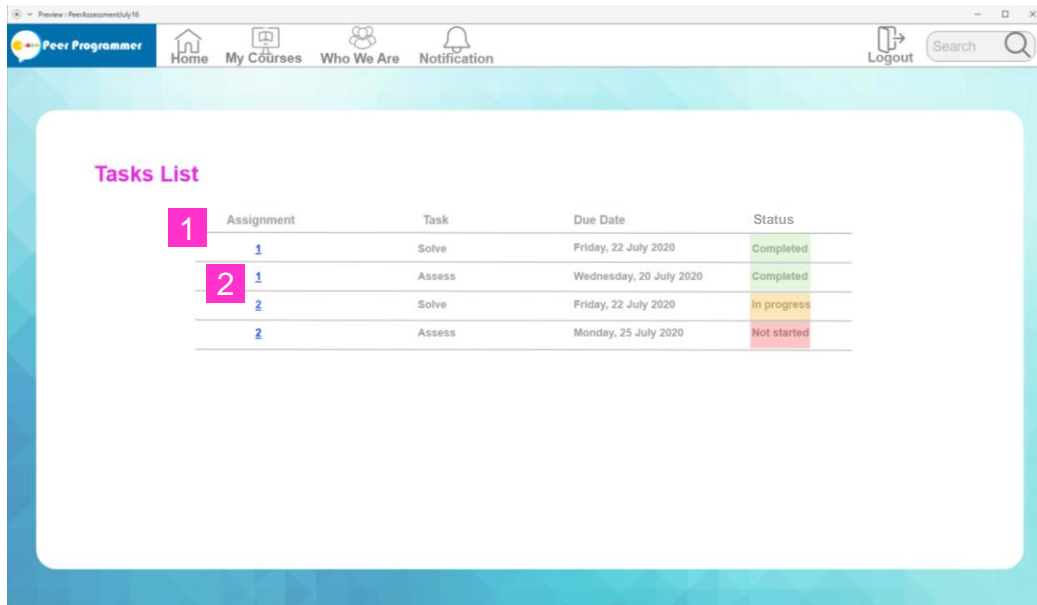


Figure 6.21. Courses page in student's account

No	Description
1	This toolbar contains buttons that execute commands rapidly.
2	This button allows a user to select a course in which to complete an assignment.

Table 6.19. Description of courses page in student's account

The students then move to the tasks list in the selected course, which contains a list of the tasks they must perform. Figure 6.22 contains a table with several rows, with each row representing the status of the task, the assignment number, and the due date. The students can select an assignment to complete next (see Figure 6.22, Table 6.20).



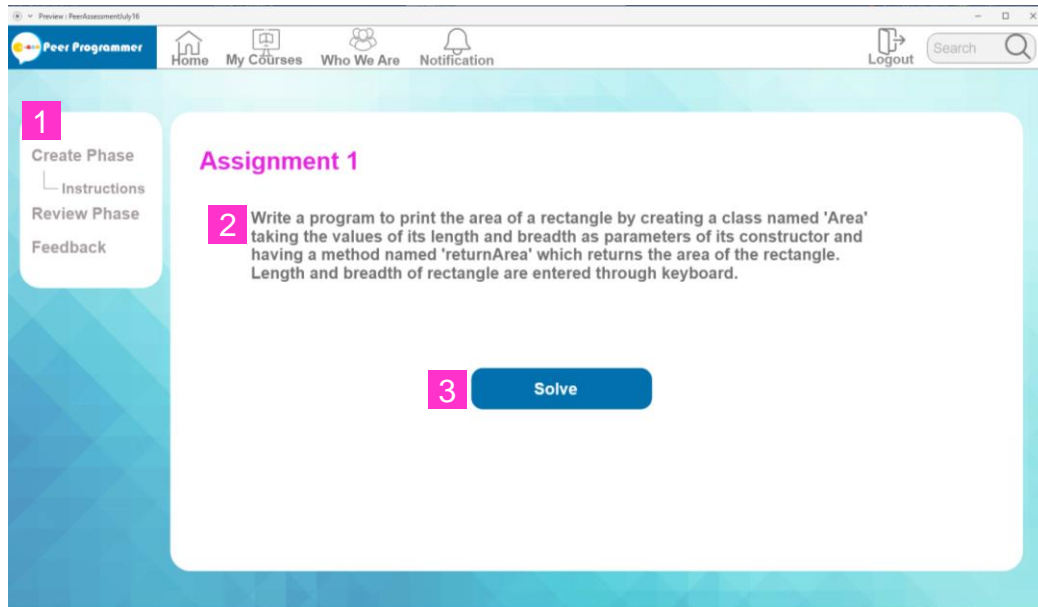
**Figure 6.22. Tasks list page**

No	Description
1	This table contains a list of assignments in the selected course.
2	This link moves the student to the selected assignment.

**Table 6.20. Description of the tasks list page**

### **Create phase**

This phase concerns completing the selected task by writing/uploading the code with the self-assessment. The system informs the students that the new assignment is available. The students (acting as authors) then read the assignment and complete it by clicking on the “Solve” button (see Figure 6.23, Table 6.21).

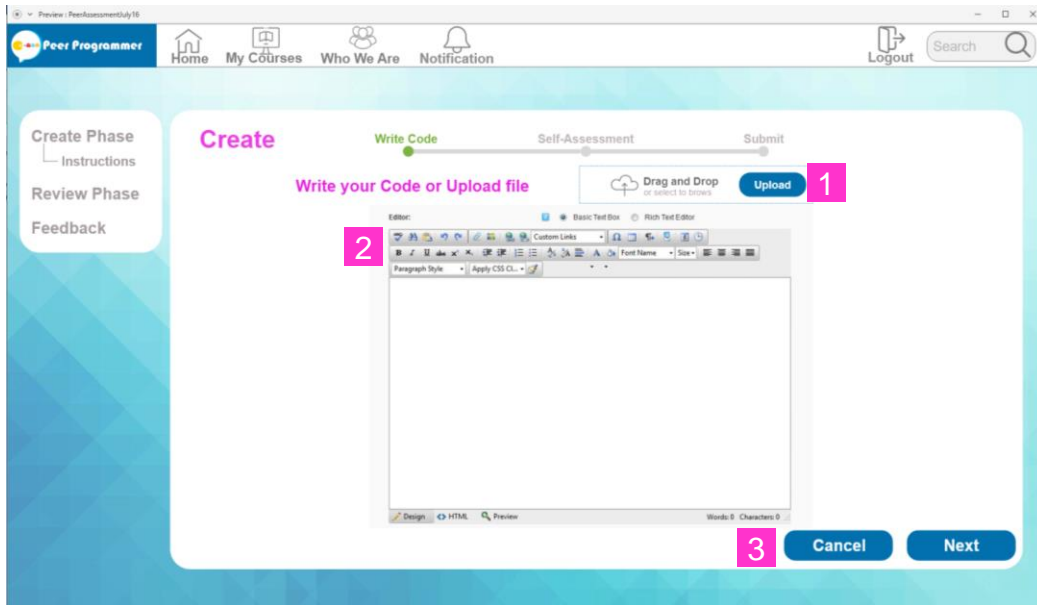


**Figure 6.23. Assignment question page**

No	Description
1	This navigation sidebar contains instructions to clarify the create phase.
2	This textbox contains an assignment question.
3	This button allows a user to solve the assignment.

**Table 6.21. Description of the assignment question page**

The 'Solve' button allows the authors to complete the assignment by writing a code or by uploading a file that contains the solution (see Figure 6.24, Table 6.22).



**Figure 6.24. Writing code page**

No	Description
1	This button allows users to upload a file.
2	This text editor allows users to write the answer to the assignment.
3	These buttons allow a user to move into the previous or next page.

**Table 6.22. Description of writing code page**

The authors then assess themselves by using the marking scheme the teacher has built. The marking scheme displayed on this figure is the same as that presented in section 3.4.3, Table 3.4. It includes ten criteria classified into five categories: correctness of the code, structure, clarity, layout of the program, and exceptions status, with five Likert scales: yes, partly, no, not applicable, and I don't know. Additionally, there are three open-ended questions with free-text boxes where the authors can write their opinions of the pros and cons of the code and how it could be improved (see Figure 6.25, Table 6.23).

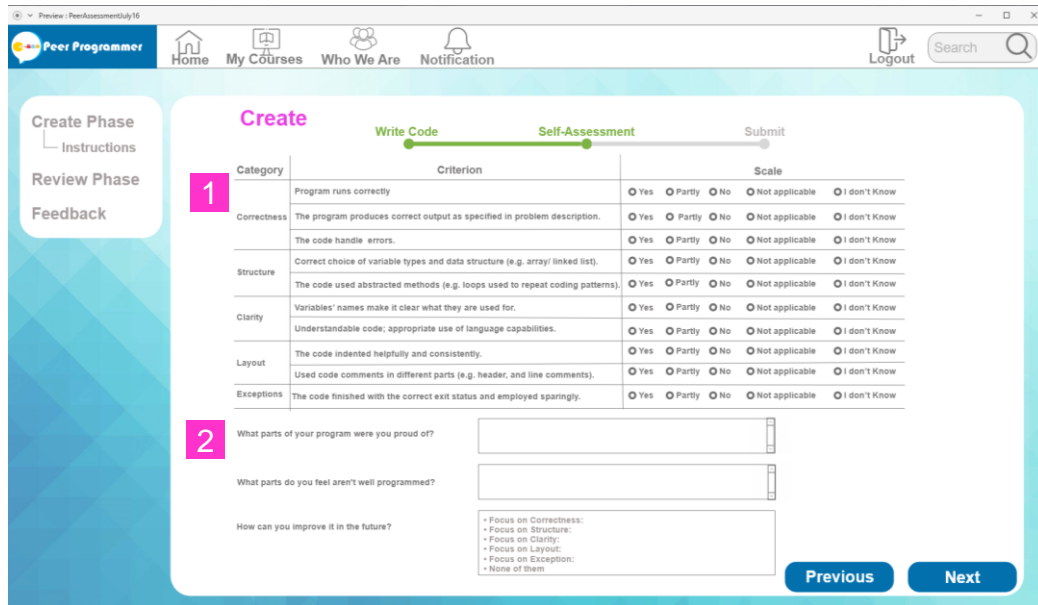
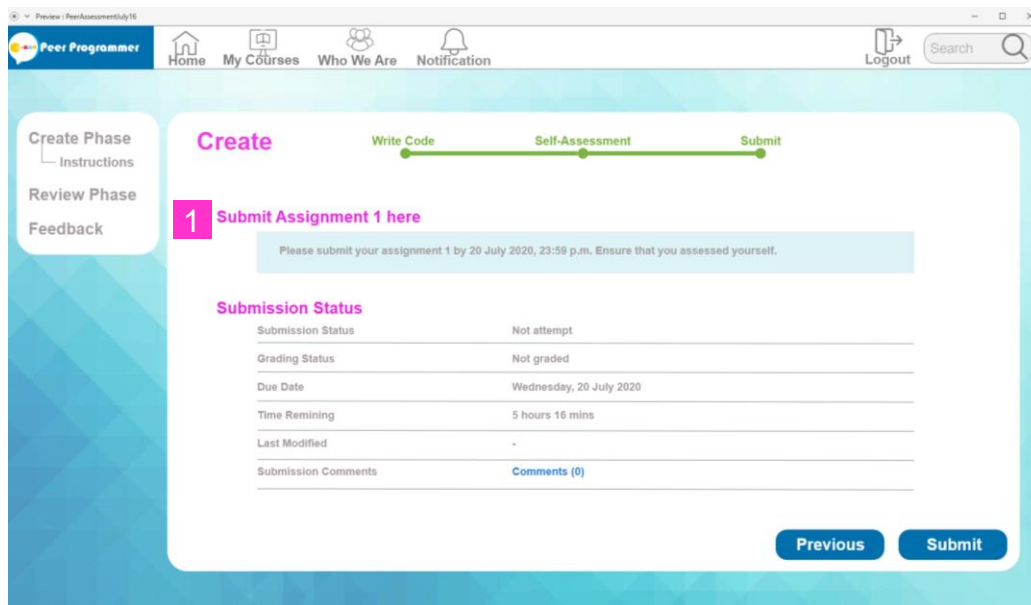


Figure 6.25. Self-assessment page

No	Description
1	This marking scheme allows users to select a suitable scale for each criterion.
2	These text boxes allow a user to write free-text comments.

Table 6.23: Description of self-assessment page

After the authors submit their work and self-assessment, the system moves to the submit page that summarises the status of the submission (see Figure 6.26, Table 6.24).



**Figure 6.26. Submission page in create phase**

No	Description
1	This section summarizes the status of the assignment that the author has solved.

**Table 6.24. Description of the submission page in create phase**

The website then matches the authors to a set of reviewers (each author has multiple reviewers). The system matches the authors and reviewers based on how difficult the authors perceived the assignment to be. The authors who submit their work specify by self-assessment how difficult the assignment was, and the system then matches the selected author to a group of reviewers who may meet the author’s needs in the selected assignment. So the matching process is not random. The following pop-up message did not appear on the original website; it was created for the focus group discussions to describe the process of assigning the reviewers (see Figure 6.27). This is the end of the create phase and the beginning of the review phase.



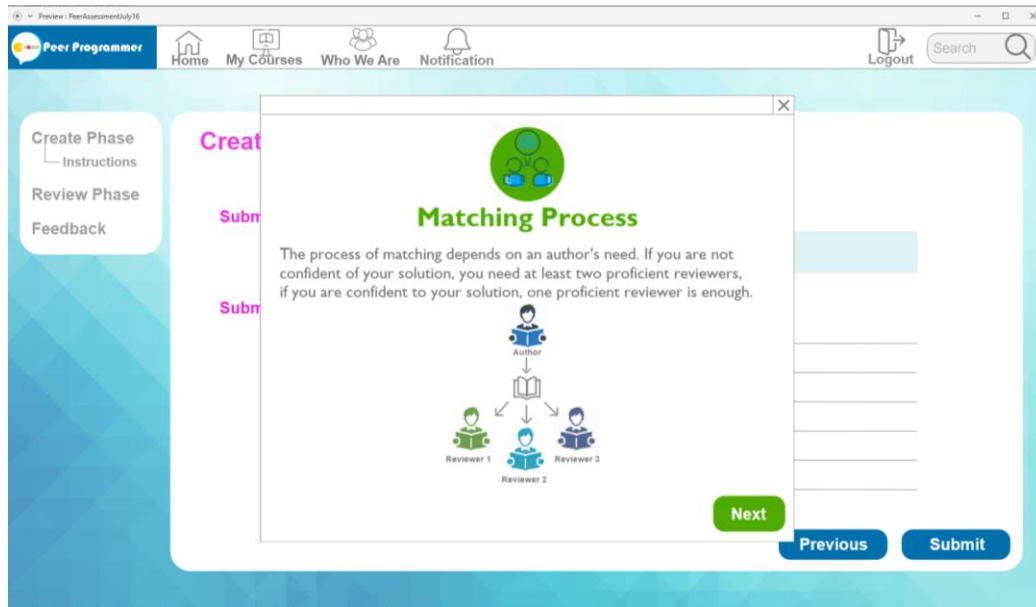


Figure 6.27. Matching process pop-up message

### ***Review phase***

In this phase, the students shift from the 'author' role to the 'reviewer' role by assessing the assignment using the marking scheme. The students are shown a list of anonymous solutions for the assignment that they have just completed, and the reviewer assesses a set of solutions. The reviewer can show the code, highlight any part of it, add comments, and zoom in or out of any part of the solution. Since students in the peer assessment may be exposed to the possibility of cheating and plagiarism, worth noting that this prototype does not allow reviewers to assess their peers' work until they submit their work as it appeared in Figure 6.26. Also, if the reviewer needs further instructions, he/she can select 'Instructions' in the navigation sidebar (see Figure 6.28, Table 6.25).

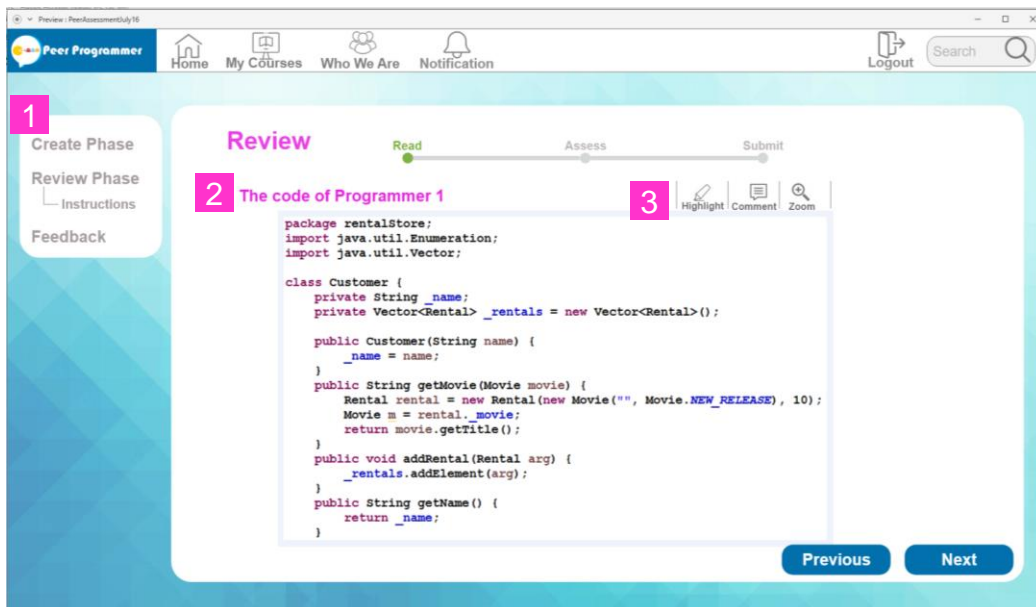


Figure 6.28. The peer's work page

No	Description
1	This navigation sidebar contains instructions that explain the review phase.
2	This section displays an anonymized code.
3	These buttons allow a user to highlight, zoom in/out, and add comments on the code.

Table 6.25. Description of the peer's work page

After that, using the same marking scheme that was used in the self-assessment, the reviewer assesses his/her peers' work by selecting the appropriate scale and filling in the open-ended questions (see Figure 6.29, Table 6.26).

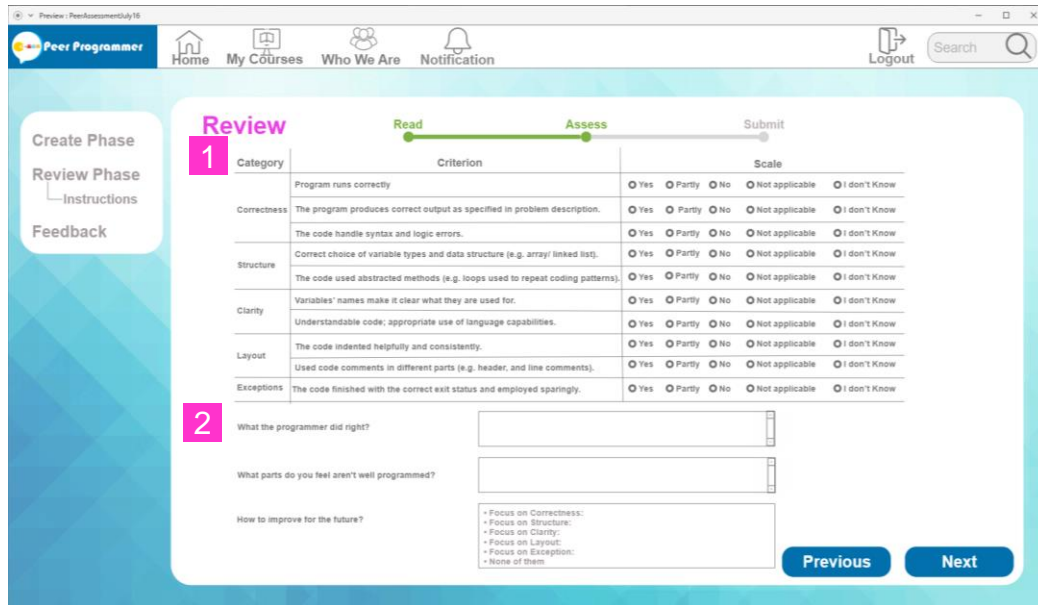
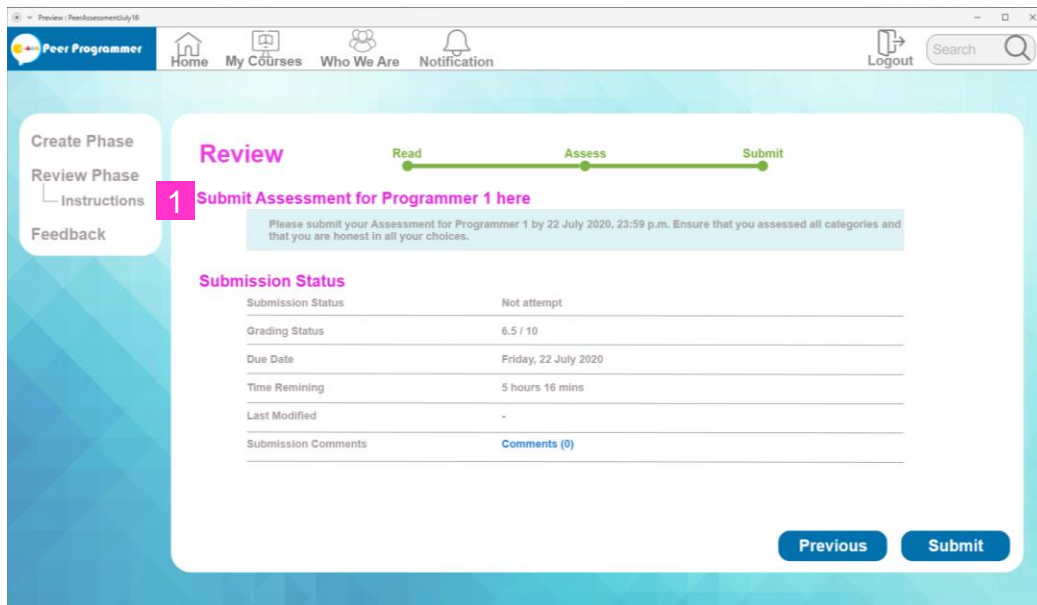


Figure 6.29. Peer assessment page

No	Description
1	This marking scheme allows a user to select a suitable scale for each criterion.
2	These text boxes allow a user to free write comments about peers' answers.

Table 6.26. Description of peer assessment page

Finally, the students can submit the assessment as a reviewer. The system then moves on to the submit page, which summarises the status of the submission (see Figure 6.30, Table 6.27).



**Figure 6.30. Submission page in review phase**

No	Description
1	This section summarizes the status of the assignment that reviewers have assessed.

**Table 6.27. Description of the submission page in review phase**

At the end of this stage, students have acted as authors in the create phase and as reviewers in the review phase. The next phase is the feedback phase. Feedback is important because it provides insight and explains what the students have done in the peer assessment.

### ***Feedback phase***

The prototype displays the feedback without the teacher’s intervention. The feedback appears from three viewpoints: feedback for the student when he/she was acting as an author, feedback for the student when he/she was acting as a reviewer, and feedback of peer assessment process in progress over time—each viewpoint displayed in a specific tab. An example of reviewer feedback is shown in Figure 6.31 and in Table 6.28.

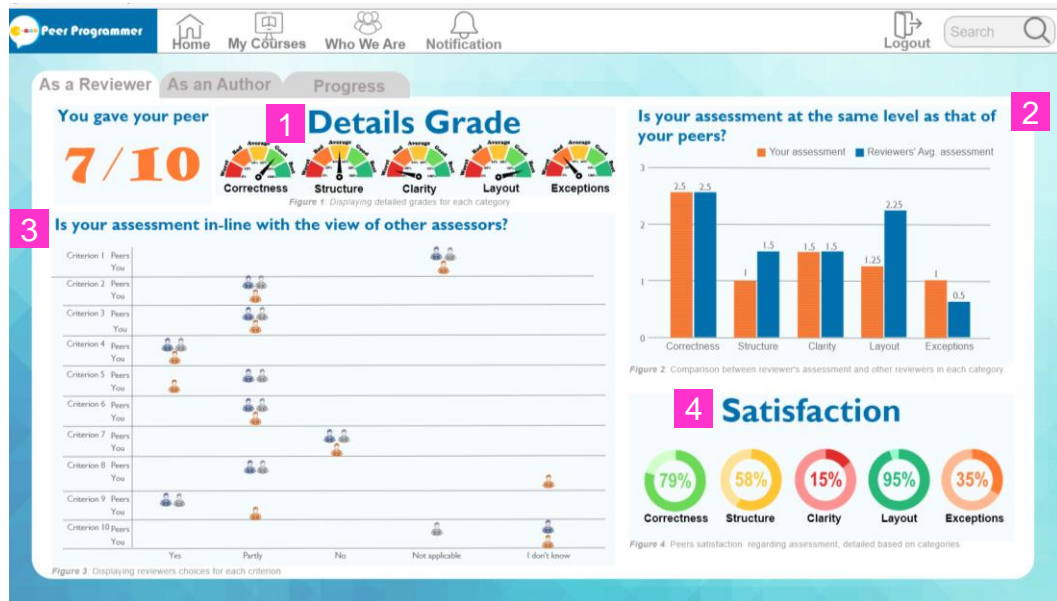


Figure 6.31. Feedback page

No	Description
1	<b>Grade Chart:</b> This displays the total grade the reviewer gave to the author, as well as the author's detailed grades. As there are five categories in the marking scheme, the reviewers can show the exact grades they gave to their peers per category; these data are displayed as a meter chart for each category. The system itself calculates grades, so the reviewer only fills in the marking scheme without affecting the authors' actual grades, because it is a formative assessment.
2	<b>Grade Comparison Chart:</b> This illustrates the comparison between one reviewer's feedback and the average of other reviewers' feedback for the same assignment in each category. It is represented as a bar chart for each category.
3	<b>Reviewers' Choices Table:</b> This table shows all the reviewers' choices for each criterion in the marking scheme, using a character to represent each reviewer. Every reviewer can thus compare his/her choices on a Likert scale with those of the other reviewers.
4	<b>Satisfaction Chart:</b> This figure is a percentage chart, displaying the authors' satisfaction with the reviewers' feedback in each category.

Table 6.28. Description of the Feedback page

All the figures are interactive content to help users visually understand the fundamental concepts (e.g., see Figure 6.32).

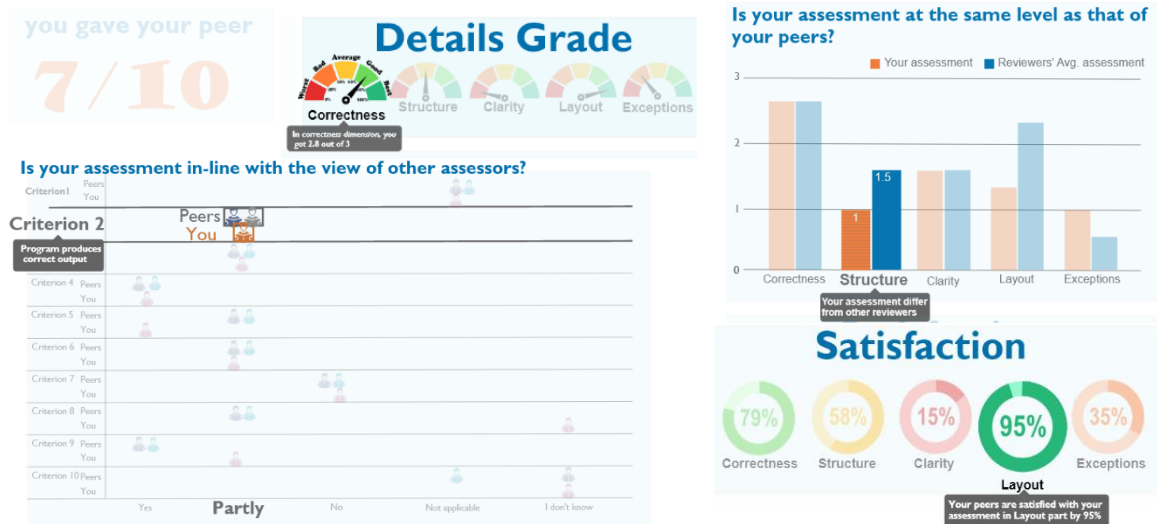


Figure 6.32. Interactive charts

The students will find the same figures as the reviewer in the Author tab, which also displays the reviewers' feedback. The last tab displays the author's progress across the peer assessment implementation period (see Figure 6.33, Table 6.29). This chart is interactive; it allows users to grow or shrink the level of detail represented in a chart by clicking on categories at the top of the chart.

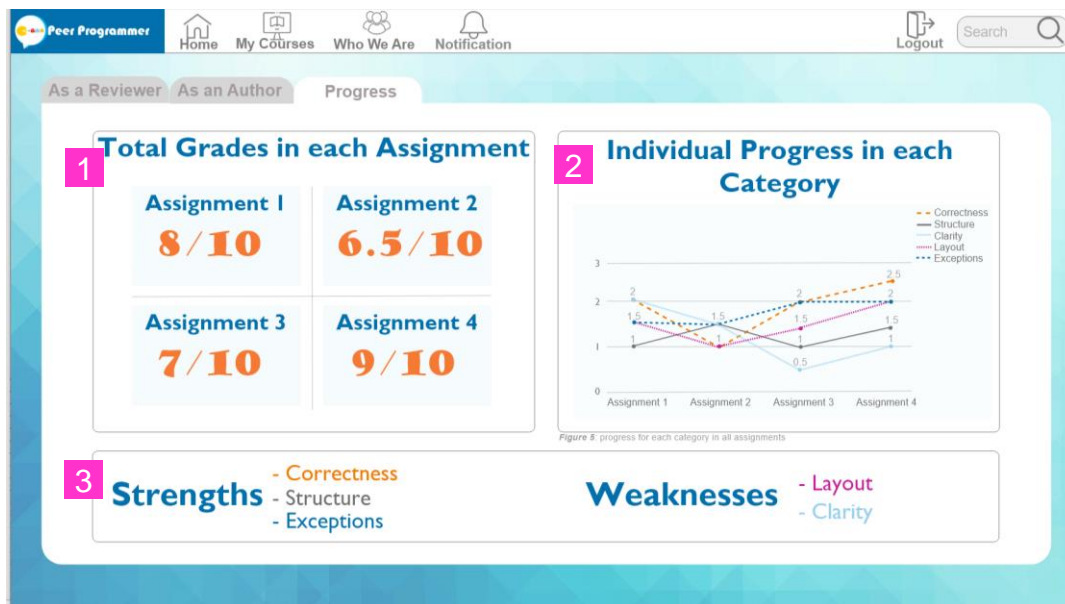


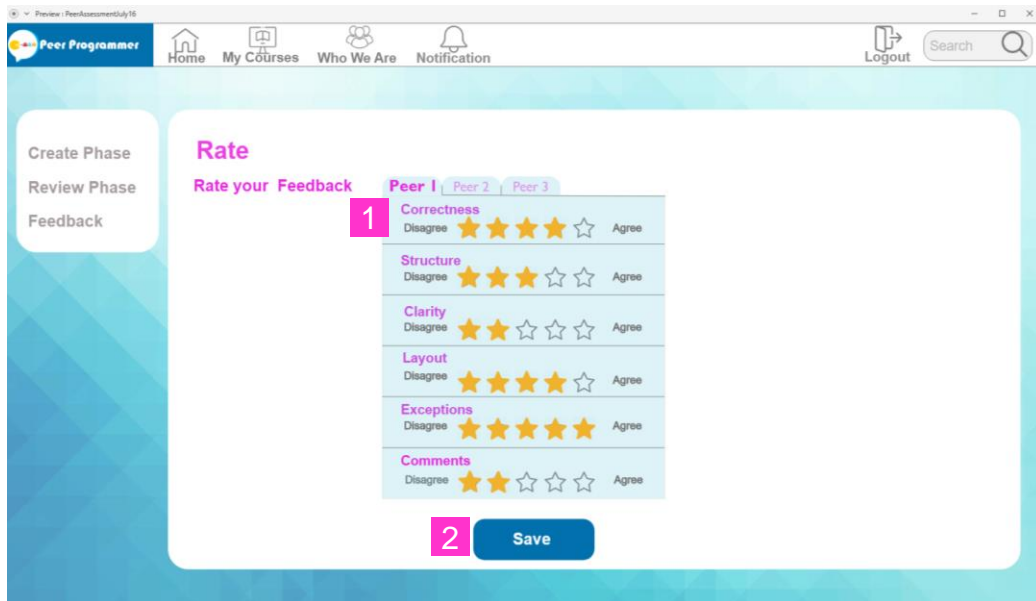
Figure 6.33. Progress page

No	Description
1	<b>Total Grades:</b> This section displays all the previous total grades in assignments that the author obtained from reviewers.
2	<b>Progress Chart:</b> This chart represents the students' progress over the span of a period of time during which peer assessment has been implemented. It is a line chart, displaying progress and clarifying the strengths and weaknesses of categories based on the slope of the lines.
3	<b>Strength and weakness:</b> This section indicates all the categories have been improved, and categories that got worse.

**Table 6.29. Description of progress page**

When designing charts for the Peer Programmer prototype, colour-blindness was considered. For example, the prototype uses a blue/orange colour scheme because it is a common colour-blind friendly palette (Colblindor, 2006). Therefore, for the Grade Comparison chart, blue and orange were used. Also, the Reviewers' Choices table adopted the following colours: blue, grey and orange. For the Progress chart, different line patterns with different colours were employed as recommended by Colblindor (2006). In the Grade and Satisfaction charts, red and green were used together as some people like colours as a visual aid. However, alternative ways to distinguish the data were introduced, by adding percentages and labels to each colour (e.g., worst, bad, average, good, and best), and annotations when clicking on any chart. This would allow the person with colour-blindness to see that there is something bad (red) versus good (green). All figures have been tested using a Colour-blindness Simulator tool called Coblis (<https://www.color-blindness.com/>). It is a tool that gives a user an impression of how a colour-blind person sees the colours. Users can upload a sample picture to see how it looks like if the user has red-, green-, blue-blindness or is completely colour-blind. As a result, all figures in the prototype are friendly to users with colour-blindness.

As a result of feedback, authors can rate the feedback received by any reviewer by giving stars in each category (see Figure 6.34, Table 6.30).



**Figure 6.34. Rating the reviewer's page**

No	Description
1	This section allows a user to rate the reviewer's assessment in each category.
2	This button allows a user to submit the rating.

**Table 6.30. Description of rating page**

After reading the reviewers' feedback, authors can edit their code and resubmit it again to the teacher (see Figure 6.35, Table 6.31). The notification of resubmission will appear to the teacher. This feature was included to support assessment for learning.



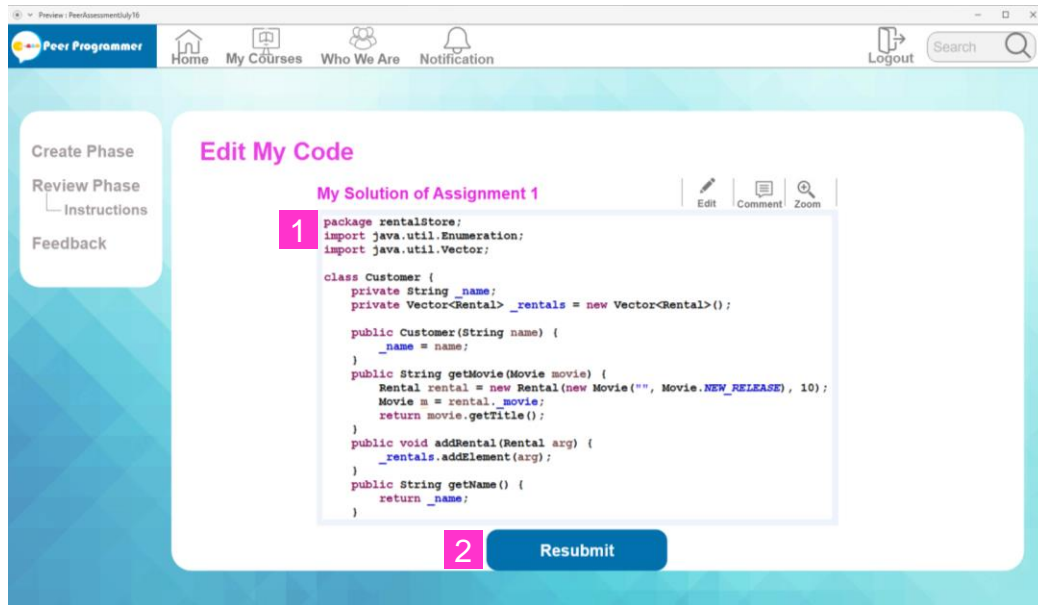


Figure 6.35. Resubmit page

No	Description
1	This section displays the author's code.
2	This button allows a user to resubmit the assignment.

Table 6.31. Description of resubmit page

### ***Instruction page***

In each phase, the instruction page contains detailed information about how the selected phase should be completed (see Figure 6.36, Table 6.32).

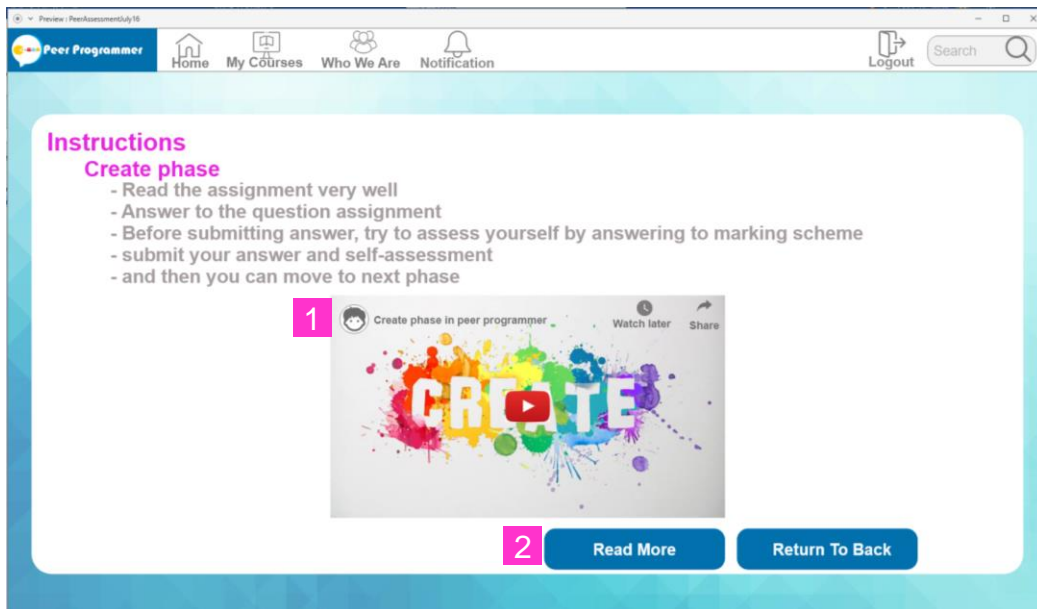


Figure 6.36. Instruction page

No	Description
1	This video displays instructions for the selected phase.
2	These buttons allow a user to read more or return to the previous page.

Table 6.32. Description of the Instruction page

### ***Who We Are page***

It is a special web page on a site where the readers/visitors learn more about the website and what it does. It contains a video explaining peer assessment and how the Peer Programmer system works (see Figure 6.37).

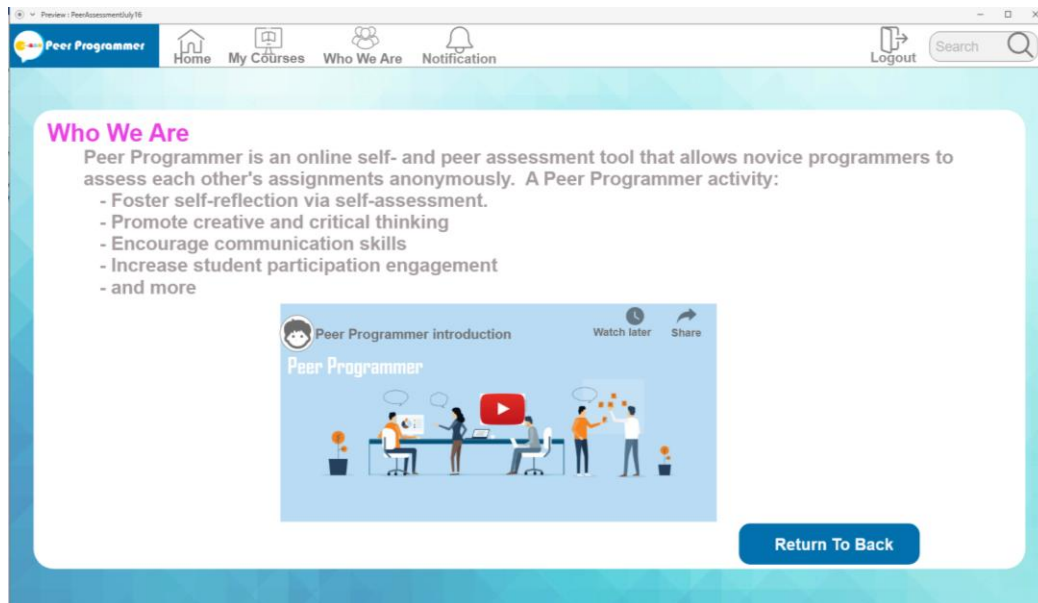


Figure 6.37. Who we are page

### ***Notification page***

This page captures the notifications users receive in relation to all their website activities (e.g., new assignment, new assessment, and new feedback) (see Figure 6.38).

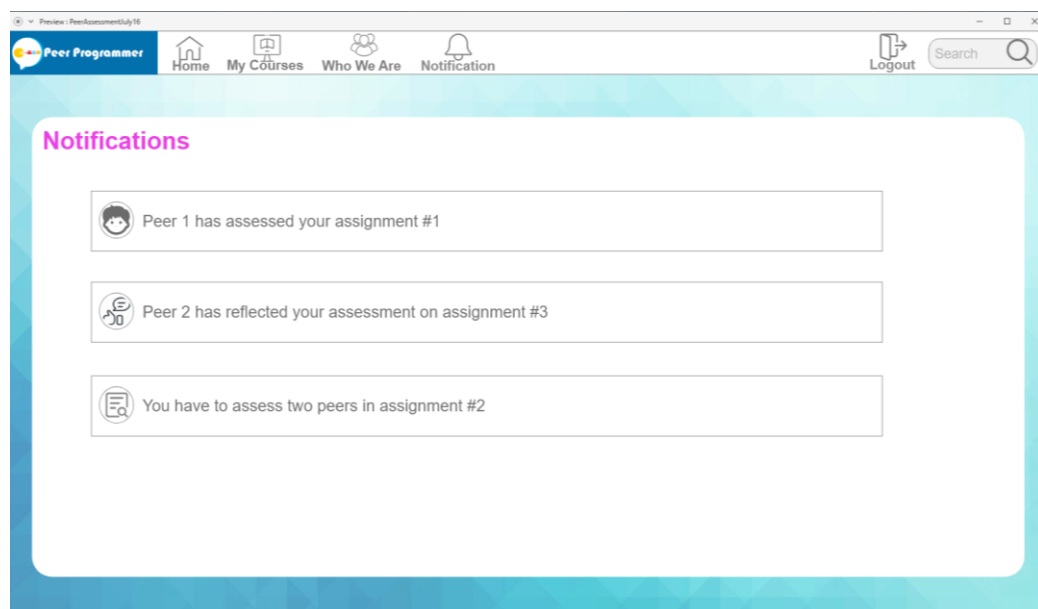


Figure 6.38. Notification page

This case study aimed to show the functionality of the website as performed by students. As part of the prototype, the student - as an author - can create a new solution for the assignment, perform a self-assessment, and submit their work in conjunction with a self-assessment. The system then matches the author and reviewers based on the author's assessment of the difficulty of the task. The user then shifts from the 'author' role to the 'reviewer' role, and the users are shown a list of anonymous solutions for the assignment they have just completed. Using the same marking scheme, the reviewers assess their peers' work. Then, the prototype displays the feedback. The feedback appears from three viewpoints: as an author, as a reviewer, and in progress over time. The author can then rate the reviewers' feedback. All of these activities can be completed by students.

## **6.6 Summary**

This chapter outlined the qualitative results gained from focus group discussions and interviews. Findings helped to answer the last two research questions outlined in Chapter 1. This chapter identified students' expectations of peer assessments in programming courses, the features they require, and the problems they were concerned with. It also described the development and evaluation of the prototype. The student participants reported favourable impressions of the Peer Programmer website as a learning tool. The students thought that the peer assessment would function very well in introductory programming courses when used at suitable times. But some teachers expressed reluctance to use peer assessment in introductory programming courses. Both teachers and students mentioned many possible features and some concerns about the Peer Programmer prototype. Significant features include using self-assessment as a pre-condition, visualising feedback to make students receive three different types of feedback based on the role they have (i.e., author or reviewer), and using rewards. One considerable concern raised by students was the need to match authors and reviewers during the peer assessment based on the author's needs. Details about the matching technique should be clarified, such as the reviewers' abilities, matching conditions, and students' satisfaction. Therefore, the next chapter outlines the matching technique suggested to pair the author and reviewers during the peer assessment activity.

## **Chapter 7. Matching authors and reviewers**

### **7.1 Introduction**

This chapter describes a tool that automatically matches authors and reviewers based on programming students' needs in a peer assessment scenario: a Balanced Allocation algorithm. Matching authors with reviewers is an adaptive technique that personalises the reviewing process based on individual author's needs. The Balanced Allocation algorithm allocates a group of reviewers to assess the work of each author, with a view to enhancing the credibility of the peer assessment and to showing students better-quality feedback. This chapter outlines the issue with peer assessment in this study. In addition, the chapter describes the Multi-Criteria Decision Making (MCDM) for making a decision when allocating reviewers between alternatives in the peer assessment process. Then a description of the development of the algorithm that matches author and groups of reviewers to review the author's work is provided. Additionally, experimental results of the algorithm with a real dataset and a mock-up dataset are presented. An evaluation of the algorithm - conducted by collecting the students' and teachers' perspectives - is provided by using focus groups discussions and interviews. Finally, the discussion section interprets and explains all significant results within this chapter.

### **7.2 Peer assessment issue in this study**

Many students who do peer assessments doubt the efficacy of the peer feedback they receive because they think that not all peers can provide useful feedback (Li, Fu and Yang, 2017). Participants discussed this in the questionnaires (see Chapter 4, section 4.2.2) and the focus groups (see Chapter 6, section 6.2.3), clarifying that suitable pairing in peer assessment increases the credibility of the peer assessment process. This view is supported by a study conducted by Patchan and Schunn (2016), who asked students whether they felt that receiving peer assessment was useful; students stated that it depends on how knowledgeable their peers are. Topping (1998) recommended further considerations for matching authors and reviewers in peer assessment; for example, to build matching based on credibility or friendship. Anaya *et al.* (2019) observed that many

studies ignore selecting reviewers in the peer assessment, as reviewers determined randomly, although peer assessment is affected by the reviewers' ability (Patchan and Schunn, 2016).

Roles of students and adequacy have commonly been analysed in the context of collaborative learning (e.g., Ward, 1987), but not so commonly in peer assessment, although both contexts are indeed similar. For instance, if an author receives feedback from a reviewer who is knowledgeable, the author may receive a significant amount of critical feedback that describes issues and suggests solutions (Patchan and Schunn, 2016). Since proficient reviewers are better at determining correctness of facts, they will be able to discover weaknesses in their peers' work. In contrast, if the author receives feedback from a reviewer who has limited knowledge of the subject being studied, the author may receive less constructive feedback that may not sufficiently describe issues or suggest solutions (Patchan and Schunn, 2016). Since a non-proficient reviewer has inferior review skills, they may not be able to make suggestions but may still offer praise for high-quality work. Another study (Patchan *et al.*, 2013) underlines this by examining the pairings of peers that were most likely to benefit all learners. The results stress that creating peer assessment groups that allowed learners to receive feedback from reviewers of different abilities seemed to be the most beneficial.

Social constructivism theory supports differentiation in abilities. Vygotsky (1980) noted that a more knowledgeable other is an important aspect of ZPD. Therefore, it is important to include more knowledgeable reviewers in each matching group. Further, Vygotsky states that the main aspect of ZPD is the focus on social, rather than individual, processes as key in the development of higher mental functions. Interactions with social agents and surrounding culture, such as more competent peers, contribute significantly to a student's intellectual development (Mishra, 2013). Therefore, as given Vygotsky's focus on social interactions and individual differences between learners, the present study built a matching technique in peer assessment based on Vygotsky's theory to ensure that peer learning takes at its maximum effectiveness.

Since students are the main focus of this study, they determined the key variables on which the matching process is built, to ensure students' engagement and satisfaction with the peer assessment. Some programming students suggested in the focus group discussions (see section 6.2.3) that the author-reviewers matching technique should be based on the author's perceived difficulty; using the self-assessment method, the author specifies how difficult the assignment was, then the system pairs the author to reviewers who did not have difficulty in the same task. Thus, this study was based on task difficulty level; further, the use of self- and peer assessments was suggested to determine the value of the difficulty level of the assignment with different weights assigned to each aspect. Selecting suitable reviewers was therefore categorised as a problem of multiple-criteria decision making in this study.

### **7.3 Multiple-criteria decision making**

Multiple-criteria decision making (MCDM) is a computational and mathematical algorithm that is common in operations research to assist in the "subjective evaluation of performance criteria" via decision makers (Mardani *et al.*, 2015, p. 516). MCDM methods seem suitable for use in author-reviewer matching to select suitable reviewers from the many options available, because the evaluation in the assessment process is based on users' subjective opinions. The MCDM process contains the following main elements: a set of criteria, a structure of preferences, a set of alternatives, and performance values (Shee and Wang, 2008). MCDM includes a number of methods that can be implemented based on the available collected data. The weighted sum model (WSM) is the most commonly used approach, particularly in single-dimension problems (Triantaphyllou, 2000), and it is easy to understand and utilise. WSM has been selected in this study because the data that can be collected and stored in the user profiles is expressed in precisely the same units (e.g., peer assessment scores, self-assessment scores). A given MCDM problem has  $m$  users' profile alternatives and  $n$  decision criteria: self-, peer, and tutor scores. All criteria are positive given that the higher the values, the better they are considered to be. Therefore, as in Equation (1),  $w_j$  indicates the relative weight of the importance of the specific criterion  $C_j$ , and  $a_{ij}$  indicates the performance value of

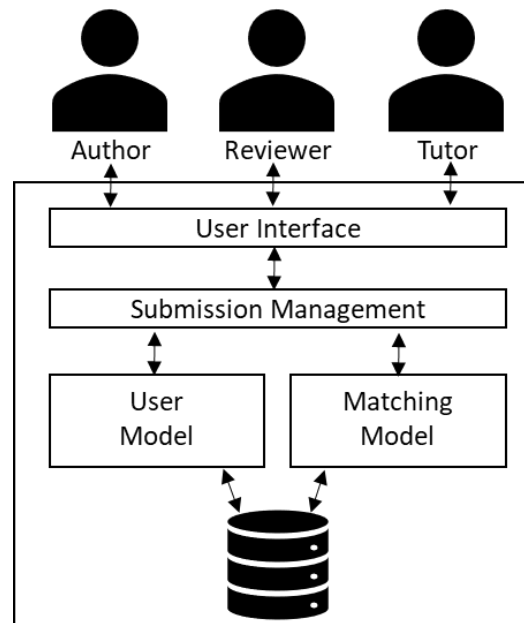
alternative  $A_i$  once it is assessed in terms of criterion  $C_j$ . The overall importance (i.e., when all the criteria are judged simultaneously) of alternative  $A_i$  is indicated as  $A_i^{\text{WSM-score}}$ . Thus, the top alternative is the one that is, in the maximisation case (Triantaphyllou, 2000), defined as follows:

$$A_i^{\text{WSM-score}} = \sum_{j=1}^n w_j a_{ij}, \text{ for } i = 1, 2, 3, \dots m. \quad (1)$$

#### 7.4 Balanced Allocation algorithm description

In a generalised peer assessment model, a set of students assessed a specific task (e.g., assignment) for a particular author. The basic tasks in peer assessment systems (e.g., PeerScholar [Collimore, Pare and Joordens, 2015] and PeerGrade [Sharma and Potey, 2018]) include self-assessment and peer assessment, followed by the official assessment and grading of the assignment by a teacher. In this study, the Balanced Allocation algorithm was able to collect the scores of a specific assignment according to three aspects - self-assessment score, peer assessment score, and teacher score. The algorithm collected the available scores and then assigned the task difficulty level for each author. Based on the resultant score, students were organised into two categories: students with difficulties and students without difficulties. Then, a set of reviewers was assigned to each author based on the author's category, taking into account that all matching groups for each peer assessment process were balanced in the number of reviewers and the ability levels of participants. Figure 7.1 shows the architecture of the algorithm.





**Figure 7.1. Architecture of the Balanced Allocation algorithm**

Figure 7.1 shows all possible stakeholders who can assess any specific task in the peer assessment process through the user interface of the peer assessment system. Author-review matching can be modelled as a scenario consisting of the following steps:

1. The author works on a given task and performs a self-assessment. The author submits the resulting document, which comprises their self-assessment and their solution for the task. The system calculates the author's score in the self-assessment and keeps the score in the user's profile the first time the system runs (the user profiles only include self-assessment scores in the first round).
2. Based on the user profiles, the students are divided into two groups based on their self-assessment scores: the proficient group who do not have task difficulty (top 50% of students) and the non-proficient group who have task difficulty (bottom 50%). The students were divided into two equal groups to avoid a lack of proficient students, especially with large datasets. Thus, the system decides the task difficulty level of the author based on their self-assessment and stores it in the corresponding user profile.
3. Each submission is assigned to a set of reviewers based on the user profiles outlined in the previous step. When assigning reviewers to students, four pairings are possible:

1) a non-proficient student reviews the work of another student with similar abilities, which is a case that should be avoided; 2) a proficient student assesses high-quality work; 3) a non-proficient student assesses work created by a proficient student; 4) and a proficient student assesses the work of a non-proficient student. For the purpose of this study, a non-proficient student's work should be reviewed by at least two proficient students, and the work of a proficient student should be reviewed by at least one proficient student. As such, the proficient students were able to benefit from at least one person at the same proficiency level, and the comments of non-proficient students would not affect their work. Thus, for all users, there were two proficient students and two non-proficient students in the review groups; hence, the algorithm achieved balanced allocation in each group.

4. After having assessed their peers, each reviewer assesses the peer's work; the system calculates the peer score and then sends it back to the author profile to update the user profile based on the new value of the WSM. Thus, the tool uses self-assessment and peer assessment to divide students into two groups, assigning different weights for each criterion.
5. Once a tutor assesses the students, the tool calculates the total sum of the authors' scores for each user profile based on the WSM. This method does not contribute to student grading to ensure that students provide feedback that is as objective as possible.

Figure 7.2 is a flowchart diagram that explains the Balanced Allocation algorithm process and provides the reader with a visual representation of what occurs in the algorithm. The algorithm was implemented using the R i386 3.6.2 platform environment (R, 2021). Since learning analytics is statistics heavy, R language is an ideal tool for implementing different statistical operations on it. It also provides aesthetic visualisation tools (e.g., scatter plots). First, the user profiles were constructed; they may consist of data on self-assessment, peer assessment, and tutor assessment. However, these three elements are not always available together; thus, the algorithm can work even if only one of the data categories has been collected. Based on the collected data, the tool divided authors into two equal

groups: the top 50% of users were categorised as proficient, and their IDs were saved in the Available Proficient List; the rest of the users were classified as non-proficient, and their IDs were saved in the Available Difficulties List. Upon completing this categorisation, the loop of assigning three reviewers to each author began, because it was not feasible to ask students to conduct more than three peer reviews (Sung *et al.*, 2010). The selected author ( $k$ ) should be removed from the Available lists, whether the Available Difficulties List or the Available Proficient List, because the author cannot review their own work. Afterwards, if the selected author was proficient, the tool selected different random reviewers from two separate lists - one reviewer from the Available Proficient List and the other two reviewers from the Available Difficulties List.

In contrast, if the author was non-proficient, the tool selected different random reviewers from two separate lists - two reviewers from the Available Proficient List and one reviewer from the Available Difficulties List. After that, the IDs of the selected reviewers were added to the Reviewer List, which is a matrix that contains two-dimensional elements. To choose another reviewer for a specific author, the selected author should be removed from the Available Proficient List or the Available Difficulties List, because the reviewer is not allowed to assess the work again. The tool counted the number of reviewers for each author to make sure the review process was equal for all users; in the present scenario, this included three reviewers. Thus, no non-proficient student could assess another non-proficient student unless there were already two proficient users listed to assess this assignment. As a result, no students assessed themselves, no reviewers were assigned to assess the same task, and there were multiple reviewers for each author.

At each stage of the peer assessment process, the algorithm used the MCDM to estimate the reviewers for each author. Each time the algorithm was used to produce reviewer–author pairs, it updated the user profiles to incorporate recent peer behaviour and decide which author should be categorised into which group.

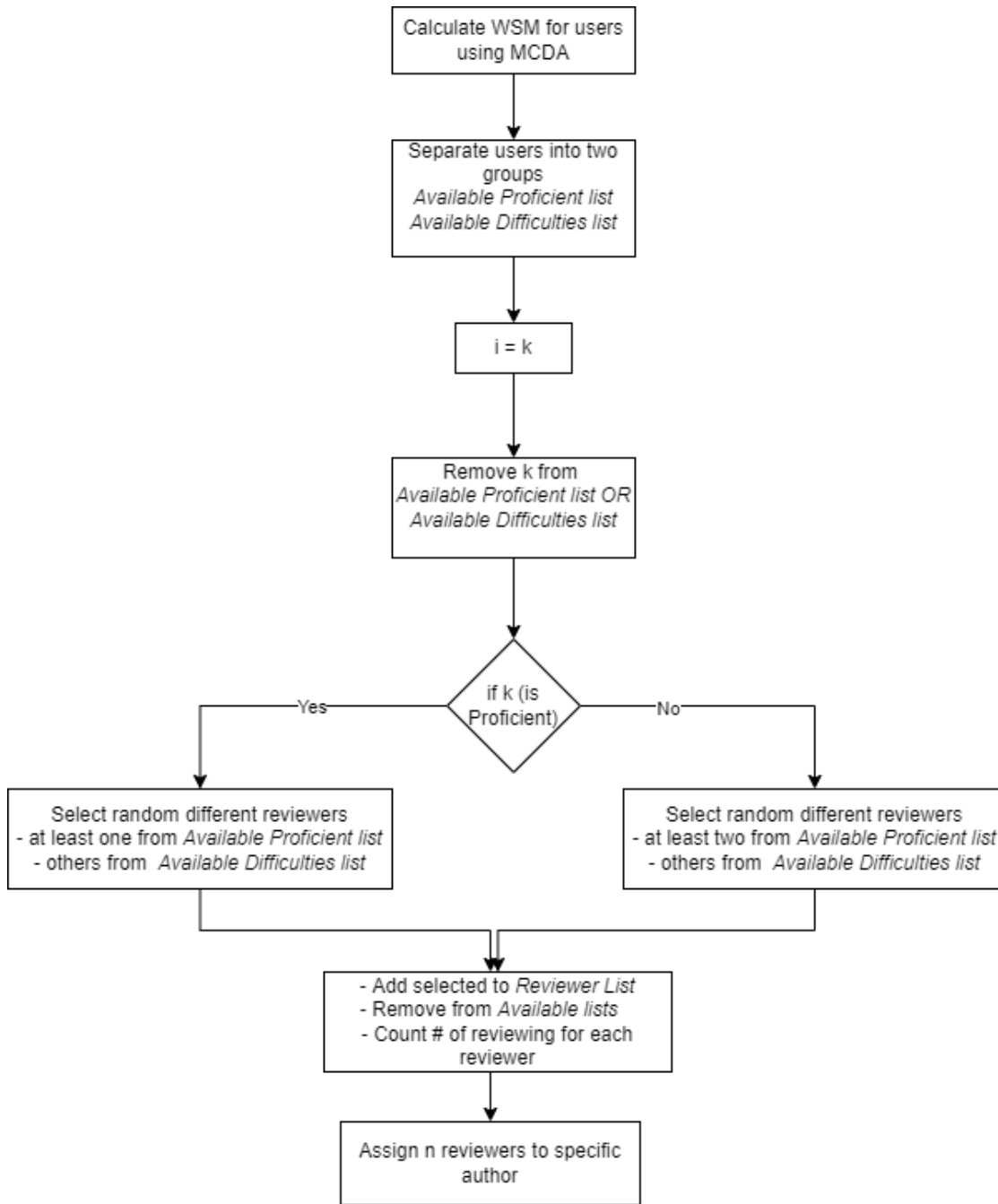


Figure 7.2. Algorithm description

## 7.5 Pseudocode of the algorithm

```
Import dataset from user profile
Assign weights
Calculate WSM Score
function(proficient){
    filter out non-proficient authors
    filter out proficient authors

    for (i in Dataset) {
        Remove i from availableDifficultyIDs
        Remove i from availableProficientIDs

        if(i is proficient){
            if(length(availableProficientIDs) >= 1){
                usedIDs = c(availableProficientIDs[1], availableDifficultyIDs[1:3], a
availableProficientIDs[2:4])
            } }

        else {
            if(length(availableProficientIDs) >= 2){
                usedIDs = c(availableProficientIDs[1:2], availableDifficultyIDs[1:3],
availableProficientIDs[3:5])
            } }

        ReviewerList[i,] = usedIDs

        return(ReviewerList);
    }
}
```

## 7.6 Experimental results related to the algorithm

This algorithm was applied to a real collected dataset and mock-up data for evaluation. In this section, the results of the selected datasets are described.

### 7.6.1 Algorithm implementation on a real dataset

#### Sources of evidence

To evaluate the Balanced Allocation algorithm, the actual users' data were selected for use. During the academic years 2005–2008, Newcastle University, UK was a partner in the Active Learning in Computing (ALiC) project in a software engineering course for second-year undergraduate students (Devlin, 2015). ALiC was a project that focused on increasing the level of student engagement within the computing curriculum and aimed to

make their experiences more relevant to the industry. In total, there were 240 students in this dataset. These data proved valuable for the present study as it was possible to use the following information: summative module marks for all students completing the software engineering course that were graded by teachers, peer assessment results from the team project, and an individual reflective report that described each student's role in the project and the areas in which they had participated. The individual reflective report was completed by the students themselves; therefore, it was classified as a self-assessment. Although the dataset is old, it was appropriate to use in this study because only self-assessment scores, peer scores, and tutor scores for the same assignment were needed, and there is a lack of open datasets that combine these variables. These results were then coded in a Microsoft Excel worksheet for the purpose of analysis. The mark data were anonymised, and all records for students who did not finish the module were removed.

### **Determining the weights of the attributes**

The attributes associated with a user profile affected the user categorisation results to different extents. As discussed above, based on the user profile, each user was classified as a proficient (having no difficulties in a task) or non-proficient user. Note that this categorisation helped to match the authors and reviewers, but it did not affect the students' official scores, as the official scores for the assignments should be decided by the tutors. The weight associated with an attribute reflects the emphasis that is to be placed on it; thus, changing the pattern of weights allocated to various attributes will impact the results of the user's categorisation. The following method was used to determine the appropriate weights for self- and peer assessment. The dataset used for this example contained self-assessment scores, peer scores, and tutor scores for 240 students. The following example (self-score for student  $i = 60$ , peer score = 50, and tutor score = 58/100) outlines the method used:

1. All scores were standardised so that each column was scored out of 10. For example, self-score for student = 6, peer score = 5, and tutor score = 5.8/10.

2. The self-assessment and peer scores were identified and found to be similar to the tutor scores, given that students' scores for themselves and their peers were subjective. Thus, if the selected score for a specific user was within 0.5 points of the tutor score, this selected score was similar to the tutor score, while the others were dissimilar. For example, because the self-assessment was 6, it was similar to the tutor score, but the peer assessment score was not similar to the tutor assessment because it was not within 0.5.
3. The number of similar students in terms of self-assessment score and peer score were counted. Table 7.1 shows the number of similar students.
4. The percentages of similar results for self-assessment scores and peer scores were calculated as 17% and 37%, respectively. As a result, the weight of the self-assessment score was 17%, the weight of peer scores was 37%, and the weight of the tutor score was 46%.

<b>Percentage Of Similar Results</b>		
<b>Total</b>	<b>224</b>	<b>100%</b>
<b>Self-Similar (<math>\pm 0.5</math>)</b>	39	17%
<b>Peer-Similar (<math>\pm 0.5</math>)</b>	82	37%

**Table 7.1. Weights of attributes**

## **Method results**

In this section, the primary experimental results of the algorithm are presented. Table 7.2 shows the accuracy results of the algorithm in matching the first one, two, and three reviewers for each author. Four random author IDs were selected by displaying the authors' scores based on the WSM model using the Balanced Allocation algorithm. Based on the data, the scores of the students were arranged in ascending order, and then the students were divided into two equal groups. The number used for separating the students into two groups - proficient and non-proficient students - was 61.68. Therefore, students who scored higher than this number were classified as proficient students, and the other students were classified as non-proficient students. In Table 7.2, the total score using the WSM method was calculated using all three attributes (self-assessment, peer, and tutor

scores). Authors who were deemed proficient were allocated two non-proficient and one proficient reviewer. In contrast, non-proficient authors were allocated two proficient reviewers and only one non-proficient reviewer. The scores of the pairs shown in Table 7.2 illustrate that there were two proficient students and two non-proficient students in each row. In addition, the IDs shown here satisfied the algorithm’s other conditions: there are three reviewers for each author, no author can assess their own work, no author can be assigned to assess the same task more than once, and no reviewer can assess more than three times during each peer assessment process.

<b>N</b>	<b>Student's ID Its score</b>	<b>Author</b>	<b>Reviewer1</b>	<b>Reviewer2</b>	<b>Reviewer3</b>
<b>#1</b>	<b>ID</b>	<b>103</b>	<b>176</b>	222	170
	<b>Score</b>	<b>71.7</b>	<b>71.4</b>	48.4	50.4
<b>#2</b>	<b>ID</b>	<b>139</b>	<b>136</b>	213	141
	<b>Score</b>	<b>69.6</b>	<b>70.3</b>	60.1	61.4
<b>#3</b>	<b>ID</b>	156	<b>208</b>	<b>87</b>	191
	<b>Score</b>	58.6	<b>81.2</b>	<b>64.4</b>	56.2
<b>#4</b>	<b>ID</b>	222	<b>173</b>	<b>208</b>	207
	<b>Score</b>	48.3	<b>72.9</b>	<b>81.2</b>	56.3

**Table 7.2. Algorithm implementation on a real dataset**

The following scatter plots show the score distributions of the authors and reviewers in the peer assessment as a result of the Balanced Allocation algorithm. Figure 7.3, (a) shows that all first reviewers’ scores were higher than 61.68 points, whether for proficient authors or non-proficient authors; this means that for each author, the first reviewer should be proficient (having no difficulty in the task). Figure 7.3, (b) illustrates the score distribution of the second reviewer for all authors. There were two clusters: the first cluster represents authors who had scores lower than 61.68 (in this case, the second reviewers should have had scores higher than 61.68); and the second cluster represents authors who had scores higher than 61.68 (in this case, second reviewers should have had scores lower than 61.68). This meant that non-proficient authors matched with proficient reviewers and vice versa. Figure 7.3, (c) shows that all third reviewers’ scores were lower



than 61.68 points for proficient authors and non-proficient' authors, which meant that for each author, the third reviewer was non-proficient.

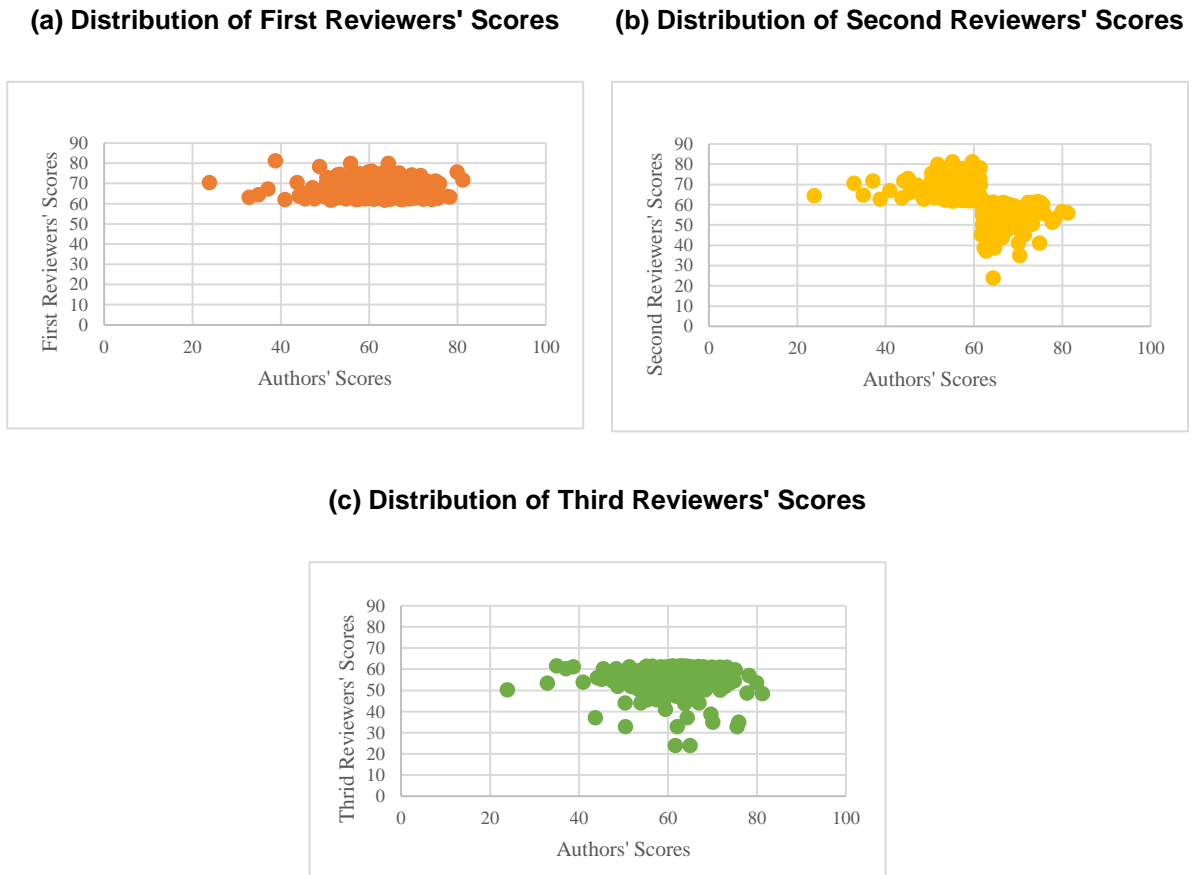


Figure 7.3. Score distribution between authors and reviewers in real dataset

### 7.6.2 Algorithm implementation on the mock-up dataset

#### Sources of evidence

To examine the algorithm using a large dataset, the dataset generation website Mockaroo was used (*Mockaroo Website*, 2021). This online tool generates random data as rows of realistic test data in various formats (e.g., CSV, JSON, SQL, and Excel). The tool creates self-assessment scores only. The Mockaroo tool was not used to produce random peer and tutor scores because these three elements are often close if they process the same project or task; thus, such a tool cannot decide random scores that relate to each other. Hence, this experiment focused on producing self-assessment scores.

### Weights of the attributes

Because this dataset was randomly generated, the same weights were used for all attributes as in the previous case study. The weights of the self-assessment scores and peer scores were 17% and 37%, respectively, and the weight of the tutor scores was 46%. As the Mockaroo only randomly generates the self-assessment scores, the total score of the WSM was 17 points, given that the weight of the self-assessment score was 0.17.

### Method results

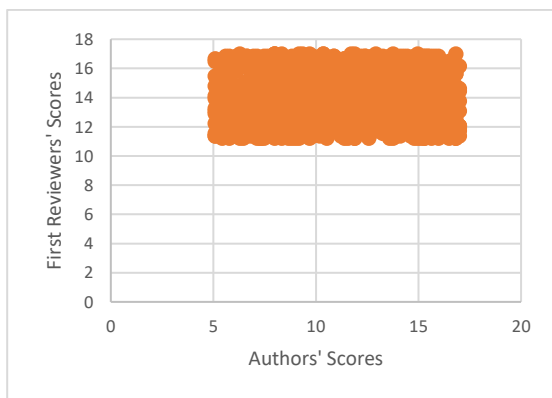
The algorithm was implemented based on the available dataset, which contained self-assessment scores for 1,000 records. Table 7.3 shows the results of the dataset. Some author IDs, matching reviewers' IDs and their scores were randomly selected to display in the table. Based on the generated data, the score that separated students with high and low abilities was 11.22. Therefore, students who had scores higher than this number were classified as proficient students, while others were classified as non-proficient students. As presented in Table 7.3, if the author had a score less than 11.22, they were classified as a non-proficient student, and therefore matched with two proficient reviewers and one non-proficient reviewer. By contrast, an author with a score higher than 11.22 was classified as a proficient student and was therefore matched with one proficient reviewer and two non-proficient reviewers.

No	Student's ID Its score	Author	Reviewer 1	Reviewer 2	Reviewer 3
#1	ID Score	1 15.6	497 16.7	604 8.8	913 6.6
#2	ID Score	92 9.4	648 12.9	345 15.0	81 10.2
#3	ID Score	252 14.6	585 16.5	67 10.7	55 5.4
#4	ID Score	593 7.3	367 16.0	29 13.8	97 6.5

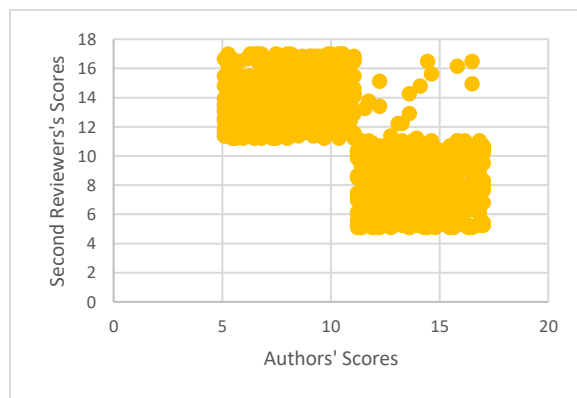
Table 7.3. Algorithm implementation on the mock-up dataset

The following scatter plots (Figure 7.4) illustrate the score distributions of the authors and reviewers in the peer assessment as a result of the algorithm using the Mockaroo dataset. Figure 7.4, (a) shows that all the first reviewers' scores were higher than 11.22 points, whether for proficient or non-proficient authors, and were therefore assessed by proficient reviewers (higher than 11.22). Figure 7.4, (b) shows the distributions of the second reviewers based on the authors' scores. As in the previous dataset, there were two clusters: the first cluster included authors who scored lower than 11.22, and the second cluster included authors with scores higher than 11.22. Reviewers for each author belonged to the opposite cluster. It is obvious that some pairs belonged to the same cluster (both members of the pair were proficient). Figure 7.4, (c) also displays all the third reviewers' scores. These were below 11.22 points, except in some instances where one of the algorithm conditions was selecting (at least) a specific number of proficient reviewers for each case; thus, the algorithm accepted more than two proficient reviewers for non-proficient authors and more than one proficient reviewer for proficient authors. This situation occurred that all students from the Available Difficulties list were busy, which does not conflict with Vygotsky's theory. The algorithm distributed students based on their abilities and then paired them in a logical way based on a Vygotsky theory rather than random selection.

**(a) Distribution of First Reviewers' Scores**



**(b) Distribution of Second Reviewers' Scores**



(c) Distribution of Third Reviewers' Scores

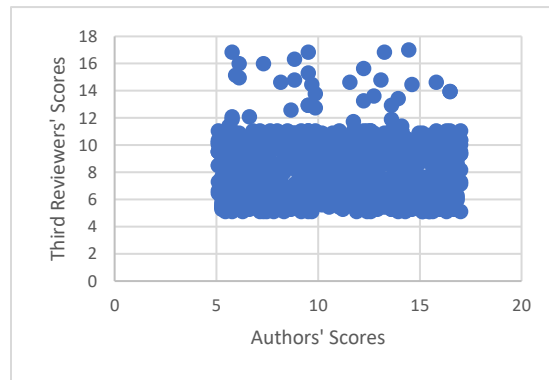


Figure 7.4. Score distribution between authors and reviewers in mock-up dataset

## 7.7 Evaluation of the algorithm

Given that learners are the core factor in peer assessment activities, the evaluation of the algorithm was based on students' opinions. Additionally, programming teachers' perspectives were considered as they have experience in developing algorithms. Students participated in interactive focus groups combined with survey questions to carefully assess the participants' perceptions of matching authors and reviewers in peer assessment. For teachers, structured interviews were conducted, and the main question form was distributed prior to the interview so they could outline their views regarding the algorithm. Table 7.4 includes the discussion questions for the students and teachers. Each participant was given the participant consent form, survey questions, and they gave their authorisation for the researchers to conduct and record the discussions.

Topic	Question	Duration
<b>Matching criteria</b>	What are the criteria you recommend for matching authors and reviewers in peer assessment?	~7 min
	Do you think the task difficulty level could determine who gets assigned to be a reviewer for a specific author? Who can assign this ability level?	~7 min
<b>Matching process</b>	What do you think the following output of matching: The process of matching depends on an author's need; if the author has difficulties in his/her solution,	~10 min

Topic	Question	Duration
	he/she needs at least two proficient reviewers, if the author has not difficulties in his/her solution, one proficient reviewer is enough?	
	What were the features of the algorithm that made a difference?	~5 min
	What is hindering the algorithm to achieve its objective?	~5 min
<b>Matching</b>	How can the algorithm achieve better outcomes?	~10 min
<b>output</b>	<p>Please rate the following points for the suggested matching process based on your personal value.</p> <p style="text-align: center;">1   2   3   4   5</p> <p>acceptable   <input type="checkbox"/>   <input type="checkbox"/>   <input type="checkbox"/>   <input type="checkbox"/>   <input type="checkbox"/>   Not acceptable</p> <p>Useful   <input type="checkbox"/>   <input type="checkbox"/>   <input type="checkbox"/>   <input type="checkbox"/>   <input type="checkbox"/>   Not Useful</p> <p>Important   <input type="checkbox"/>   <input type="checkbox"/>   <input type="checkbox"/>   <input type="checkbox"/>   <input type="checkbox"/>   Not Important</p> <p>Fair   <input type="checkbox"/>   <input type="checkbox"/>   <input type="checkbox"/>   <input type="checkbox"/>   <input type="checkbox"/>   Unfair</p>	~10 min
	Is there anything else you want to add to the conversation about this suggested algorithm?	~5 min

**Table 7.4. Discussion questions to evaluate the algorithm**

### **7.7.1 Procedure of data collection**

The focus group discussions with students were the same as the third focus group discussions that were conducted between the 30 September and the 16 October 2020 with 29 undergraduate students who had studied, or were currently studying, computer programming at various Saudi universities (see Chapter 6, section 6.2.2). However, the attention here was only questions related to matching authors and reviewers.

The staff interviews were conducted online between the 15 and the 30 June 2021. Seven programming teachers participated in the interviews - three from PNU University and four from Newcastle University. The teachers had various roles - assistant professor (29%), associate professor (57%), and professor (14%). Four programming teachers had used peer assessment in their courses. The other teachers had never applied peer assessments before.

All participants were given the consent form that included the question form so that they could outline their ideas. After the study was concluded, the student participants were given an online certificate of thanks for their voluntary participation.

### **7.7.2 Data analysis**

Most of the discussions were conducted in Arabic; the only four interview discussions conducted in English were at Newcastle University. All interviews were recorded through Zoom (*Zoom Website*, 2021). Recordings were transcribed using the 'summarised transcript' technique (Baxter, Courage and Caine, 2015) for thematic analysis. After the transcripts were complete, they were revised for accuracy. The coding process followed and included open, axial, and selective coding, where all phases were used to analyse the transcript data. All codes were written in English and were reviewed to identify the main themes. The frequencies of the codes' appearances in the transcripts were used to provide credibility to these codes. Concerning inter-coder reliability, consensus coding was used; an external bilingual researcher volunteered to analyse the discussions to review the codes and themes. Then, the researcher and volunteer researcher discussed and confirmed the codes and themes.

### **7.7.3 Evaluation results**

The following three main points were discussed with participants, both students and teachers, to determine the aspects of the algorithm: 'input elements of algorithm', 'process and output of algorithm', and 'tool efficiency'.

#### **Theme 1: Input elements of the algorithm**

**Teachers' viewpoints:** At the beginning, the interviewer asked participants about the criteria they would recommend for pairing students; the most frequent input was previous knowledge (43%). During the interviews, some teachers indicated that they believed that the amount of one's previous knowledge influences one's ability to use higher-order cognitive problem-solving skills, as required in peer assessment. The interviewer highlighted the impossibility of collecting previous knowledge in some cases - for example, in introductory programming courses at undergraduate level, where no previous tutor

assessment data existed. One participant suggested: “*As a part of that self-assessment, you can ask students if he/she would have done any programming before, and you could give a list of languages.*” Other participants also suggested asking students about other criteria that could affect previous knowledge in programming courses, such as mathematical background, history of learning multiple languages, and experience with various problem-solving games (e.g., puzzles). Five teachers agreed on using task difficulty level as an input; however, many teachers mentioned that self-assessment should not be considered as the only factor to determine the difficulty level. The interviewer asked the participants about distributing percentages among the factors they suggested. Some participants agreed to allocate a high percentage to previous knowledge, but they failed to decide what the percentage should be. Moreover, all participants agreed that self-assessment should not account for more than 30% of an overall weight in the algorithm. As a result, many teachers agreed to use task difficulty level as an input, but some of them suggested adding previous knowledge as another input to the algorithm.

**Students’ viewpoints:** Eighty-six percent of students agreed on selecting the difficulty level to determine who is assigned to be a suitable reviewer for each author; only 14% disagreed. One student who agreed said, “*If my actual score was not affected, I’m able to determine the difficulty level of the task by myself.*” All of the participants who agreed selected both self-assessment and peer assessment as methods for assigning the difficulty level for each user, which meant students could be used as a source of data for user profiles. The interviewer asked the participants about distributing percentages among the three vectors: self-assessment, peer assessment, and tutor assessment. All participants agreed that self-assessment should not account for more than 20%, while peer assessment and tutor assessment could account for 30–60%. Figure 7.5 shows students’ distributions of assigning the ability level between self, peers, and tutor. This result was in line with the weight of the attributes determined in the experimental results, as self-assessment was 17%, peer assessment was 37%, and tutor assessment was

46%. Thus, most of the students supported the use of the task’s difficulty level in their user profiles, which can be assigned by themselves and their peers.

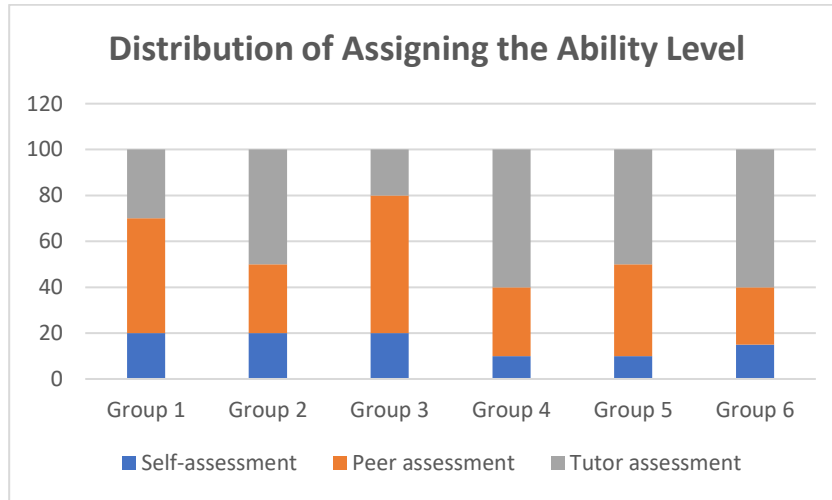


Figure 7.5. Students’ distribution of assigning the ability level

## Theme 2: Process and output of the algorithm

**Teachers’ viewpoints:** The matching process was explained to the participants, and they were informed that matching would depend on the author’s needs, as determined in the Balanced Allocation algorithm. If the author had difficulties in their work, they needed at least two proficient reviewers; if the author had no difficulties in their work, one proficient reviewer out of three reviewers was sufficient. The main features that caught the teachers’ attention included the fact that the pairings in the peer assessment activities were not randomly selected; the algorithm was based on Vygotsky theory; and students were required to complete the self-assessment upon submission, which they noted as something that is not typically done. Six of the teachers thought the algorithm included a good balance, and one participant said: *“It is important to have a mix of students who have different abilities level within each group in peer assessment, so that all students in such group can learn something from that diversity in the group.”* However, one participant suggested dividing groups into three rather than two groups might lead to better outcomes. The participant considered the motivation of students with difficulties if they received feedback from two proficient students, saying, *“That is a big jump in the ability, you have*



*to be careful with first-year students, they scare easily.*” Therefore, many teachers were satisfied with student distribution in the algorithm.

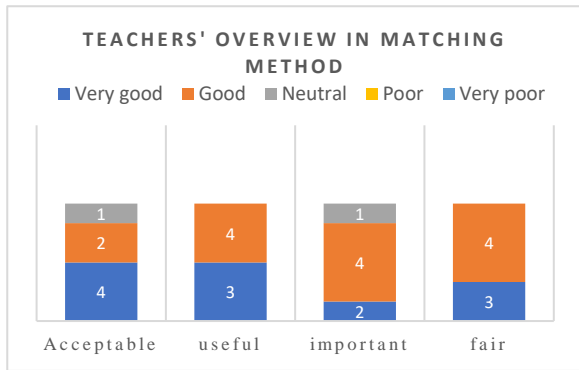
**Students’ viewpoints:** Before explaining the process of matching, most of the participants mentioned that reviewers with a high level of proficiency in programming skills should assess their tasks, which indicated the importance of allocating a proficient reviewer in programming skills in each group. The matching process was explained to the participants too. Most of the students (93%) agreed with the process of this matching and its output. One participant suggested another method for matching: *“It is supposed that the skills be based on our mutual strengths and weaknesses (author and reviewer). I suggest that the reviewer will be chosen based on his/her strengths, which are my weaknesses, to complement each other.”* As a result, many students were satisfied with the process of the Balanced Allocation algorithm in the distribution of reviewers between groups based on the authors’ needs.

### **Theme 3: Efficiency of the algorithm**

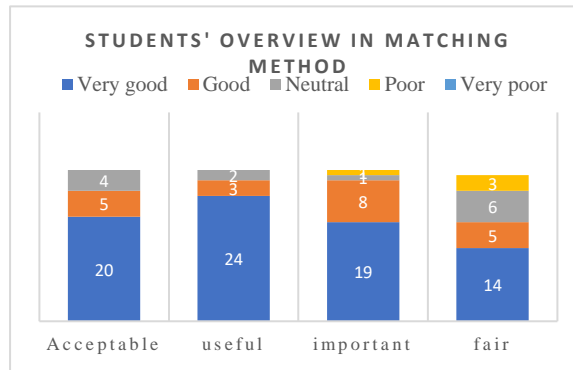
**Teachers’ viewpoints:** Figure 7.6 details the teachers’ acceptance of the matching technique, their view on the fairness of the matching, its usefulness, and how important this technique would be to students. The teachers believed that this technique was highly acceptable (6 out of 7 accepted the algorithm) and useful (all believed in its usefulness). The teachers also thought it was important (only one participant was neutral) and fair (all agreed on its fairness). None of the teachers selected a negative perspective regarding the algorithm.

**Students’ viewpoints:** Figure 7.7 shows the students’ viewpoints regarding the algorithm. The students also believed that this technique was acceptable ( $n = 25/29$ ), useful ( $n = 27/29$ ), and important ( $n = 27/29$ ); however, there was lower agreement regarding fairness ( $n = 19$  out of 28). Indeed, there were a few doubts about fairness (neutral = 6 and poor = 3). The students who did not agree with the fairness of the algorithm thought the proficient authors would not be fair if only one proficient and two non-proficient students assessed them. They suggested other techniques; for instance,

matching based on strengths and weaknesses between the authors and reviewers; or allowing only proficient reviewers to assess peers.



**Figure 7.6. Teachers' overview of the matching method**



**Figure 7.7. Students' overview of the matching method**

In summary, the evaluation results were very promising. Most of the results depicted a high level of acceptance of the Balanced Allocation algorithm. Teachers agreed with the matching process in peer assessment, but they stressed the need to incorporate prior knowledge into the self-assessment question form and that it should hold weight when dividing students into groups. The students also agreed on the matching process. Most students agreed that the difficulty level should be set as a value for choosing which reviewer should be assigned to a specific author. Teachers and students stated their satisfaction according to four aspects: acceptance, usefulness, importance, and fairness. Ultimately, the Balanced Allocated algorithm was found suitable for application.

### 7.8 Discussion of matching author-reviewers

As has been shown in this thesis, peer assessment is well suited for author-reviewer pairing; thus, this chapter outlined the development of a Balanced Allocation algorithm to match authors and reviewers. This personalised matching is a type of adapting in a learning process because it carefully assigns a group of reviewers to a particular author to maximise the benefit of peer assessment. The purpose of adaptive learning tools is to provide efficient access to relevant content for the current user, besides providing many features (e.g., just-in-time feedback, pathways, and resources), by creating particular

information-based content support (Sarıyalçınkaya *et al.*, 2021). The matching technique used in the current study creates a personalised learning path (by assigning optimal reviewers) and adapts it to the learner's abilities (by considering the task difficulty level), using learning analytics. Learning analytics seek to support the individual learner experience and derive maximum benefit for the learner based on an analysis of data. However, the relationship between adaptive learning and learning analytics has not been clearly defined yet (Sarıyalçınkaya *et al.*, 2021), particularly in the context of peer assessment activities. Therefore, this study suggests that further research in peer assessment data analytics is needed so that adaptive peer assessment systems can be developed.

The pedagogical premise on which the present study is based is ZPD theory. Vygotsky's work on ZPD is instrumental in understanding “cognitive disorientation”, which is characterised in this study by a learner experiencing concern if the assignment presented is too difficult, and learners need a more knowledgeable person to help them. When students are in the zone of proximal development, they will improve their knowledge by joining more experienced and competent others (Li and Gao, 2016). This thesis has argued that building a tool based on ZPD can assist learners to identify issues they need to consider and finding reviewers who can help them with these issues. The algorithm developed as part of this thesis can then address poor engagement in peer assessment because it meets users' personal needs. Further, such an algorithm may reduce frustration and boredom when students use peer assessment, as this study identified students' issues with peer assessment and it provided support to students in their issues; thus, they can be more engaged in peer assessment. Other studies have developed adaptive e-learning systems based on the ZPD theory (Maravanyika, Dlodlo and Jere, 2017; Imhof, Bergamin and McGarrity, 2020). However, according to the available data, peer assessments have never included adaptative matching using ZPD. The Balance Allocation algorithm applied the matching based on ZPD theory. In the interviews and focus group discussions conducted as part of this study, the participants agreed with this distribution based on ZPD, and they thought it was acceptable and useful. Hsiao and

Brusilovsky (2008) similarly used a questionnaire to collect students' opinions regarding the usefulness of stronger students in the peer review process. Most of the students in their study were satisfied with the overall peer-reviewing experience, and they strongly agreed with the need for matching in peer review to ensure the quality of the reviewer comments.

Since students are the central part of peer assessment, the input variables of the algorithm that control the matching process were selected based on students' perspectives. Many students in the focus group discussions (see Chapter 6, section 6.3.3) proposed using the task difficulty level as a variable that would assign a set of reviewers; as the students indicated, task difficulty level could sufficiently determine suitable reviewers for specific programming assignments. Thus, the algorithm developed as part of this study was built based on a task difficulty level. In fact, many adaptive learning systems are based on difficulty levels as a source of personalisation information (Kritikou *et al.*, 2008; Tseng *et al.*, 2008). Self-assessment scores were chosen in this study to source the task difficulty level in the user profile. Self-assessment is a valuable activity in this regard as it offers unique data about the learner. Learners have a realistic sense of their own strengths and weaknesses and can use knowledge of their own achievements to steer their studies in productive directions (Lew, Alwis and Schmidt, 2010). According to Machado *et al.* (2008), self- and peer assessment scores seem to be reliable, although not necessarily valid. This study also used peer assessment data to support user profiles; however, peer assessment must be given more weight than self-assessment. This result does not depart significantly from the findings of Birjandi and Siyyari (2010), who indicated that peer assessment seemed to be more efficient than self-assessment. Moreover, user profile requires data to be generated continuously in order to personalise learning for each individual in real time (Maravanyika, Dlodlo and Jere, 2017). As this algorithm performs, it updates the user profile at every matching process to make sure the author will get a suitable reviewer every time he/she uses a peer assessment activity. However, the user profile is currently based on a few dimensions (e.g., self-assessment, peer assessment, and teacher

assessment), so there is a need to expand this set of variables to improve the sensitivity of the matching provided to learners.

The algorithm worked well in this study and efficiently determined the optimal reviewers for a particular author during the peer assessment process. MCDM was used to decide a group of reviewers for each author. This method is usually used to address complex problems with conflicting objectives. It fits a situation when a researcher has a set of options in real life, and he/she wants to select the ideal one. MCDM provided a base for selecting and prioritising reviewers. More generally speaking, MCDM has proven its effectiveness in the field of e-learning systems. For example, MCDM has been used to specify tailored learning units for individual students (Chrysafiadi *et al.*, 2019; Kurilovas, 2019). Therefore, MCDM seems to be an effective method for making decisions regarding selecting reviewers for authors during the peer assessment process. There are several MCDM methods available such as the analytical hierarchal process (AHP), the analytical network process (ANP), data envelopment analysis (DEA), Technique for the Order of Prioritisation by Similarity to Ideal Solution (TOPSIS), and fuzzy decision-making (Başaran, 2016). Each method is suitable for special cases depending on the data available for decision-making. Future research in this field may consider how researchers can use different methods of decision-making to choose appropriate reviewers in the peer assessment according to the availability of data in the user profile.

## **7.9 Summary**

This chapter suggested a technique to match authors and reviewers in a peer assessment process to improve programming students' engagement in the peer assessment activity. The pedagogical approach that underpins this algorithm is social development theory. The Balanced Allocation algorithm matched authors with sets of reviewers using an MCDM methodology. WSM was used to decide the total scores for all users based on scores from different sources (e.g., peer assessment scores, self-assessment scores, and teachers' scores). The algorithm then assigned each author's task difficulty level and applied matching authors and reviewers based on the authors' needs, taking into consideration that all matching groups to each peer assessment process are equal in the number of

reviewers and their abilities level. Real and mock-up datasets were used to test the algorithm. The output emphasised the accuracy of the algorithm. Two methods were used to evaluate the algorithm - interviews with programming teachers and focus groups with programming students. The results indicated the promising effect of the algorithm based on students' and teachers' satisfaction.

The next chapter summarises all significant findings of this study to help teachers and practitioners build an effective peer assessment activity for their courses and to encourage further research in this area.

## **Chapter 8. Discussion of findings**

### **8.1 Introduction**

This research set out to explore the benefits and best approaches for integrating a peer assessment activity into introductory programming courses as a learning process. In the previous chapters, the results were displayed in two phases. In the first phase, quantitative data were presented, while qualitative data were presented in the second phase. In this chapter, attention turns to the interpretation of these results. The research questions are addressed throughout this chapter, and significant findings are summarised and connected to those of previous studies in the same field. Appropriate recommendations for programming teachers and researchers are made. This chapter makes several contributions to the existing literature. Firstly, it provides an in-depth examination of first-year programming students' experiences and attitudes toward peer assessment as well as teachers' perspectives on this activity in introductory programming courses. Moreover, it provides evidence of the impact of peer assessment on students' learning performance. Furthermore, it enriches our understanding of what factors influence students' engagement with peer assessment and their critical issues. Finally, it provides a structural model for teachers and practitioners to follow as they implement peer assessment with first-year students.

### **8.2 Discussion of the findings in relation to the research questions**

This section contains a summary and discussion of the key contributions of this thesis in relation to the existing literature in terms of its findings and the methodology employed.

#### ***8.2.1 The perceptions of students and teachers on peer assessment***

The first research question asks: How do programming students and teachers perceive peer assessment in introductory programming courses? Before implementing peer assessment, it was necessary to identify students' and teachers' receptivity to peer assessment activities by asking their viewpoints. Understanding their views and the conditions under which peer assessment can be conducted was valuable to build a peer

assessment activity successfully. This question takes into consideration students' and teachers' perspectives on the benefits and challenges of peer assessment, and their position regarding key elements of peer assessment.

### **Findings: Most common benefits of peer assessment in programming assignments**

Although most programming students had not participated in peer assessment activities before, they were generally positive about peer assessment and found benefits in the context of introductory programming courses. They showed a high degree of willingness to participate in peer assessment as they viewed it as valuable for learning how to program. The analyses (see Chapter 4, section 4.2.2) showed that peer assessment offers benefits to first-year students in a programming education context. Many benefits appeared in the data collected in this study, such as improved knowledge and skills development, supporting the learning process, and improving code quality. However, the most common benefits mentioned in the literature differed from the most common benefits of this study. For example, the most common benefit of peer review in the systematic literature review was improved knowledge and skills development (Indriasari, Luxton-Reilly and Denny, 2020). While, in this study, the greatest benefit, according to programming students, was comparing their solutions in programming assignments with other solutions, as Trytten (2005) also observed. These differences may depend on the sample from which the data were taken or students' previous experience in peer assessment, as most participants in this study did not have previous experience in peer assessment. Therefore, more studies are needed to explore the most common benefits of peer assessment. This provides an insight into the areas that teachers have to expand on and create interest in to motivate students to assess their peers in programming courses. It also helps avoid making any changes to benefits that students noticed. Additionally, social benefits of peer assessment for students were mentioned in the literature (Sondergaard, 2009); however, sampled students did not mention any social benefits in this study. It seems that students are not aware of social benefits of peer assessment in comparison to academics or researchers who have this awareness. Besides, it can be argued that the sampled students benefitted from being reviewers



rather than from receiving feedback which is supported by current research (Nicol, Thomson and Breslin, 2014). This might be because being a reviewer activates cognitive processes and then develops the ability to make assessment judgements. In general, students hold a positive view of peer assessment, and they believe peer assessment has benefits when used in programming assignments.

Teachers' perception of peer assessment in programming assignments is significant because it can determine the intended learning outcome of a peer assessment and can thus define the overall purpose or goal of integrating this educational activity in programming courses. From the teachers' perspective, the most common benefit that emerged was the advantage of seeing alternative solutions for the programming assignments. Furthermore, many teachers thought that peer assessments make students into active learners. They also considered that students' confidence can be increased by providing feedback to their peers. Accordingly, the collected data confirmed that peer assessment improves students' learning and skills, and supports the collaborative environment. The literature has not shown programming teachers' perception of the benefits of peer assessment, but has appeared in other disciplines. Several studies in different disciplines observed a positive perspective toward peer assessment and found many benefits, such as reduction in marking time, improved student satisfaction with feedback, and making students more responsible (see, e.g., (Koc, 2011; Atkinson and Lim, 2013; Panadero and Brown, 2017)). Furthermore, the data collected found that programming teachers did not think that peer assessment can improve the quality of the code product; in fact, they suggested that it does not add much to improving code. This is contrary to Hundhausen *et al.*, (2009) who found that reviews between peers improved the quality of students' codes. As a result, programming teachers' view of the benefits of peer assessment focuses on learning rather than assessment.

Seeing more than one solution to the same task - as programming tasks are likely to have several solutions - was thus seen as the biggest benefit of peer assessment by both teachers and students. The next section discusses the most common challenges in peer assessment activities for programming courses.

## **Findings: Most common challenges of peer assessment in programming assignments**

Negative perceptions of peer assessment were also evident in this research (see Chapter 4, section 4.2.2). The most dominant view was that some of the students would not be capable of assessing their peers' work. The students were convinced that there were classmates who were able to assess well, and some who were not. Patchan and Schunn (2016) emphasised this viewpoint in terms of reviewing writing tasks. This issue can be mitigated by using rubrics that guide students during the assessment. Another common concern students raised was the fear of objectivity (e.g., too harsh, lenient, or biased) and the credibility of their peers' assessment which can be mitigated by anonymising the peer assessment activity. This issue has also been reported in other disciplines such as the medical field (Papinczak, Young and Groves, 2007), engineering design (Nicol, Thomson and Breslin, 2014), and management, entrepreneurship and human resources (VanSchenk Hof et al., 2018). Another problematic issue from the students' point of view is the risk of providing their peers with incorrect or misleading feedback during their peer assessment because they are not yet qualified. This has also been raised in studies with students studying English as a foreign language (Yang and Meng, 2013) and with medical students (Papinczak, Young and Groves, 2007). Here, it is clear that the goal of peer assessment is not to obtain feedback at the level of the teacher's assessment, but rather it is a formative exercise that helps to develop and practise a high level of thinking. Students in higher education have the ability to assess their peers with reasonable accuracy, consistency and without bias when the assessment does not count towards the final grade, as Sridharan, Tai and Boud (2018) stated. Consequently, there seemed to be no significant issues that should prevent the use of peer assessment with first-year programmers, and all the problems that concerned the programming students were also problems that concerned students of other disciplines.

The teachers who participated in this study highlighted several challenges to the use of peer assessment. The most common challenge was that not all students had the ability to assess their peers' work in programming assignments. Although teachers were also

convinced that there are students who are able to assess very skilfully. This indicates that the assessment ability differs between programming students; some students can assess their peers' work very well, while others cannot. Similarly, Huisman *et al.* (2018) found that the ability to assess peers was related to assessors' performance in the subject; i.e., the higher the ability to assess peers, the higher the ability in an essay performance in the subject. Therefore, and considering these perspectives, the relationship between the reviewers' ability and the authors' ability was considered in this study to address this issue. The suggestion this study makes is to match authors and reviewers based on their abilities to avoid such challenges in peer assessment. Another barrier mentioned by teachers was the low engagement from students. This point was discussed with first-year students, and the students clarified the need for rewards to encourage them to participate in peer assessment. In this study, other challenges, such as reliability and validity issues, lack of expertise, and time and resource constraints were also discussed. These barriers are also prominent in other disciplines (Adachi, Tai and Dawson, 2018), but peer assessment remains successful and popular. Furthermore, this study has found that a) multiple reviewers can reduce the validity issue; b) using rubrics counter the lack of experience; and c) using online platforms can reduce the time execution problem. Consequently, no significant barriers have been found that prevent teachers from using peer assessment in introductory programming courses.

These two findings have contributed to the discussion of the benefits and challenges of peer assessment in introductory programming courses from the perspective of first-year students and programming teachers. The consideration of the benefits was to provide insight into the areas that teachers should pay attention to in order to encourage the use of peer assessment. The considering of the challenges aimed to find solutions to these issues. From these findings, it appears that both students and teachers believe in the benefits of peer assessment for first-year students. Teachers and students also have concerns; the biggest concern was that not all peers can assess programming tasks; this study, however, offers a solution for this problem, i.e., peer matching. As a result, implementing peer assessment in introductory programming courses for first-year

students is a decision that should be taken seriously, as other disciplines benefit from peer assessment despite facing the same challenges.

### **Findings: Best practices for implementing peer assessment**

The main elements of peer assessment in introductory programming courses were identified based on the results of the first phase of this study (see Chapter 4, section 4.2.5). The study considered stakeholders' opinions of these elements in the context of programming courses specifically. Thus, it helps to establish the base of an effective peer assessment activity in introductory programming courses for practitioners or for further research, and makes the available evidence more accessible to them.

#### **1. Finding: Building a formative peer assessment activity**

Formative assessment was selected as the approach for this study. Most of the teachers emphasised that if peer assessment was used in introductory programming courses, it should be a formative assessment strategy to encourage students to comment on the work of their peers without the pressure of losing marks. Most of the students also preferred to assess peers by providing feedback on their solutions without giving a grade; they mentioned that being assessed without assigning scores was an interesting idea. Several previous studies have similarly used peer assessment in programming courses formatively (Shui Ng, 2017; Luxton-Reilly, Lewis and Plimmer, 2018; Sun *et al.*, 2019). This thought drives the formulation of a well-known rule: "assessment drives learning", which means an assessment that creates feedback which is then used to improve students' performance should be adopted. Thus, this finding recommends that researchers and practitioners adopt this approach and conduct more studies on frequent formative assessment, particularly in the early weeks, to investigate its impact on first-year students' performance and dropout decisions.

#### **2. Finding: Developing a marking scheme for peer assessment**

Marking schemes are viewed as an essential requirement for peer assessment. The results from the different methods used in this study all pointed towards the importance of a marking scheme with assessment criteria to help first-year programmers make

decisions and justify their feedback, as well as to remind them what they must complete for the assessment. The marking scheme is most suitable for those who do not have significant background knowledge, such as first-year programmers, as criteria can guide students when assessing peers. They can help students understand the learning objectives and the required standards of quality for a particular task in programming courses and help them make judgments about their own and their peers' work. Many studies on programming have used marking schemes in peer assessment (e.g., Sitthiworachart and Joy, 2004; Hamer *et al.*, 2009; Turner, Pérez-Quiñones and Edwards, 2018). This indicates that the use of marking guidelines, clear marking criteria, and appropriate marking scales are good practices to improve the process of peer assessment in computer programming.

However, studies that have used marking schemes differed in their marking guide models, especially in detail and specificity of scales; some studies used a simple list of criteria without detailed levels but with Likert scales instead (Sitthiworachart and Joy, 2008; Shui Ng, 2017). Another study employed a set of criteria with several descriptive levels and detailed rubrics (Hamer *et al.*, 2009). The pilot-experiment method was conducted (see Chapter 4, section 4.4.2) to investigate an effective scale that can guide first-year programmers during the peer assessment. In this study, two marking guides were created, differing in their scales and level of detail. Using this method showed no statistically significant difference for the students between using a rubric with descriptive details in the scale or a marking scheme that used a normal Likert scale in terms of their accuracy in peer assessment. Originally, it was thought that the rubric form (descriptive details in the scale) would standardise the results more clearly than the marking scheme form. However, during the experiments, the rubric took longer to read and prolonged the decision-making process, and many students expressed their preference for the marking scheme, perhaps because they were unfamiliar with the use of the rubric form in programming courses. Although there was no significant difference, the mean score for the marking scheme was higher than the one for the rubric. Lopez-Real and Chan (1999) recommended that it is preferable to use rubrics that are simple as possible; in contrast,

Miller (2003) argued that more detailed rubrics facilitated better quantitative judgment of performance. Moreover, Cateté, Snider and Barnes (2016) claimed that detailed rubrics were more suitable for non-experts (e.g., teaching assistants) for assessing programming assignments. As a result, the marking scheme focused on concepts, whereas the rubric focused on the scale and making judgments; thus, tutors can select the appropriate form based on their aims. As this research targeted first-year programmers and formative assessment, the marking scheme developed in this study was selected to focus on concepts rather than making judgments.

Most of the students agreed in their responses to the questionnaire that the teacher should involve them in creating the marking criteria before starting the assessment process. In contrast, teachers' opinions about allowing students to have input when creating the rubric varied during the questionnaire and interviews. The majority emphasised that typically a teacher was responsible for providing a rubric and assigning expectations for each level in the scale. Criteria in rubrics are standards used to measure whether or not the objective has been achieved and specifies how well the student performs the behaviour. Thus, teachers are normally responsible for translating assignment objectives as criteria for measurement. Accordingly, in this study, teachers were responsible for creating criteria in the marking scheme. Fraile, Panadero and Pardo (2017) similarly found that co-creating assessment criteria does not affect students' performance compared to others who did not participate. They suggested that students may need to get an opportunity to discuss the assessment criteria with their teachers instead of having a rubric imposed on them. Therefore, it could be better for teachers to create criteria related to the assignment's objectives, then discuss them with their students to establish expectations before assigning the programming tasks.

### 3. Finding: Anonymity in peer assessment

Teachers and students both preferred anonymous forms of assessment, as shown by the results of the quantitative and qualitative research methods in this study. Anonymous peer assessment seems to offer advantages for students in terms of its learning value and the credibility of the assessment. For instance, most of the students who completed the

questionnaire expressed a desire for peer assessment to be anonymous. They argued that it increased the credibility of the peer assessment and reduced its misuse, hostility, and bias. In the focus group discussions, some of the students requested that the peer assessment process remain anonymous, as they felt more comfortable and positive with their anonymity preserved (see Chapter 6, section 6.2.3). Students studied by Sitthiworachart and Joy (2004); Li and Gao (2016); and Rotsaert, Panadero and Schellens (2018) also emphasised this viewpoint. Furthermore, teachers in this current study suggested that anonymity makes the author and reviewer focus on analysing the task rather than the person behind it. Similar views have been expressed elsewhere (Panadero, 2016; Li et al., 2020; Harris, 2011). Although this study used peer assessments as a formative assessment, the data concluded that providing anonymity is an effective way to create a safe peer assessment setting and high quality of peer feedback.

#### 4. Finding: Online peer assessment

Students preferred online peer assessment because they think that the anonymity provided by the online environment makes them more comfortable with judging their colleagues objectively and without bias. Conducting online peer assessment allows the identities of the authors and reviewers to be anonymised easily through usernames. It also allowed students to do peer assessments anytime and anywhere. Teachers also preferred the online peer assessment, but some of them mentioned that a face-to-face activity could create a collaborative discussion environment. Moreover, as teachers complained of lack of time, they considered it one of the barriers to implementing peer assessment, which encouraged the use of an online peer assessment in this study. Similarly, several other studies have emphasised that computer-mediated peer assessment works well with students (Hundhausen, Agrawal and Agarwal, 2013; Li *et al.*, 2016, 2020). Furthermore, an online peer assessment application can help provide data to support learning analytics. This study employed two aspects in learning analytics: visualising peer feedback (see Chapter 6, section 6.4) and adapting peer assessment by matching authors and reviewers (see Chapter 7, section 7.8). This encourages

researchers and practitioners to use online peer assessment applications, take advantage of the available data, and employ learning analytics to improve the overall quality of learning.

#### 5. Finding: Evaluating peers' work individually

This study found that students and teachers preferred individual evaluations to assess peers rather than collective discussions. Teachers were more in agreement than students. Teachers thought that assessing peers' code in a quiet environment and at their own pace would increase the chance of students understanding the work they are assessing. Additionally, teachers stated that individual reviews avoided the possibility of "freeloading" off peers so that all students become fully invested in the peer assessment process. However, Hundhausen, Agrawal and Agarwal (2013) found that a combination of individual and collaborative reviews has more pedagogical benefit. And Sitthiworachart and Joy (2004) found that group discussions are an important factor in peer assessment. This study used an individual peer assessment - without accompanying discussions between the reviewers - as this was indicated as the participants' preferred option. However, other reviewers' feedback appeared after reviewing the task to make reviewers compare their assessment with other reviewers' assessments. It is possible that the wording of the question in the questionnaire influenced the participants' answers as they could only select either individual assessment or group assessment, but students may prefer combining individual and group discussions in peer assessment. Further studies should be conducted to determine the effectiveness of discussions between reviewers after individual evaluations.

As a result, peer assessment in this study is related to learning, not assessment. Langan *et al.* (2005, p. 31) state that "benefits of learner inclusion and active learning dimensions merit [peer assessment] inclusion in future courses". Overall, the student and teacher feedback can be viewed as positive, with many participants expressing a liking and satisfaction with the inclusion of a peer assessment model as part of course activity. Despite the potential challenges when applying peer assessment, the pedagogical and practical arguments for incorporating it into introductory programming courses are clear



and strong, particularly in courses that have multiple solutions for one single assignment. In addition, the student' and teachers' questionnaires generated a range of student perspectives and commentary about peer assessment main factors. The ones that were mentioned in this section are: the assessment must be formative; using a simple marking scheme that focuses on the concepts; anonymity for the author and the reviewer; individual practice to linger in the assessment process; and it can be online practice to employ computer technologies. With careful consideration to design and implementation, the 'learning from assessing' that results will make up for the effort made, and challenges may be encountered.

### ***8.2.2 Accuracy and impact of peer assessment on students' performance***

The second research question asks: Are first-year students who participated in peer assessment accurate and more likely to perform better in their programming skills than those who do not? The purpose of this question was to evaluate the validity of the activity by measuring the accuracy of first-year students' peer assessment ability, and to examine whether this activity contributes to improving learning in introductory programming courses or not. Other research has considered the validity of the activity by comparing the accuracy of the assessment given by the student with that given by the tutor or their peers to demonstrate that students are generally able to make reasonably accurate judgments. Scholars have also examined the impact of peer assessment by measuring the effect size of peer assessment on students' performance in different activities (e.g., homework). Thus, the existing literature on peer assessment has been dominated by empirical studies to examine the accuracy of peer assessment and measure its impact. Since this current study is also empirical in nature, the following sections interpret and explain the results from the experimental parts of the study.

#### **Findings: First-year students are accurate enough to conduct peer assessment**

An experiment to determine if there was a correlation between peer assessment grades given by students and grades awarded by the teacher was conducted. Results of student reviewers and teachers were similar at a moderately medium level (see Chapter 4, section

4.4.5). This result has been examined twice: in the pilot-experiment method, with a small sample size, to find the best marking scheme form, and in the pseudo-experiment method, with a large sample size, to find the correlation between teacher assessment and student assessment. This result does not depart significantly from the findings of Li *et al.* (2016), and Sitthiworachart and Joy (2008), who found that peer and teacher assessments tended to agree with one another at a moderate level, although their methods of data analysis were different due to the nature of the data collected. It should be acknowledged that the students in this study did not assign scores in the peer assessment activity; this study compared the peers' responses and the teachers' responses for each criterion to determine the students' scores. This is unlike other studies which have only compared the scores given by peers with those given by teachers. These results indicate that peer assessment is an acceptable assessment method for first-year undergraduate students in programming courses and that first-year programmers are close enough to the teachers' assessment. This is sufficient to ensure the benefits of peer assessment for first-year students.

The data collected also found differences between the two universities, PNU and Newcastle University, in the assessment. There were different scores among the two universities' students for each category. For example, PNU students scored higher in correctness when they used the marking scheme. The frequencies of the Newcastle University students' responses were analysed in terms of the correctness criteria, most Newcastle University students chose '*I don't know*'. When the interviewer returned to the tutors, they said, "*Newcastle University students are unfamiliar with assessing code on paper and generally use computers to assess*". On the other hand, students at PNU were familiar with paper questions and writing code on paper during exams, so they obtained higher mean scores in the correctness category. Meanwhile, Newcastle University students were awarded higher scores in the layout and clarity categories, and PNU tutors explained that "*layout is not required during exams*". In the same vein, Sitthiworachart and Joy (2008) conducted a peer assessment activity three times with the same students, and they found differences in the detailed scores of categories in each assignment. This

indicated that it is difficult to predict the categories that students' assessments are closest to tutor assessment. This is because the similarity of students' assessment to that of their teachers may depend on the difficulty of the assignment, the focus of the teacher during the explanation, and the students' characteristics. As a result, there is no category or criterion in which programming students can always be aligned with a teacher assessment in programming assignments based on these results.

### **Findings: Impact of peer assessment on students' performance**

In view of the positive impact of peer assessment, the present study attempted to measure the impact of this approach on enhancing the learning effectiveness of an undergraduate introductory programming course using experimental method (Chapter 4, section 4.4.6) that conducted a peer assessment activity - as an external activity - between two midterm exams. To assess the effect of the peer assessment activity on the programming course scores, the mean of the second midterm exam score for students' who participated was compared with those who did not participate in the peer assessment activity. The data found a positive effect of peer assessment on students' academic performance in the mid-term exams in computer programming courses, with a large effect size. The large effect size indicated that first-year students improved their performance in programming more when they engaged in peer assessment than when they did not engage in the peer assessment activity. This result can be aligned with prior research findings. For example, Li and Gao (2016); Shui Ng (2017); and King (2018) found enhancement of students' learning outcomes in programming skills. This study was distinguished in measuring the effect of peer assessment on a large sample size of participants, and conducted a peer assessment activity between two mid-term exams (official measurement). Furthermore, the data in this study measured how participants performed when repeating the peer assessment activity; the findings suggest that repeating the peer assessment activity could improve the quality of the assessment they provided, as also reported by Brutus, Donia and Ronen (2013). The finding supports the position that peer assessment can be an influential activity for enhancing academic performance compared to no peer assessment, which often involves teaching as usual. However, further studies are

required to measure the impact of peer assessment on other contextual factors, including various types of feedback, and educational characteristics (e.g., learning styles or students' skill levels). Assessing other aspects that impact peer assessment increases the validity of the existing findings.

Despite what the results of this study indicate, accurately measuring the impact of peer assessment on first-year programmers is categorised as a limitation in this study. The participants who performed better after peer assessment may have been positively influenced by other factors and experiences. Therefore, the assumptions made here are based on their feedback in addition to their performance scores and the effect size that was found in the statistical results with a group of 170 participants. In a full class where teachers implement peer assessment directly in a module, they apply peer assessment to all the cohorts and measure its impact in terms of student feedback in addition to scores. This is because many factors can influence student performance and their scores in an assessment. Peer assessment may have a significant impact on student performance, as the statistics have demonstrated in this study, but further qualitative work is required, and different contextual factors could be measured in order to explore this more extensively.

### ***8.2.3 Requirements and critical issues during implementing peer assessment***

The third research question was: What are students' requirements and critical issues related to implementing peer assessment in introductory programming courses? Since existing literature has not clarified whether programming students have specific needs regarding peer assessment, adopting qualitative methods – focus groups and interviews – to determine students' requirements and their concerns was an important way to assess the requirements before developing a prototype model for peer assessment. The research question aimed to establish users' needs as well as reduce the implementation cost. With prototyping, designers can determine early what the stakeholders want with faster and less expensive software. This type of study is referred to as student-centred research to increase student engagement. Therefore, qualitative methods in this study (see Chapter 6, section 6.2) provided crucial details about implementing peer assessment with first-year students. The sampled students in the focus groups suggested many features that

students would like to see in the peer assessment process, and they also mentioned some concerns they had about peer assessment. The following sections outline significant findings in this regard.

### **Findings: When and where to use peer assessment**

The data collected in this study suggest that peer assessment is most effective when students are novices. Students stated that they did not wish to use peer assessment in all situations, but rather use it according to their needs. They said, for instance, that they would like to implement it until they become proficient at programming. Also, they mentioned that they want to use peer assessment if there is more than one solution to a problem, or when the task was particularly difficult. Consequently, programming students preferred to use peer assessment when they were working towards transferring to the next phase of cognitive development. It should be implemented when the students are in the Zone of Proximal Development, i.e., when they can develop their knowledge by connecting with more skilled and competent others (Li and Gao, 2016). From the standpoint of the Zone of Proximal Development, the core goal of education is to keep learners in their own zones as long as possible to achieve maximum learning gain (Li and Gao, 2016). This is confirmed by another study that demonstrated that pair programming works well when pairs are novices and encounter difficulties in programming tasks (Lui and Chan, 2006). The data in the current study suggest that it is better to use peer assessment as a learning activity when students face difficulties, even if they are in the early weeks of the introductory programming course, as the ZPD recommends constructing knowledge through collaborating with more capable peers.

In contrast, many teachers in this study concluded that peer assessment should be used in advanced programming courses rather than in introductory programming courses. Accordingly, many of them did not use peer assessment as a learning method with first-year programmers in introductory programming courses. The most common use of peer assessment has been in advanced computing courses, as Hundhausen *et al.* (2013) confirmed. This was because students in their first year face a diversity of challenges, one of which is studying a new subject, so they need time to understand its fundamentals and

to become familiar with the concepts and new knowledge. Furthermore, some teachers in this study believed that students on advanced courses were more accurate reviewers than those on introductory courses. However, researchers have found that peer assessment carried out on advanced-level courses is no more valid than that conducted on introductory courses (Falchikov and Goldfinch, 2000). And yet, peer assessment may help first-year programmers more as they are not starting to formulate their own approach yet – this is in comparison to those with more experience who have already developed their own style and confidence. A study found that pair programming with novice pairs is more productive than with expert pairs because novice students work with new solutions they have not encountered before (Lui and Chan, 2006). It is thus not necessary for students to have full knowledge to assess peers' work which is also what the current study found.

### **Findings: Students' requirements in peer assessment**

Analysis of the quantitative data indicated the main requirements students had of peer assessments, such as making peer assessment formative, using a rubric during the assessment, and so on. The following requirements are the result of the analysis of the qualitative data acquired in the second phase of this study.

#### **1. Finding: Make a self-assessment a pre-condition of peer assessment**

Many students who participated in the focus group suggested the need for a self-assessment as a pre-condition of a peer assessment. They mentioned that simultaneously applying self- and peer assessment encourages reflective practice, critiquing their own work, identifying their weakness, developing the ability to evaluate, and understanding criteria before assessing peers' work. In the same vein, other studies have reported that incorporating self-and peer assessment and feedback contributed to students' cognitive and affective development (e.g., Sadler, 1989; Lynch, McNamara and Seery, 2012). The role of self-assessment was not only to raise self-reflection and think about standards in this study though; it was also used as an indicator of the perceived level of difficulty of the task and helped pair students with each other in the peer assessment. The self-assessment data thus played a significant role in determining the

task's difficulty in relation to the students, and match students who did not find it difficult with those who did (see Chapter 7, section 7.8). As a result, it is recommended to incorporate self-assessment with peer assessment, because combining these two ways of feedback can help guide student learning and encourages reflection as these assessments allow students to navigate the learning process through the evaluation of themselves and their peers. For developers of peer assessment activities, the results suggest that utilising self-assessment data as a source of user profiles can help to personalise the learning process.

## 2. Finding: Displaying the feedback from peer assessment

Feedback is a crucial component of assessment for learning because it supports learners in gaining insight into their current position in the learning process and provides information on how to move from their current position to their desired position. One of the students' requirements determined in this study was the need for feedback from the assessment process, both as reviewers and authors. Authors need to display all the reviewers' feedback and compare their perspectives, and then the author can be selective to choose which feedback is acceptable and which is not. Reviewers also need to see how other reviewers assessed the same task, and compare their assessment with other reviewers' assessments. This finding was found to concur with those of other researchers (Lynch, McNamara and Seery, 2012; Li and Grion, 2019). Previous research recommends that giving and receiving feedback, as authors and reviewers, play separate roles in peer assessment. Therefore, when teachers aim to assess for learning, feedback should be seen as a key component to driving learning forward. Current research provided evidence regarding the process of using peer assessment and feedback to enhance the effectiveness on learning introductory programming courses in such a way that contributes positively to the learning outcomes of students.

One of the aims of this study was also to examine how peer feedback should be presented to students. Visual feedback for the author and reviewer were evaluated in this study, and students greatly appreciated visual feedback in the Peer Programmer prototype. The students stated that visualisation in peer assessment successfully directed their attention

to critical information and enhanced their comprehension of the feedback; further, visualisation supported their awareness and self-reflection. In the same vein, Shatri and Buza (2017) found that integrating visualisation with learning positively impacts students' motivation to learn, increases learning, and fosters critical thinking. However, there has been little evidence that these visualisations of peer assessment data could support awareness and metacognitive process, as Vieira *et al.*, (2018) claimed. Only two studies visualised self-and peer assessment data (Ueki and Ohnishi, 2016; Park *et al.*, 2017), and their results found a positive effect of promoting a higher quality of peer feedback. As a result, a significant contribution of this study is the activation of visualisation in peer assessment for the benefit of students; this visual feedback has been built based on students' perspectives, and the prototype visualised feedback for both author and reviewers. This study recommends measuring the impact of visualisation on students' motivation and engagement in peer assessment using quantitative methods.

### 3. Finding: Rewards for good peer feedback

Rewards generate interest which leads to effort in peer assessment. Many students underlined the importance of rewards in peer assessments, either tangible (e.g., bonus grades) or intangible (e.g., competitions between the best reviewers, or displaying the best reviewers' nicknames in a dashboard) to motivate them to carry out peer assessment. Through rewards, students' reviewers could show interest and increased participation in formative assessment activities and put more effort into the assessment; as for the author, he/she will increase the acceptance of criticism if the reviewer who got rewards assesses their work (e.g., by displaying nickname and star as a sign of rewards). Motivating students to engage in peer assessment was a challenge that instructors face when using peer assessment with their students; however, giving rewards could result in more effort because rewards tend to create a feeling of achievement among students and motivate them to be more productive. Topping (2010) raised the idea of offering rewards for good peer feedback, considering it as a variable that increased students' accuracy and merited more investigation. However, only a few studies reward the reviewers (Zheng *et al.*, 2019).



Consequently, this study recommends using appropriate rewards in peer assessment activities with students.

### **Findings: Suggestions to cope with critical challenges with peer assessment**

Students raised concerns about credibility issues which was evident in the qualitative and the quantitative data. Students do not trust all their peers to give them appropriate feedback, therefore they might not receive the desired benefits from peer assessment. Students discussed ways to improve the credibility of peer assessment and their receptivity to peer feedback. The following section outlines some solutions developed in this study to manipulate credibility issues.

#### 1. Finding: Pairing between students according to their abilities

The focus group discussions (see Chapter 6, section 6.2.3) showed that students who participated in this study doubted the efficiency of peer feedback because they believed that not all students can give them effective feedback, so they might not benefit from peer assessment. Many students believed that the quality of the feedback in peer assessment depended on the reviewers' abilities. Therefore, most of the students requested that only highly proficient reviewers should provide feedback on their work. This result is consistent with students' beliefs examined in other studies (Kaufman and Schunn, 2011; Patchan and Schunn, 2016), where students thought that feedback from high-ability peers was valuable. The most likely explanation for this is that higher-ability reviewers could have more knowledge about issues that can occur in programming tasks. This extensive knowledge can support higher-ability reviewers to identify more issues and do so more efficiently. On the other hand, lower-ability reviewers might tend to focus on handling errors, thus limiting their assessment to very localised changes that make very few modifications. Furthermore, low-ability authors might benefit more from receiving feedback from high-ability reviewers because their criticism might be more constructive, and they might suggest more solutions. Hence, in this study, the dissimilarity in the task difficulty level between authors and reviewers has been applied, as other studies have demonstrated (Hsiao and Brusilovsky, 2008; Patchan *et al.*, 2013). This study

recommends assigning reviewers in each peer assessment process according to the diversity in their ability, taking into account the dissimilarity between them, and following Vygotsky's theory which recommends collaboration with more capable peers.

## 2. Finding: Develop an algorithm that matches author and reviewers

As part of this study, the Balanced Allocation algorithm that allocates a group of reviewers to evaluate each author's work to enhance the credibility of peer assessment and show authors better-quality feedback was developed (see Chapter 7, section 7.8). The mechanism to match authors and reviewers was based on their abilities, considering dissimilarity in each assessment group. The algorithm first collects available scores of a specific assignment (e.g., self-assessment score, peer assessment scores, and teacher score) to decide the task difficulty level. Self- and peer assessment are used because these data are available and reliable (see Machado *et al.*, 2008). However, the current study assigns more weight to peer assessment scores than self-assessment, in accordance with Birjandi and Siyyari (2010). Based on this total score, students are then categorised into two groups: students who face difficulties in the task, and those who do not face difficulties. Then, the algorithm allocates a group of reviewers to each author based on the author's category, taking into account dissimilarity in matching; all matching groups are equal in the number of reviewers and balanced in their abilities level. Thus, the balance was achieved in the distribution of student assessors. The algorithm was evaluated with real and mock-up datasets as well as through feedback from students and teachers. In the interviews with programming teachers and focus group discussions with students, most of the participants strongly agreed with the need for such a tool to ensure the quality of the feedback. This indicates the importance of considering the selection of reviewers rather than using a random matching algorithm as peer assessment is adapted to students' needs and preferences to give students the chance to receive good-quality peer feedback. As a result, the findings recommend examining the algorithm in a real environment that conducts a peer assessment, then measure the impact of using such algorithm on the effectiveness of the quality of feedback in peer assessment.

### 3. Finding: Several reviewers should assess the assignment

Many students raised concerns over validation issues as they discussed ideas for validating a peer assessment – e.g., multiple reviewers should assess the same assignment to help students receive as much relevant feedback as possible (see Chapter 6, section 6.2.3). The students stated that the validity of a peer assessment might increase with the ability of multiple peers to produce an overall assessment. Many studies have confirmed the advantages of multiple reviews (e.g., Topping, 1998; Kaufman and Schunn, 2011). The Balanced Allocation algorithm assigns three reviewers with different abilities to each assignment. In the same manner, many scholars found that the reliability and validity coefficients were high when three or four reviewers assessed an individual's work (Falchikov and Goldfinch, 2000; Sung *et al.*, 2010). The assessment of other reviewers must be presented to a specific reviewer after reviewing the task in order to allow reviewers to compare their assessment with other reviewers' assessment; this should happen after submitting the assessment, to avoid affecting the assessment of others. This is what this finding suggests to practitioners and teachers: assigning multiple reviewers to assess a code and then display each reviewer's perspectives after completing the assessment benefits both parties, authors and reviewers.

As a result, this question desired to collect first-year students' requirements and their critical issues around implementing peer assessment. These requirements led to implementing the Peer Programmer prototype website. The significance of this result lies in its attempt to integrate peer assessment activities into introductory programming courses based on students' needs, in what activities they want to integrate it, how they want to implement it, and how to avoid their concerns.

#### ***8.2.4 Integrating peer assessment into introductory programming courses***

The fourth research question asked: How can a peer assessment, as a learning process, be integrated into introductory programming courses? This question aimed to develop a prototype website with suggestions for peer assessment tasks that meets programming students' and teachers' requirements and avoids their concerns. The Peer Programmer

prototype has been built, developed incrementally, and evaluated. Although the website was a prototype, the website allows users to explore and interact with many tasks as real tasks. The Peer Programmer prototype was divided into three parts. The first part allowed the students to complete the assignment and assess themselves; the second part allowed the students to assess multiple peers' work, personalised based on their needs, using the rubric anonymously; and the third section displayed visual feedback from the peer assessment (as an author and a reviewer), and then allowed them to edit their work and rate the reviews they have received. The teacher can also add a new assignment, manage the assignment by building marking scheme criteria, adjust the assignment settings, and assign the assignment grade. The website architecture was logical to users and allowed users to follow their own personal journey, having different paths depending on their roles. Participants reported that they found the Peer Programmer prototype supportive, and they will benefit from it in future programming assignments. Meaningful users' involvement in this co-design process and their evaluation led to an intuitive and functional prototype.

In this section, a structure was suggested that could guide programming teachers and designers when implementing a peer assessment in programming courses. The structure contains eighteen elements of peer assessment. Elements are grouped in three clusters: 1) preparation phase; 2) implementation phase; and 3) reflection phase. Each combination of elements in the different variables results in a different type of peer assessment. The suggested structure could guide programming teachers and designers when implementing a peer assessment in programming courses. Teachers need a structure that allows them to support their students when applying a peer assessment so that teachers are not overwhelmed, and have the support they need to achieve their goals using such an activity. Designers, on the other hand, need to understand the requirements identified by stakeholders to build an effective peer assessment system that fit users' needs. Thus, the following structure helps to provide a comprehensive understanding of the peer assessment integration in programming courses. Some of these elements in the suggested structure have been discussed in previous sections, but combining these elements under one subject helps to establish the base of an effective way to integrate

peer assessment activity in introductory programming courses for practitioners or for further research, and makes the available evidence more accessible to them.

Table 8.1 shows the structure for integrating peer assessment in programming courses. The first column clarifies the phases of applying peer assessment in the course. The second column defines the variables under each phase, the third column shows the range of variances to each variable, and the last column recommends the fit variable in each element according to programming students' and teachers, as this study is specialised in the subject of programming. Although a list of elements will never be exhaustive, this study was able to suggest several important variables that benefited from adding technologies to the activity. The following section clarifies the elements of each phase.

<b>Phase</b>	<b>Variable</b>	<b>Range of variation</b>	<b>Study's recommendation</b>
<b>Preparation phase</b>	1) Product	Assignment? Or project?	Programming assignments.
	2) Objectives	See alternative solutions? Time-saving? Enhance learning?	Active participation Provide multiple solutions to each task. Provide instant feedback to each student.
	3) Type	Summative or formative?	Formative assessment.
	4) Standards used	A simple list of criteria? Rigid criteria? Or commentary feedback?	Marking scheme that contains simple criteria and open-ended questions.
	5) Time	In the beginning or at the end of the semester? Within lecture or lab sessions or through an online platform?	In early weeks through an online platform, so assessment conducted anytime and anywhere.
	6) Outcomes	Ability to critique code, improve programming skills, or develop teamwork skills	Students will be able to critique and make a judgment of programming codes.
<b>Implementation phase</b>	7) Calibration standards	Discussion with students? Or co-create with students?	Discussion with students.

Phase	Variable	Range of variation	Study's recommendation
	8) Self-assessment	Before peer assessment or after?	Prerequisite prior to peer assessment.
	9) Privacy	Anonymous or public?	Anonymous.
	10) Peer configuration	Individual or team or both?	Individual assessment.
	11) Peer matching	Based on task difficulty? Or preferences?	Matching based on task difficulty level.
	12) Multiple reviewers	Two or three or four reviewers in each assignment?	Three reviewers.
	13) Technology used	Automated peer assessment or using extra technologies (e.g., learning analytics)	Learning analytics (visualisation and adaptation).
<b>Reflection phase</b>	14) Official weight	Contributing to the final official score or not?	It does not contribute to students' official score.
	15) Feedback strategy	Feedback for reviewer or author? Visual feedback or written?	Different visual feedback for both author and reviewer.
	16) Set rewards	Course credit or competitions between the best reviewers?	competitions between the best reviewers.
	17) Editing work	Allow to edit student's own work or not?	Allow revision assignment.
	18) Personalised process	A number of reviewers? Characteristics of reviewers?	Matching based on students' characteristics.

**Table 8.1. A structured form for integrating peer assessment**

### **Preparation phase**

Preparation means getting ready and planning with a goal in mind. It includes all advance preparations made to get the peer assessment ready for implementation. It includes six elements:

### 1. Determine the product to assess

Possible products to assess in programming courses are either programming assignments or projects. The teacher should select the product to be evaluated (examples of programming assignments are described in Sitthiworachart and Joy, 2003; Hamer *et al.*, 2009; and projects in Ng and Fai, 2017). This study suggests assessing programming assignments because programming assignments are usually assigned to students weekly; they contain questions that can have more than one solution and can be formative tasks, therefore do not affect students' grades.

### 2. Determine the objectives of the activity

Teachers should identify the desired objectives of the peer assessment to set shared expectations between students and them (Wilson, Diao and Huang, 2015). Identifying the objectives of a peer assessment provides a guideline to design the activity, develop the appropriate tools and perform the activities effectively. Teachers, in this study, recognised their desired objectives as, for example, seeing alternative solutions to programming tasks, activating students' participation in the learning process, and receiving instant feedback.

### 3. Type of assessment

Assessment practice varies in whether the assessment should be primarily for learning (formative) or of learning (summative). Teachers should decide the type of peer assessment based on their objectives before applying the activity. The finding in the current study strongly supports the formative use of peer assessment for first-year students as an opportunity for learning by providing and getting feedback. Thus, the relationship of the peer assessment to the teacher assessment should be supplementary because it formatively adds value, rather than being substitutional as a summative assessment.

### 4. Preparing the standards used

Rubrics include a considerable amount of variation for design and implementation (Dawson, 2015). Teachers should develop a rubric in line with the objectives of the activity.

For example, some rubrics include a simple list of criteria; some are requested to provide peer feedback; others include rigid criteria to allocate accurate marks. This study concluded using a simple marking scheme that focuses on criteria was most effective and easy to use by students. Besides, teachers should encourage students to provide different types of feedback information, as in the suggested marking scheme. For example, some criteria in the marking scheme were quantitatively related to rating (e.g., scale), while others were qualitative through commentary feedback.

#### 5. Execution time

Teachers should decide when they use a peer assessment, at the beginning of the semester or at the end? Is it used during a lecture or lab sessions, or through an online platform? How long will it take: 30 minutes or one hour? According to Liu and Carless (2006), time is one of the constrictions teachers face in peer assessment. The timing of peer assessment in programming courses was a controversial issue between students and teachers in this study. However, this study recommends applying peer assessment with first-year students in the early weeks and through online platforms; the peer assessment does not take long because it will apply to programming tasks.

#### 6. Determine learning outcomes.

Learning outcomes of peer assessment could be related to the course's aims, program, or institution (Van Den Berg, Admiraal and Pilot, 2006). Teachers should identify the intended learning outcomes of using a peer assessment; for example, the knowledge, skills, attitudes, behaviours, or values students should gain from the activity. Several teachers in this study have expressed a set of intended outcomes of the peer assessment in programming courses, such as the ability to critique code, improve programming skills, and develop teamwork skills.

#### **Implementation phase:**

The implementation phase includes putting the peer assessment plan into action. It involves seven elements:



## 7. Calibration standards

It is an action or process of calibrating marking criteria before using them to better understand the standards. Gielen *et al.* (2011) stated that calibrating marking criteria was a good way to not only to teach the criteria but also to check students' understanding of the criteria before students use a peer assessment. Teachers should make this pedagogical decision to provide an opportunity for students to discuss, ask questions, and check their understanding of the marking criteria before a peer assessment practice.

## 8. Combining self-and peer assessment

Teachers should use self-assessment with peer assessment as it has been argued that "there are close links between peer assessment and self-assessment" (Carnell, 2015, p. 1271). Some teachers include a self-assessment as a prerequisite to the peer assessment; others require it following the peer assessment. This study included self-assessment as a prerequisite; students assessed themselves before they completed the peer assessment. This showed that they understood the criteria before assessing their peers. The self-assessment data helped built individual user-profiles and then match them.

## 9. Privacy

Teachers should make the interaction between peers anonymous, as this study emphasises. Anonymity can occur in two ways; when reviewers assess their peers' tasks, they do not know who completed the task; and when authors receive peer feedback, they do not know who made the assessment. Providing anonymity is effective because it creates a safe peer assessment setting, but it does prevent students from experiencing two-way interactive feedback dialogues (Rotsaert, Panadero and Schellens, 2018). Online applications can offer tools for anonymous two-way communications (e.g., chatting).

## 10. Peer configuration

A peer assessment can be conducted individually between a pair of students, teams of students or a combination of both (Gielen, Dochy and Onghena, 2011). Teachers should decide the peer configuration: individual or team assessments, or both? This study

suggests using individual assessments because first-year students need time to trace a code and make judgments; further, this study gave insight into the reviewers' perspectives on the same task. There were no discussions between reviewers to avoid influencing the opinion of others.

#### 11. Peer matching

Teachers should decide the appropriate criteria for peer matching, and it should be based on learning theories. The way students are paired in a peer assessment is significant to improve feedback quality (Topping, 1998). Teachers can build the matching according to many factors such as students' ability, previous knowledge, or students' preferences. This study developed a tool that matches authors and reviewers based on the task difficulty level that has been decided using self-and peer assessment data and represented a high level of accuracy and satisfaction for the matching tool.

#### 12. Assign multiple reviewers

Teachers should employ multiple reviewers to assess a specific task to increase the validity of the assessment. If one reviewer does not assess accurately, there are alternative reviewers (Topping, 1998). Thus, authors can receive a number of ideas to improve the quality of their code - the Balanced Allocation algorithm assigned three reviewers with different abilities to each assignment.

#### 13. Technologies used

Teachers can use technologies to facilitate a peer assessment, as technology often influences assessments design (Bearman *et al.*, 2016). A wide range of online platforms can assist in a peer assessment; for example, PeerScholar (Collimore, Pare and Joordens, 2015), and PeerGrade (Graham, 2017). Using technologies has added advantages to the peer assessment activity; for example, visualisation added awareness and self-reflection, and adaptation improved the quality of the peer feedback.

## **Reflection phase**

The reflection phase is characterised by “learning through and from experience towards gaining new insights of self and practice” (Finlay, 2008, p. 1). Reflection allows a person to make connections between experiences, aiming to obtain maximum student progress. It involves five elements:

### **14. Official weight**

Teachers should decide whether the peer assessment contributes to the student’s official score or not. This matter moves peer assessment from a pure learning activity to a summative accountability evaluation (Panadero and Brown, 2017). When peer assessment counts as a part of the final grade in introductory programming courses, student scores must be credible. Therefore, this study checked the opinions of teachers and students, then avoided assigning weight to peer assessments.

### **15. Feedback strategy**

A peer assessment in conjunction with a feedback strategy improve the effectiveness of learning computer programming (Shui Ng, 2017). This study displayed peer feedback for both the reviewer and the author to support the learning process for each role separately. It used visualisation to represent peer feedback, and help students to become more aware of their abilities.

### **16. Set rewards**

Teachers should set rewards for students who excel in a peer assessment. According to Topping (2010), offering rewards for good peer feedback must be considered in peer assessment to increase reviewers' accuracy. This study supports the importance of using rewards in peer assessment. Rewards can confirm appropriate behaviour and motivate students to be more productive because they create a feeling of pride and achievement. Besides, students become more self-confident and responsible for their learning.

### 17. Editing the work

Since the peer assessment is a formative assessment, the author can edit his/her code and resubmit it again to the teacher after reading the reviewers' feedback. Further, reviewers can edit their own work when reading someone else's work (To and Panadero, 2019). However, the notification of resubmission should appear to the teacher. The prototype allows students to resubmit their work after the peer assessment process.

### 18. Personalising the peer assessment process

Adaptive learning analytics provide more effective learning experiences and opportunities that increase learning and student satisfaction (Sarıyalçınkaya *et al.*, 2021). The personalised peer assessment process allows students to engage more in the activity and explore their needs using tools and strategies that highlight their abilities. This study developed a personalised matching process tool based on authors' needs; this was regarded highly by the students in the focus groups.

As a result, this study developed a structured form that involved important elements of peer assessment that can be followed when integrating the activity in introductory programming courses. The Peer Programmer prototype has been built based on these 18 elements. This study recommends using this structure as a guideline to describe peer assessment practices in future studies. Further, the Peer Programmer prototype can be developed to become a real system as it represents all of these elements.

## 8.3 Summary

This chapter summarised the findings in relation to each of the four research questions that concerned the peer assessment in introductory programming courses. The first question identified students' and teachers' perspectives of peer assessment, and students' receptivity to the process. Understanding their perspectives and the conditions under which peer assessment can be conducted was invaluable as a first step in this study. The second question evaluated the accuracy of the peer assessment activity with first-year students to demonstrate those students are able to make reasonably accurate judgments. It examined whether this activity contributes to improving students'

performance in introductory programming courses or not. The third question determined the appropriate time to apply a peer assessment and discovered the factors that encouraged programming students to use the peer assessment activity continually. It also provided solutions to critical issues related to implementation of peer assessment. The final question led to a structured form containing all important peer assessment elements for teachers who intend to implement peer assessment in introductory programming courses. All these elements have been represented in the Peer Programmer prototype website and gained students' and teachers' approval. The findings of this study support earlier research that identified peer assessment as a fertile context for enhancing student learning, yet this study focused on introductory programming courses.

The following chapter summarises the outcomes of the work so far and details further work that can be carried out.

## **Chapter 9. Conclusion**

This chapter concludes the study by summarising the key research findings in relation to the research aim, as well as the research contributions. It also details the limitations of this study, and presents recommendations for further research or action.

### **9.1 Summary of the study and main findings**

In introductory programming courses, teachers are familiar with the students' failure, dropouts, and irregular attendance. For a long time, failure and dropout rates among programming students have been trending research topics among computer science education scholars. They have suggested many interventions that might affect the decision to remain in programming courses; they found that the active and collaborative learning approach for computing education is practical for first-year students. The involvement of students in the learning process is the foundation of the most widely accepted theory describing the developmental process of dropout and retention. In this vein, this study aimed to integrate peer assessment into introductory programming courses because peer assessment engages students in the learning process and generates deep learning, critical thinking, increased performance, and teamwork skills.

The literature review detailed the current knowledge of peer assessment, which included the definition of peer assessment and its types, methods, benefits, barriers, impacts, existing successful tools, and potential issues that may hinder its success. This information was reviewed in order to build a foundation of knowledge on peer assessment and identify debates and gaps in these aspects to give a clear picture of the state of knowledge on the subject, and how this thesis contributes to existing knowledge. It should be noted that the literature on peer assessments in introductory programming courses has gaps in the following points: measure the suitability of peer assessment in the programming courses; attention to the stakeholders' perspectives; their requirements and concerns, then, integrate this activity and getting stakeholders' approval. Research questions and research objectives were formulated accordingly.

After that, the thesis explained in detail the research methods used in the study, allowing readers to evaluate its validity and reliability. A mixed-method approach was used to answer the research questions and expand and strengthen the study's conclusions by combining qualitative and quantitative research components. The combination of quantitative and qualitative methods divided the research into two phases. The first phase, which comprised quantitative methods, investigated two research questions. The first research question identified students' and teachers' opinions on peer assessment in introductory programming courses. This thesis answered the question using the questionnaire method, which was concerned with finding out the stakeholders' perceptions and the readiness of students and teachers to apply peer assessment. Students and teachers were excited to apply peer assessment, and they stated their position on key elements in peer assessment. For example, they wanted the assessment to be formative, using a marking scheme form; they wanted it to be anonymous and not be part of their final grades. The first question's results expressed the benefits of inclusion of a peer assessment in programming courses. Further, it determined the main concerns in relation to peer assessment.

This thesis also answered the second research question, which measured the closeness of student assessment to teacher assessment. The experimental method was used, and the results showed that student assessment was close enough to teacher assessment to obtain the benefits of peer assessment. Besides, the statistical results also showed that the peer assessment positively affected the performance of the programming students. Quantitative methods provided a wide variety of numerical data from differing sources to be explored. These findings helped build a base for the peer assessment activity, which can support teachers and practitioners when they adopt peer assessment with first-year programming students.

Furthermore, expanding the qualitative methods was necessary for this thesis to answer the rest of the research questions. In the second phase, qualitative methods were employed to hear the voices of the stakeholders. This thesis answered the third research question by identifying students' expectations of implementing peer assessments in

programming courses, i.e., the factors they required and the problems they were concerned with. Using iterative focus group discussions showed that first-year students' have special requirements and critical issues. Students expressed their need for visual feedback from different aspects of the peer assessment process, their desire for self-assessment prior to peer assessment, and their need for rewards for conducting a peer assessment effectively. The students also expressed the need to assign reviewers to each task according to the author's need, and the need for multiple reviewers for each task to raise the validity of the peer assessment.

Additionally, one of the key concerns raised by students was doubting the efficacy of the peer feedback they receive because they think that not all peers can provide useful feedback. This thesis described a technique for author-reviewer matching that can be applied in peer assessment systems - a Balanced Allocation algorithm. The algorithm determines how difficult the task is for each author based on self-and peer assessment, then divides students into two categories: with or without difficulties. After that, a group of reviewers is reserved for each author based on the section to which he/she belongs, considering that all matching groups to each peer assessment process are balanced in their abilities level and the number of reviewers. The algorithm was examined with real collected and mock-up datasets. Furthermore, the algorithm was evaluated based on students' and teachers' viewpoints. The results showed a high accuracy and satisfaction for the Balanced Allocation algorithm.

The final research question that aimed to integrate peer assessment into introductory programming courses identified the 18 elements that constitute peer assessment in introductory programming courses. Programming teachers seeking to use peer assessment can follow recommendations in these 18 elements. Also, designers who seek to develop peer assessment systems can easily find functional requirements as set out by stakeholders themselves. Careful consideration of each can help them avoid assumptions about peer assessment, and encourage them to investigate alternatives. Further, this thesis developed the Peer Programmer prototype website that contains all these elements. The prototype was designed based on users' requirements. In this



prototype, two actors are represented: teachers and students. Each actor has an initial set of use cases. For example, the teacher can add new assignments, build marking schemes, adjust the assignment settings, inspect, and grade, and display the feedback from the peer assessment if needed. On the other hand, the student can complete the assignment, complete the self-assessment, complete the peer assessment, rate the assessment of a peer, and display feedback as an author and reviewer. As this study followed a user-centred design approach, the students and teachers co-designed, refined, and evaluated the prototype until they were satisfied with it.

The overwhelming majority of students had a positive attitude towards peer assessment when participating in this experiment; this was probably not as positive as their attitude towards teacher assessment, but positive enough to render the activity effective. However, teacher assessment is still the most desired type of assessment among first-year programmers, even when they are trained to use other types of assessment and feedback, such as peer assessment. This preference for teacher assessment was based largely on students' assumptions that some of their peers might not be as qualified as their teachers to assess and provide feedback due to their level of experience and proficiency in programming. As a result, peer assessment was not designed in this study as a substitute for teacher assessment but rather as a formative practice that can help first-year students in their own learning process by providing feedback to peers and analysing the feedback received by these peers. This study also suggested a solution to reduce the gap between the quality of tutor assessments and peer assessments by matching authors and reviewers in order to receive better peer feedback.

## **9.2 Research contributions**

This study contributes to the body of knowledge on peer assessment, filling a gap in the previous research in the context of introductory programming courses.

The first contribution of this study revolves around exploring students' and teachers' perspectives towards peer assessment in programming assignments; particularly, perspectives from programming teachers were absent in the literature. Many aspects of

peer assessment examined in this thesis had not been discussed previously but should be explored when implementing peer assessment; these include when to implement, how to implement, users' requirements, and their concerns. The study benefited from both the generalisable, externally valid insights of quantitative data, and the detailed, contextualised insights provided by qualitative data. This is deemed to be significant to improve data quality. Consequently, this study added to the current literature the perceptions of the main stakeholders (programming teachers and students) about peer assessment in introductory programming courses.

The second contribution is the design and evaluation of a suitable marking scheme form for first-year students. Most past studies have focused on criteria and categories of marking schemes, with a lack of consideration to quality scales. Therefore, this study designed two forms of marking schemes that differ in scale and detail of criteria, then examined the best for first-year students. The findings conclude that first-year students need to focus on the concepts of criteria rather than scales and making judgments. Thus, teachers should design a simple marking scheme form that focuses on the concept of criteria.

Another contribution of this study is the statistical investigation. This study provides a unique examination of the correlation between student assessment and teacher assessment by comparing teachers' choices with students' choices rather than by comparing their final grades, as has been the focus in other studies. The study found that programming students were accurate enough in assessing peers to benefit of the activity. Moreover, the study measured the impact of peer assessment on the students' performance with a large sample size compared to previous studies. As statistical data investigated in this thesis has demonstrated, peer assessment has a significant positive impact on student performance.

Importantly in this study, peer assessment was adapted to the developments of technologies by activating some aspects in learning analytics, through, for example, personalising peer assessment by matching authors and reviewers. The Balanced Allocation algorithm was developed and evaluated to match authors and reviewers based

on students' abilities and the need to improve peer feedback quality. Visualisations of feedback were designed and recommended to improve the self-reflection and awareness of students.

This thesis has built a structural model that programming teachers can follow when they intend to integrate peer assessment into programming assignments. This model contains three stages: preparation, implementation, and reflection. Each stage contains a set of elements. These 18 elements should be considered when defining, designing, implementing, or researching peer assessment in programming courses.

Finally, these contributions in regard to peer assessment in introductory programming courses are worth disseminating to benefit teachers who might be thinking about implementing peer assessment with their programming students. It also supports scholars to better understand peer assessment practices, and it supports software developers to design tools that are suitable for both teachers and students because it responds to users' requirements.

However, there are several issues relating to the research, as it did not always progress as expected. The next section outlines the most significant issues in this research.

### **9.3 Research limitations**

In order to address the research questions, a range of data were collected and a variety of analysis techniques were used. However, some limitations affected the findings of this study. An obvious limitation of the study are the questionnaires. The questionnaires were created in the first year of the research, so the items were preliminary, and they did not explore some concepts around peer assessment clearly; for example, details about combining individual and collaborative activity in a peer assessment. In addition, the frequency of the items in all the variables varied widely (e.g., participants who have experience in peer assessment, and who did not have experience, male and female, Saudi and British participants), which made it difficult to make comparisons or conduct tests between those items. However, the main aim of the questionnaire was to obtain programming students' opinions about peer assessment and details of how they wanted

to implement it. The essential data were collected; however, if more data had been available, the questionnaires would have produced more valuable results.

Another limitation was the uneven participation of student and staff volunteers. One of study's goals was to compare Newcastle University and PNU students to generalise the findings between the two different communities, and to ensure that the results were reliable. Because the researcher was studying most of the academic year in Saudi Arabia, the researcher was able to control student participation at PNU by encouraging students, contacting students, offering incentives, and re-doing the experiments several times to obtain the required sample. However, the researcher was not able to do any of this at Newcastle University, and the researcher often faced the issue of Newcastle students not wanting to participate in the research experiments. Also, during the data collection period, the COVID-19 crisis occurred, which affected the collection of data and prevented the researcher from visiting the UK for two years. Collecting data at Newcastle University has undoubtedly been affected by the pandemic. Even though the researcher's academic supervisor supported her by providing staff to conduct the experimental method and focus group method, and by providing incentives to students, the researcher continued to face student reluctance. This makes it difficult to generalise the findings to the wider context globally.

Another weakness in this study is the lack of male participants in some of the samples. This may cause some bias in the results. The reason for this weakness is that the PNU is a female university; there are no male students at all. Also, social constraints in Saudi Arabia meant that it was difficult to include male students, because they are taught separately, and this constraint had to be borne in mind when the data collection plan was designed. Therefore, a follow-up assessment that repeats the same activities with only male participants and compares the results of such an assessment with the results from this study could reveal many more insights.

Furthermore, one critical limitation of the study is the validity of the qualitative data. The study collected a detailed set of qualitative data that allowed an in-depth analysis of peer assessment with programming students in introductory programming courses. However,

the analysis of the qualitative data may have been influenced by the researcher's subjective opinion. The qualitative method focuses on specific, detailed, rich data and context but it can be subjective (Cohen, Manion and Morrison, 2012). This study does contain personal interpretations, and it must be acknowledged that the researcher had an influence on the research results presented here. Despite that, information was explained in detail concerning the data collection, coding, and analysis processes (see Chapter six, 6.2 and 6.3) in order to ensure that the research process was adequately transparent for the readers. Additionally, the researcher did the best to put aside any preconceived judgments about peer assessment activities in programming courses and the researcher remained open-minded, trying to prevent personal opinions and assumptions from affecting the research process. To avoid subjectivity in this study, the researcher was not associated with the Introductory programming modules, nor did she teach students or intervened in selecting participants during the data collection phase. The researcher attempted to play the role of a facilitator to encourage participants to reflect on their experiences instead of leading them to comply with her assumptions or impose her own view on them. Though there are always inevitable weaknesses in research, presenting the data as authentically and completely as possible was one of the study's aims, to avoid bias, and to not misinterpret participants' responses.

Also, as stated by Denzin and Lincoln (2005), a researcher of a multicultural subject such as the one examined in this thesis should describe their role in the study, how their situation may possibly affect the research design, and how their biases could influence the understanding of the data. When selecting samples for the study under discussion here, the researcher initially investigated Arabic students who come from a similar cultural background and have similar educational experiences, so it was easy to empathise with the participants' experiences, feelings, and attitudes. Also, participants communicated fluently in Arabic, which enabled the researcher to gain more insider information. The researcher used the same methods with English participants, and because the researcher had investigated Arabic participants already, the researcher was able to understand the challenges the UK participants were likely to encounter and raised questions that a non-

native speaker might not have considered. However, it is possible that a researcher might unintentionally neglect information that native researchers often take for granted or deem important. After all, Bryman (2008, p. 389) claims that “since measurement is not a major preoccupation among qualitative researchers, the issue of validity would seem to have little bearing on such studies”.

Finally, while the use of students’ views as a platform to understand their learning experiences has its restrictions, the evidence does reveal that it can be a worthy tool (Morgan, 2011; Arnot and Reay, 2007). Pedagogic research has greater meaning when students are engaged in the process (Lewis, Florian and Porter, 2007). Consequently, it was necessary to consider student opinions on peer assessment, and feedback was needed to understand how to improve this aspect of learning. Moreover, this type of study is still important for programming teachers who want to apply peer assessment; it provides guidance which is the overall intention of the study.

#### **9.4 Future directions for research**

Having investigated the use of peer assessment for first-year programming students in an introductory programming course, this study helped to partly answer the research questions, but it has elicited a number of issues that can be explored further in future research.

It is possible to involve a wider range of programming students in future studies, particularly by avoiding being gender specific. A possible solution to overcome such restrictions is to develop contacts with other universities that admit male students and find volunteer staff who can act on behalf of the researcher, in order to distribute questionnaires, interview students, and conduct experiments. Furthermore, in geographical terms, engaging more participants from Newcastle University could help to generalise the findings of the research and ensure that the computer science students community sample is represented. Researchers at Newcastle University - as an example of a UK university - can evaluate the impact of peer assessment on first-year students’

programming performance or ascertain their requirements then compare between two results.

Moreover, it would be valuable to investigate how first-year students interact and perform during peer assessment, which means adding more data collection tools such as observation and think-aloud protocols. Such tools would shed light on students' original performances during a peer assessment, and provide additional insight into how programming skills can be developed. Also, adding group discussions or chat sessions after individuals have assessed a peer might offer useful information and lead to an improved peer assessment.

In this study, an algorithm that matches the author to a set of reviewers was implemented and evaluated. In future, to determine the effectiveness of the Balanced Allocation algorithm, matching technique could be applied in a peer assessment activity with a group of participants (experimental group), and random matching with another group of participants (control group). A t-test could then be performed to determine the impact of matching on students and its effect on the quality of feedback. Moreover, several variables can be added to the algorithm (e.g., previous knowledge, students' preferences) in order to improve the performance of the matching process. Furthermore, another method in MCDA can be used based on the type of attributes collected, such as the analytic hierarchy process (AHP) if different types of attributes are available, or an artificial neural network (ANN) if fuzzy data are accessible.

Finally, the suggested prototype met the intended requirements, and the participants had a positive opinion of it. Therefore, it would be worthwhile developing this website fully and then evaluating the effectiveness of the peer assessment activity with a real class.

## References

- Adachi, C., Tai, J. H. and Dawson, P. (2018) 'Academics' perceptions of the benefits and challenges of self and peer assessment in higher education', *Assessment & Evaluation in Higher Education*, 43(2), pp. 294–306. doi: 10.1080/02602938.2017.1339775.
- Akubuilu, F. (2012) 'Holistic Assessment of Student's Learning Outcome', *Journal of Education and Practice*, 3(12), pp. 56–60. Available at: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.911.9038&rep=rep1&type=pdf>.
- Al-Ohali, M. and Shin, J. (2013) 'Knowledge-Based Innovation and Research Productivity in Saudi Arabia', in *Higher Education in Saudi Arabia: Achievements, Challenges and Opportunities*. New York, USA: SpringerLink, pp. 95–102. doi: 10.1007/978-94-007-6321-0\_6.
- Al-Sa'di, A. and McPhee, C. C. A. (2021) 'User-Centred Design in Educational Applications: A systematic literature review', in *2021 International Conference Engineering Technologies and Computer Science (EnT)*, pp. 105–111. doi: 10.1109/EnT52731.2021.00025.
- Aman, R. R. (2009) *Improving student satisfaction and retention with online instruction through systematic faculty peer review of courses*, PhD thesis. Oregon State University. Oregon State University. Available at: <https://www.proquest.com/dissertations-theses/improving-student-satisfaction-retention-with/docview/304974844/se-2?accountid=35481>.
- Anaya, A. R. et al. (2019) 'Automatic assignment of reviewers in an online peer assessment task based on social interactions', *Expert Systems*, 36(4), pp. 1–19. doi: 10.1111/exsy.12405.
- Anderson, G. and Arsenault, N. (1998) *Fundamentals of Educational Research*. 2nd ed. London: Routledge Falmer.
- Andrade, H. G. (2000) 'Using Rubrics to Promote Thinking and Learning', *Educational Leadership*, 57(5), pp. 13–18.
- Andreasen, M. M. and Hein, L. (1987) *Integrated Product Development*. IPU, Lundtofte.
- Arisholm, E. et al. (2007) 'Evaluating pair programming with respect to system complexity and programmer expertise', *IEEE Transactions on Software Engineering*, 33(2), pp. 65–86. doi: 10.1109/TSE.2007.17.
- Arnot, M. and Reay, D. (2007) 'A sociology of pedagogic voice: power, inequality and pupil consultation', *Discourse: studies in the cultural politics in education*, 28(3), pp. 311–325.
- Ashenafi, M. M. (2017) 'Peer-assessment in higher education – twenty-first century practices, challenges and the way forward', *Assessment and Evaluation in Higher Education*. Routledge, 42(2), pp. 226–251. doi: 10.1080/02602938.2015.1100711.
- Atkinson, D. and Lim, S. L. (2013) 'Improving assessment processes in higher education: Student and teacher perceptions of the effectiveness of a rubric embedded in a LMS', *Australasian Journal of Educational Technology*, 29(5), pp. 651–666.



- Bacchelli, A. and Bird, C. (2013) 'Expectations, outcomes, and challenges of modern code review', in *Proceedings - International Conference on Software Engineering*, pp. 712–721. doi: 10.1109/ICSE.2013.6606617.
- Baleni, Z. G. (2015) 'Online formative assessment in higher education: Its pros and cons', *The Electronic Journal of e-Learning*, 13(4), pp. 228–236.
- Balla, J. and Boyle, P. (1994) 'Assessment of student performance: a framework for improving practice', *Assessment & Evaluation in Higher Education*. Taylor & Francis, 19(1), pp. 17–28.
- Barak, M. (2017) 'Science Teacher Education in the Twenty-First Century: a Pedagogical Framework for Technology-Integrated Social Constructivism', *Research in Science Education*, 47(2), pp. 283–303. doi: 10.1007/s11165-015-9501-y.
- Barry, D. (2017) *Do not use averages with Likert scale data*. Available at: <https://bookdown.org/Rmaddillo/likert/> (Accessed: 2 November 2021).
- Başaran, S. (2016) 'Multi-Criteria Decision Analysis Approaches for Selecting and Evaluating Digital Learning Objects', *Procedia Computer Science*, 102, pp. 251–258. doi: 10.1016/j.procs.2016.09.398.
- Baxter, K., Courage, C. and Caine, K. (2015) *Understanding Your Users A Practical Guide To User Research Method*. Second Edi. Morgan Kaufmann Publishers.
- Bearman, M. et al. (2016) 'Support for assessment practice: Developing the assessment design decisions framework', *Teaching in Higher Education*, 21, pp. 545–556.
- Ben-Ari, M. (2004) 'Situated learning in computer science education', *Computer Science Education*, 14(2), pp. 85–100. doi: 10.1080/08993400412331363823.
- Bennedsen, J. and Caspersen, M. E. (2006) 'Abstraction ability as an indicator of success for learning Object-Oriented Programming?', *SIGCSE Bulletin*, 38(2), pp. 39–43.
- Bergin, S. and Reilly, R. (2005) 'The influence of motivation and comfort-level on learning to program', in *Proceedings of the 17th Workshop of the Psychology of Programming Interest Group, PPIG 05*, pp. 293–304.
- Birjandi, P. and Siyyari, M. (2010) 'Self-assessment and Peer-assessment: A Comparative Study of Their Effect on Writing Performance and Rating Accuracy', *Iranian Journal of Applied Linguistics*, 13(1), pp. 23–45.
- Black, P. and Wiliam, D. (2009) 'Developing the theory of formative assessment', *Educational Assessment, Evaluation and Accountability*, 21(1), pp. 5–31. doi: 10.1007/s11092-008-9068-5.
- Bloom, B. S. (1969) 'Some theoretical issues relating to educational evaluation', in *Teachers College Record.*, pp. 26–50. doi: 10.1177/016146816907001003.
- Booch, G., Rumbaugh, J. and Jacobson, I. (2005) *Unified Modeling Language User Guide*. 2nd Ed. Addison-Wesley Professional.

- Boud, D. and Falchikov, N. (2007) *Rethinking assessment in higher education*. Routledge London.
- Boud, D. and Soler, R. (2016) 'Sustainable assessment revisited', *Assessment & Evaluation in Higher Education*, 41(3), pp. 400–413. doi: 10.1080/02602938.2015.1018133.
- Boudia, C., Bengueddach, A. and Haffaf, H. (2019) 'Collaborative Strategy for Teaching and Learning Object-Oriented Programming Course: A Case Study at Mostafa Stambouli Mascara University, Algeria', *Informatica*, 43, pp. 129–144.
- Braun, V. and Clarke, V. (2012) 'Thematic analysis, APA Handbook of Research Methods in Psychology', *Research designs: Quantitative, qualitative, neuropsychological, and biological*. Washington, DC: American Psychological Association., pp. 57–71. doi: 10.1037/13620-004.
- Brown, N. C. C. et al. (2013) 'Bringing Computer Science Back into Schools: Lessons from the UK', in *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*. New York, NY, USA: Association for Computing Machinery (SIGCSE '13), pp. 269–274. doi: 10.1145/2445196.2445277.
- Brutus, S., Donia, M. B. and Ronen, S. (2013) 'Can business students learn to evaluate better? Evidence from repeated exposure to a peer-evaluation system', *Academy of Management Learning & Education*, 12(1), pp. 18–31.
- Bryman, A. (2008) *Social research methods*. 3ed Ed. Oxford ; New York: Oxford UP.
- Bucciarelli, L. (2002) 'Between thought and object in engineering design', *Design Studies*, 23(3), pp. 219–231. doi: [https://doi.org/10.1016/S0142-694X\(01\)00035-7](https://doi.org/10.1016/S0142-694X(01)00035-7).
- Butler, D. L. and Winne, P. H. (1995) 'Feedback and Self-Regulated Learning: A Theoretical Synthesis', *Review of Educational Research*, 65(3), pp. 245–281.
- Butler, S. and McMunn, N. (2006) *A Teacher's Guide to Classroom Assessment: Understanding and Using Assessment to Improve Student Learning*. San Francisco: Jossey-Bass.
- Canziba, E. (2018) *Hands-On UX Design for Developers: Design, prototype, and implement compelling user experiences from scratch*. Packt Publishing Ltd.
- Carbonaro, A. and Ravaioli, M. (2017) 'Peer assessment to promote Deep Learning and to reduce a Gender Gap in the Traditional Introductory Programming Course', *Journal of e-Learning and Knowledge Society*, 13(3). doi: 1826-6223 e-ISSN 1826-6223.
- Carless, D. (2011) *From Testing to Productive Student Learning: Implementing Formative Assessment in Confucian Heritage Settings*. New York: Routledge.
- Carless, D. (2013) 'Sustainable feedback and the development of student self-evaluative capacities', *Reconceptualising Feedback in Higher Education: Developing Dialogue with Students*, pp. 113–122. doi: 10.4324/9780203522813.
- Carnell, B. (2015) 'Aiming for autonomy: Formative peer assessment in a final-year undergraduate course', *Assessment & Evaluation in Higher Education*, 41, pp. 1269–1283.

- Carver, J. C. *et al.* (2007) 'Increased Retention of Early Computer Science and Software Engineering Students Using Pair Programming', in *20th Conference on Software Engineering Education Training (CSEET'07)*, pp. 115–122. doi: 10.1109/CSEET.2007.29.
- Cateté, V., Snider, E. and Barnes, T. (2016) 'Developing a rubric for a creative CS principles lab', *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, 11-13-July, pp. 290–295. doi: 10.1145/2899415.2899449.
- Celepku, M. and Boyer, K. E. (2018) 'The importance of producing shared code through pair programming', in *SIGCSE '18: 49th ACM Technical Symposium on Computer Science Education*. Baltimore, MD, USA, pp. 765–770. doi: 10.1145/3159450.3159516.
- Chatti, M. A. *et al.* (2012) 'A Reference Model for Learning Analytics', *International Journal of Technology Enhanced Learning*, 4(5–6), pp. 1–22.
- Chinn, D. *et al.* (2010) 'Study habits of CS1 students: What do they do outside the classroom?', in *In Proceedings of the Twelfth Australasian Conference on Computing Education*, pp. 53–62.
- Cho, K., Schunn, C. D. and Wilson, R. W. (2006) 'Validity and reliability of scaffolded peer assessment of writing from instructor and student perspectives', *Journal of Educational Psychology*, 98(4), pp. 891–901. doi: 10.1037/0022-0663.98.4.891.
- Chou, C. and Zou, N. (2020) 'An analysis of internal and external feedback in self-regulated learning activities mediated by self-regulated learning tools and open learner models', *international journal of educational technology in higher education*, pp. 17–55. doi: <https://doi.org/10.1186/s41239-020-00233-y>.
- Chris, H. (2016) *peerScholar – Steve Joordens (co-founder)*, *The University of British Columbia*. Available at: <https://virtual.educ.ubc.ca/wp/etec522/2016/06/02/peerscholar-steve-joordens-co-founder/> (Accessed: 16 January 2022).
- Chrysafiadi, K. *et al.* (2019) 'Intelligent Mechanism for the Creation of Dynamically Adaptive Learning Material', *International Frequency Sensor Association*, 234(6), pp. 22–29.
- Clarke, V. and Braun, V. (2014) 'Thematic Analysis', in *Encyclopedia of Critical Psychology*. Springer, New York, NY., pp. 223–246. doi: [https://doi.org/10.1007/978-1-4614-5583-7\\_311](https://doi.org/10.1007/978-1-4614-5583-7_311).
- Cockburn, A. and Williams, L. (2001) 'The Costs and Benefits of Pair Programming', in *eXtreme Programming and Flexible Processes in Software Engineering*, pp. 223–243. doi: 10.1108/00012530210448235.
- Cohen, L., Manion, L. and Morrison, K. (2012) *Research methods in education*. sixth Ed., *Professional Development in Education*. sixth Ed. Routledge, Taylor & Francis Group. doi: 10.1080/19415257.2011.643130.
- Colblindor (2006) *Color Blind Essentials*, [www.colblindor.com](http://www.colblindor.com). Available at: <https://www.color-blindness.com/> (Accessed: 16 October 2022).
- Collimore, L., Pare, D. E. and Joordens, S. (2015) 'SWDYT: So What Do You Think? Canadian

students' attitudes about peerScholar, an online peer-assessment tool', *Learning Environments Research*, 18(1), pp. 33–45. doi: 10.1007/s10984-014-9170-1.

Creswell, J. and Clark, V. (2007) 'Designing and Conducting Mixed Methods Research', in *SAGE Publications*. 3ed ed., pp. 58–89. Available at:  
[https://www.sagepub.com/sites/default/files/upm-binaries/10982\\_Chapter\\_4.pdf](https://www.sagepub.com/sites/default/files/upm-binaries/10982_Chapter_4.pdf).

Dagley, M. *et al.* (2016) 'Increasing Retention and Graduation Rates Through a STEM Learning Community', *Journal of College Student Retention: Research, Theory & Practice*, 18(2), pp. 167–182.

Dale, B. (2006) 'Most difficult topics in CS1: results of an online survey of educators', *SIGCSE Bull.*, 38, pp. 49–53.

Dawson, P. (2015) 'Assessment rubrics: Towards clearer and more replicable design, research and practice', *Assessment & Evaluation in Higher Education*, 42, pp. 347–360.

Denzin, N. K. and Lincoln, Y. S. (2005) 'Introduction: The Discipline and Practice of Qualitative Research', in *Strategies of qualitative inquiry*. Sage Publications Ltd., pp. 1–43.

Devlin, M. (2015) *The Effect of Programming Competency on Success in Undergraduate Team Projects in Computing Science*. PhD thesis. Newcastle University. Available at:  
<http://hdl.handle.net/10443/2890>.

Dewey, J. (1938) *Experience and Education*. Edited by Kappa Delta Pi lecture series. New York : Macmillan.

Divjak, B., Grabar, D. and Maretić, M. (2016) 'Assessment analytics for peer-assessment: A model and Implementation', in *Program and Curricular Learning Analytics Workshop*, pp. 27–31.

Double, K. S., McGrane, J. A. and Hopfenbeck, T. N. (2020) 'The Impact of Peer Assessment on Academic Performance: A Meta-analysis of Control Group Studies', *Educational Psychology Review*. *Educational Psychology Review*, 32(2), pp. 481–509. doi: 10.1007/s10648-019-09510-3.

Drake, P. and Heath, L. (2010) *Practitioner Research at Doctoral Level: Developing Coherent Research Methodologies*, Routledge. doi: <https://doi.org/10.4324/9780203841006>.

Earl, L. M. (2012) *Assessment as learning: Using classroom assessment to maximize student learning*. Corwin Press.

Ecclestone, K. and Pryor, J. (2003) "'Learning Careers" or "Assessment Careers"? The Impact of Assessment Systems on Learning', *British Educational Research Journal*, 29(4), pp. 471–488. doi: 10.1080/01411920301849.

Ely, M. *et al.* (2003) *Doing Qualitative Research: Circles Within Circles*. Routledge.

Ertmer, P. A. and Newby, T. J. (2011) 'Behaviorism, Cognitivism, Constructivism: Comparing Critical Features From an Instructional Design Perspective', 24(3), pp. 55–76. doi: 10.1002/piq.

Falchikov, N. and Goldfinch, J. (2000) 'Student Peer Assessment in Higher Education: A Meta-

Analysis Comparing Peer and Teacher Marks', *Review of Educational Research*, 70(3), pp. 287–322. doi: 10.3102/00346543070003287.

Finlay, L. (2008) *Reflecting on 'Reflective practice'*, Practice-based Professional Learning Centre. Available at: [http://oro.open.ac.uk/68945/1/Finlay-\(2008\)-Reflecting-on-reflective-practice-PBPL-paper-52.pdf](http://oro.open.ac.uk/68945/1/Finlay-(2008)-Reflecting-on-reflective-practice-PBPL-paper-52.pdf).

Fowler, M. (2003) *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley Longman Publishing Co.

Fraille, J., Panadero, E. and Pardo, R. (2017) 'Co-creating rubrics: The effects on self-regulated learning, self-efficacy and performance of establishing assessment criteria with students', *Studies in Educational Evaluation*, 53, pp. 69–76. doi: 10.1016/j.stueduc.2017.03.003.

Galer, Margaret *et al.* (1992) *Methods and Tools in User-Centred Design for Information Technology*. North-Holland Publishing Co.

García, R. M. C. and Pardo, A. (2010) 'A Supporting System for Adaptive Peer Review based on Learners' Profiles', in *Proceedings of the Workshop on Computer-Supported Peer Review in Education CSPRED-2010*, pp. 22–31.

Geertz, C. (1973) *The Interpretation of Cultures: Selected Essays*. New York: Basic Books.

Gielen, S., Dochy, F. and Onghena, P. (2011) 'An inventory of peer assessment diversity', *Assessment & Evaluation in Higher Education*, 36(2), pp. 137–155.

Gikandi, J. W., Morrow, D. and Davis, N. E. (2011) 'Online formative assessment in higher education: A review of the literature', *Computers and Education*. Elsevier Ltd, 57(4), pp. 2333–2351. doi: 10.1016/j.compedu.2011.06.004.

Gonzalez, G. (2006) 'A systematic approach to active and cooperative learning in CS1 and its effects on CS2', *SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education*, pp. 133–137. doi: <https://doi.org/10.1145/1121341.1121386>.

Gopalan, V. *et al.* (2017) 'A review of the motivation theories in learning', in *The 2nd International Conference on Applied Science and Technology 2017 (ICAST'17)*. AIP Publishing., pp. 1–7. doi: 10.1063/1.5005376.

Graham, K. (2017) 'TechMatters: Peer to "Peergrade": Exploring an Online Tool to Facilitate Peer Evaluation', *LOEX Quarterly*, 44(1), pp. 4–6.

Gray, J. A. and DiLoreto, M. (2016) 'The Effects of Student Engagement, Student Satisfaction, and Perceived Learning in Online Learning Environments', *International Journal of Educational Leadership Preparation*, 11(1), p. 20.

Greene, J. C., Carcelli, V. J. and Graham, W. F. (1989) 'Toward a Conceptual Framework for Mixed-Method Evaluation Designs', *Educational Evaluation and Policy Analysis*, 11(3), pp. 255–274.

Gribbons, B. and Herman, J. (1997) 'True and Quasi-Experimental Designs True and Quasi-

- Experimental Designs', *Practical Assessment, Research, and Evaluation*, 5(14), pp. 1–3. doi: <https://doi.org/10.7275/fs4z-nb61>.
- Hämäläinen, H. *et al.* (2011) 'Applying peer-review for programming assignments', *International Journal on Information Technologies & Security*, 1, pp. 3–17.
- Hamer, J. *et al.* (2009) 'Quality of peer assessment in CS1', in *Proceedings of the Fifth International Workshop on Computing Education Research Workshop*. Association for Computing Machinery, pp. 27–36. doi: 10.1145/1584322.1584327.
- Hanks, B. *et al.* (2011) 'Pair programming in education: a literature review', *Computer Science Education*, 21(2), pp. 135–173. doi: 10.1080/08993408.2011.579808.
- Hanson, W. E. *et al.* (2005) 'Mixed methods research designs in counseling psychology', *Journal of Counseling Psychology*, 52(2), pp. 224–235. doi: 10.1037/0022-0167.52.2.224.
- Harland, D. J. (2015) 'An Introduction to Experimental Research An Introduction to Exploratory Research', *Nursing*, 4(3), p. 6.
- Harlen, W. and James, M. (1997) 'Assessment and Learning: differences and relationships between formative and summative assessment', *Assessment in Education: Principles, Policy & Practice*, 4(3), pp. 365–379. doi: 10.1080/0969594970040304.
- Harris, J. R. (2011) 'Peer assessment in large undergraduate classes: an evaluation of a procedure for marking laboratory reports and a review of related practices', *Advances in Physiology Education*, 35(2), pp. 178–187. doi: 10.1152/advan.00115.2010.
- Hawllitschek, A. *et al.* (2020) 'Drop-out in programming courses – prediction and prevention', *Journal of Applied Research in Higher Education*, 12(1), pp. 124–136. doi: <https://doi.org/10.1108/JARHE-02-2019-0035>.
- Hoxmeier, J. A. (2002) 'Experiential Learning within Computer Information Systems Courses', in *New Perspectives on Information Systems Development*. Springer, Boston, MA., pp. 611–618. doi: 10.1007/978-1-4615-0595-2\_49.
- Hsiao, I. H. and Brusilovsky, P. (2008) 'Modeling peer review in example annotation', in *16th International Conference on Computers in Education (ICCE 2008)*, pp. 357–361.
- Huisman, B. *et al.* (2018) 'Peer assessment in MOOCs: The relationship between peer reviewers' ability and authors' essay performance', *British Journal of Educational Technology*, 49(1), pp. 101–110. doi: <https://doi.org/10.1111/bjet.12520>.
- Huisman, B. *et al.* (2019) 'The impact of formative peer feedback on higher education students' academic writing: a Meta-Analysis', *Assessment and Evaluation in Higher Education*, 44(6), pp. 863–880. doi: 10.1080/02602938.2018.1545896.
- Hundhausen, C. *et al.* (2009) 'Integrating pedagogical code reviews into a CS 1 course: An empirical study', *ACM SIGCSE Bulletin*, 41(1), pp. 291–295. doi: 10.1145/1508865.1508972.
- Hundhausen, C. D., Agrawal, A. and Agarwal, P. (2013) 'Talking about Code: Integrating

- Pedagogical Code Reviews into Early Computing Courses', *ACM Transactions on Computing Education*, 13(3), pp. 1–28. doi: 10.1145/2499947.2499951.
- Hurtado, S., Eagan, K. and Chang, M. (2010) 'Degrees of Success Bachelor's Degree Completion Rates among Initial STEM Majors', *Higher Education Research Institute at UCLA*.
- Hwang, G.-J., Liang, Z.-Y. and Wang, H.-Y. (2016) 'An Online Peer Assessment-Based Programming Approach to Improving Students' Programming Knowledge and Skills', in *2016 International Conference on Educational Innovation through Technology (EITT)*, pp. 81–85. doi: 10.1109/EITT.2016.23.
- Hwang, W. Y. *et al.* (2008) 'A web-based programming learning environment to support cognitive development', *Interacting with Computers*. Elsevier, 20(6), pp. 524–534. doi: 10.1016/j.intcom.2008.07.002.
- Ibrahim, R. *et al.* (2011) 'Students Perceptions of Using Educational Games to Learn Introductory Programming', *Canadian Center of Science and Education*, 4(1), p. 205.
- Iglesias Pérez, M. C., Vidal-Puga, J. and Pino Juste, M. R. (2020) 'The role of self and peer assessment in Higher Education', *Studies in Higher Education*. Routledge, pp. 1–10. doi: 10.1080/03075079.2020.1783526.
- Imhof, C., Bergamin, P. and McGarrity, S. (2020) 'Implementation of Adaptive Learning Systems: Current State and Potential BT - Online Teaching and Learning in Higher Education', in *Online teaching and learning in higher education*. Springer, Cham., pp. 93–115. doi: 10.1007/978-3-030-48190-2\_6.
- Indriasari, T. D., Luxton-Reilly, A. and Denny, P. (2020) 'A Review of Peer Code Review in Higher Education', *ACM Transactions on Computing Education*, 20(3), pp. 1-25. doi: 10.1145/3403935.
- Isaac, O., Christian, N. and Amana, Y. (2021) 'Enhancing the Teaching and Learning of Computer Programming using Collaborative Method of Delivery', *International Journal of Advances in Scientific Research and Engineering (ijasre)*, 7(1), pp. 18–23.
- Jamieson, S. (2004) 'Likert scales: how to (ab)use them', *Medical education*, 38(12), pp. 1217–1218. doi: 10.1111/j.1365-2929.2004.02012.x.
- Jensen, M. B., Elverum, C. W. and Steinert, M. (2017) 'Eliciting unknown unknowns with prototypes: Introducing prototrials and prototrial-driven cultures', *Design Studies*, 49, pp. 1–31. doi: <https://doi.org/10.1016/j.destud.2016.12.002>.
- Jisc (2021) *Jisc Website*, Jisc. Available at: <https://www.jisc.ac.uk/> (Accessed: 12 March 2019).
- Johnson, B., Onwuegbuzie, A. J. and Turner, L. A. (2007) 'Toward a definition of mixed methods research', *Journal of Mixed Methods Research*, 1, pp. 112–133. doi: 10.1017/9781316418376.015.
- Jones, I. and Alcock, L. (2014) 'Peer assessment without assessment criteria', *Studies in Higher Education*. Taylor & Francis, 39(10), pp. 1774–1787. doi: 10.1080/03075079.2013.821974.

- Joordens, S., Shakinaz, D. and Paré, D. (2009) 'The Pedagogical Anatomy of Peer-Assessment: Dissecting a peerScholar Assignment', *Systemics, Cybernetics And Informatics*, 7, pp. 11–15.
- Kahraman, Z. E. H. (2010) 'Using user-centered design approach in course design', *Procedia - Social and Behavioral Sciences*, 2(2), pp. 2071–2076.
- Kaufman, J. H. and Schunn, C. D. (2011) 'Students' perceptions about peer assessment for writing: Their origin and impact on revision work', *Instructional Science*, 39(3), pp. 387–406. doi: 10.1007/s11251-010-9133-6.
- Kavanagh, S. and Luxton-Reilly, A. (2016) 'Rubrics used in peer assessment', in *Proceedings of the Australasian Computer Science Week Multiconference on - ACSW '16*, pp. 1–6. doi: 10.1145/2843043.2843347.
- Keim, D. et al. (2008) 'Visual Analytics: Definition, Process, and Challenges Daniel', in *Information visualization*. Springer, Berlin, Heidelberg., pp. 154–175.
- King, C. E. (2018) 'Feasibility and Acceptability of Peer Assessment for Coding Assignments in Large Lecture Based Programming Engineering Courses', in *2018 IEEE Frontiers in Education Conference (FIE)*. IEEE, pp. 1–9. doi: 10.1109/FIE.2018.8659246.
- Kinnunen, P. and Malmi, L. (2006) 'Why students drop out CS1 course?', in *ICER 2006 - Proceedings of the 2nd International Computing Education Research Workshop*, pp. 97–108. doi: 10.1145/1151588.1151604.
- Kinnunen, P. and Malmi, L. (2008) 'CS minors in a CS1 course', in *ICER'08 - Proceedings of the ACM Workshop on International Computing Education Research*, pp. 79–90. doi: 10.1145/1404520.1404529.
- Kirk, A. (2012) *Data Visualization: a successful design process*. Packt Publishing Limited.
- Koc, C. (2011) 'The Views of Prospective Class Teachers about Peer Assessment in Teaching Practice', *Kuram Ve Uygulamada Egitim Bilimleri*, 11(4), pp. 1979–1989.
- Kollar, I. and Fischer, F. (2010) 'Peer assessment as collaborative learning: A cognitive perspective', *Learning and Instruction*, 20(4), pp. 344–348.
- Kritikou, Y. et al. (2008) 'User Profile Modeling in the context of web-based learning management systems', *Journal of Network and Computer Applications*, 31(4), pp. 603–627. doi: <https://doi.org/10.1016/j.jnca.2007.11.006>.
- Kurilovas, E. (2019) 'Advanced machine learning approaches to personalise learning: learning analytics and decision making', *Behaviour and Information Technology*. Taylor & Francis, 38(4), pp. 410–421. doi: 10.1080/0144929X.2018.1539517.
- Langan, A. M. et al. (2005) 'Peer assessment of oral presentations: effects of student gender, university affiliation and participation in the development of assessment criteria', *Assessment & Evaluation in Higher Education*. Routledge, 30(1), pp. 21–34. doi: 10.1080/0260293042003243878.



- Lauff, C., Menold, J. and Wood, K. L. (2019) 'Prototyping canvas: Design tool for planning purposeful prototypes', in *Proceedings of the International Conference on Engineering Design, ICED*, pp. 1563–1572. doi: 10.1017/dsi.2019.162.
- Lázaro Alvarez, N., Callejas, Z. and Griol, D. (2020) 'Predicting computer engineering students' dropout in cuban higher education with pre-enrollment and early performance data', *JOTSE: Journal of Technology and Science Education*, 10(2), pp. 241–258.
- Leach, L., Neutze, G. and Zepke, N. (2001) 'Assessment and Empowerment: Some critical questions', *Assessment & Evaluation in Higher Education*, 26(4). doi: 10.1080/02602930120063457.
- Lejk, M. and Wyvill, M. (2001) 'Peer assessments of contributions to a group project: a comparison of holistic and category-based approaches', *Assessment & Evaluation in Higher Education*, 26(1), pp. 61–72. doi: 10.1080/0260293002002229.
- Lemons, G. *et al.* (2010) 'The benefits of model building in teaching engineering design', *Design Studies*, 31(3), pp. 288–309. doi: <https://doi.org/10.1016/j.destud.2010.02.001>.
- Lesser, V. M. *et al.* (2016) 'Mixed-Mode Surveys Compared with Single Mode Surveys: Trends in Responses and Methods to Improve Completion', *Journal of Rural Social Sciences*, 31(3), p. 7.
- Lew, M. D. N., Alwis, W. A. M. and Schmidt, H. G. (2010) 'Accuracy of students' self-assessment and their beliefs about its utility', *Assessment & Evaluation in Higher Education*, 35(2), pp. 135–156. doi: 10.1080/02602930802687737.
- Lewis, A., Florian, L. and Porter, J. (2007) 'Research and pupil voice', in *Handbook of Special Education*. Sage Publications Ltd, pp. 222–232.
- Li, H. *et al.* (2016) 'Peer assessment in the digital age: a meta-analysis comparing peer and teacher ratings', *Assessment and Evaluation in Higher Education*. Routledge, 41(2), pp. 245–264. doi: 10.1080/02602938.2014.999746.
- Li, H. *et al.* (2020) 'Does peer assessment promote student learning? A meta-analysis', *Assessment and Evaluation in Higher Education*. Routledge, 45(2), pp. 193–211. doi: 10.1080/02602938.2019.1620679.
- Li, J., Fu, X. and Yang, Q. (2017) 'Choosing Peers , Improve the Quality of Peer Assessment', in *4th International Conference on Machinery, Materials and Information Technology Applications (ICMMITA 2016)*, pp. 1124–1127. doi: 10.2991/icmmita-16.2016.206.
- Li, L. and Gao, F. (2016) 'The effect of peer assessment on project performance of students at different learning levels', *Assessment and Evaluation in Higher Education*. Routledge, 41(6), pp. 885–900. doi: 10.1080/02602938.2015.1048185.
- Li, L. and Grion, V. (2019) 'The Power of Giving Feedback and Receiving Feedback in Peer Assessment', *All Ireland Journal of Higher Education*, 11(2), pp. 1–17.
- Lin, L. (2015) *Exploring Collaborative Learning: Theoretical and Conceptual Perspectives*,

*Investigating Chinese HE EFL Classrooms*. Springer, Berlin, Heidelberg. doi: [https://doi.org/10.1007/978-3-662-44503-7\\_2](https://doi.org/10.1007/978-3-662-44503-7_2).

Lincoln, Y. S. and Guba., E. G. (1985) *Naturalistic inquiry*, sage.

Liu, N.-F. and Carless, D. (2006) 'Peer feedback: the learning element of peer assessment', *Teaching in Higher Education*. Routledge, 11(3), pp. 279–290. doi: 10.1080/13562510600680582.

Lopez-Real, F. and Chan, Y. P. R. (1999) 'Peer assessment of a group project in a primary mathematics education course', *Assessment and Evaluation in Higher Education*, 24(1), pp. 67–79. doi: 10.1080/0260293990240106.

Lui, K. M. and Chan, K. C. C. (2006) 'Pair programming productivity: Novice-novice vs. expert-expert', *International Journal of Human Computer Studies*, 64(9), pp. 915–925. doi: 10.1016/j.ijhcs.2006.04.010.

Luxton-Reilly, A. (2016) 'Learning to program is easy', in *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, pp. 284–289. doi: 10.1145/2899415.2899432.

Luxton-Reilly, A., Lewis, A. and Plimmer, B. (2018) 'Comparing Sequential and Parallel Code Review Techniques for Formative Feedback', in *Proceedings of the 20th Australasian Computing Education Conference*, pp. 45–52. doi: 10.1145/3160489.3160498.

Lynch, R., McNamara, P. M. and Seery, N. (2012) 'Promoting deep learning in a teacher education programme through self- and peer-assessment and feedback', *European Journal of Teacher Education*, 35(2), pp. 179–197. doi: 10.1080/02619768.2011.643396.

Machado, J. L. M. *et al.* (2008) 'Self- and peer assessment may not be an accurate measure of PBL tutorial process', *BMC Medical Education*, 8, pp. 1–6. doi: 10.1186/1472-6920-8-55.

Machanick, P. (2007) 'A social construction approach to computer science education', *Computer Science Education*. Routledge, 17(1), pp. 1–20. doi: 10.1080/08993400600971067.

Mah, D.-K. (2016) 'Learning Analytics and Digital Badges: Potential Impact on Student Retention in Higher Education', *Technology, Knowledge and Learning*, 21(3), pp. 285–305. doi: 10.1007/s10758-016-9286-8.

Maravanyika, M., Dlodlo, N. and Jere, N. (2017) 'An adaptive recommender-system based framework for personalised teaching and learning on e-learning platforms', in *IST-Africa Week Conference (IST-Africa)*, pp. 1–9. doi: 10.23919/ISTAFRICA.2017.8102297.

Mardani, A. *et al.* (2015) 'Multiple criteria decision-making techniques and their applications - A review of the literature from 2000 to 2014', *Economic Research-Ekonomiska Istrazivanja*. Routledge, 28(1), pp. 516–571. doi: 10.1080/1331677X.2015.1075139.

McDowell, C. *et al.* (2006) 'Pair programming improves student retention, confidence, and program quality', *Communications of the ACM*, 8(49), pp. 90–95.

- Menold, J., Jablokow, K. and Simpson, T. (2017) 'Prototype for X (PFX): A holistic framework for structuring prototyping methods to support engineering design', *Design Studies*, 50, pp. 70–112. doi: <https://doi.org/10.1016/j.destud.2017.03.001>.
- Mildner, V. (2019) 'The SAGE Encyclopedia of Human Communication Sciences and Disorders Experimental Research', *SAGE reference*, pp. 728–732. doi: 10.4135/9781483380810.n242.
- Miller, P. J. (2003) 'The effect of Scoring criteria Specificity on peer and Self-assessment', *Assessment and Evaluation in Higher Education*, 28(4), pp. 383–394. doi: 10.1080/0260293032000066218.
- Mishra, R. K. (2013) 'Vygotskian Perspective of Teaching-Learning', *Innovation: International Journal of Applied Research*, 1(1), pp. 21–28.
- Mockaroo Website* (2022) *Mockaroo, LLC*. Available at: <https://www.mockaroo.com/> (Accessed: 15 June 2021).
- Mogessie, M., Firlab, S. R. L. and Riccardi, G. (2016) 'Exploring the Role of Online Peer-Assessment as a Tool of Early Intervention', in *International Symposium on Emerging Technologies for Education*. Springer, Cham., pp. 635–644. doi: 10.1007/978-3-319-71084-6.
- Morgan, B. (2011) 'Consulting pupils about classroom teaching and learning: policy, practice and response in one school', *Research Papers in Education*, 24(4), pp. 445–467.
- Mullan, J., Sclater, N. and Peasgood, A. (2016) 'Learning Analytics in Higher Education: A review of UK and international practice', *JISC CETIS Center of Educational Technology & Interoperability Standards*, p. 40.
- Nachmias, C. F. and Nachmias, D. (1996) *Research Methods in the Social Sciences*. 5th ed. London: Arnold.
- Neeley, W. L. *et al.* (2013) 'Building fast to think faster: exploiting rapid prototyping to accelerate ideation during early stage design', in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers., pp. 1–8. doi: <https://doi.org/10.1115/DETC2013-12635>.
- Newby, P. (2010) *Research Methods for Education*. Routledge.
- Ng, V. and Fai, C. M. (2017) 'Engaging Student Learning through Peer Assessments', in *Proceedings of the 2017 International Conference on E-Education, E-Business and E-Technology*, pp. 30–35.
- Nicol, D. (2009) *Quality Enhancement Themes: The First Year Experience*. Mansfield: Quality Assurance Agency for Higher Education.
- Nicol, D., Thomson, A. and Breslin, C. (2014) 'Rethinking feedback practices in higher education: a peer review perspective', *Assessment and Evaluation in Higher Education*, 39(1), pp. 102–122. doi: 10.1080/02602938.2013.795518.
- Norman, M. and Hyland, T. (2003) 'The role of confidence in lifelong learning', *Educational*

*studies*, 29(2–3), pp. 261–272.

Oldfield, K. A. and Macalpine, J. M. K. (1995) 'Peer and Self-assessment at Tertiary Level—an experiential report', *Assessment & Evaluation in Higher Education*, 20(1), pp. 125–132. doi: 10.1080/0260293950200113.

Olson, J. D. *et al.* (2016) 'Applying constant comparative method with multiple investigators and inter-coder reliability', *The Qualitative Report*, 21(1), pp. 26–42.

Orsmond, P., Merry, S. and Reiling, K. (2002) 'The Use of Exemplars and Formative Feedback when Using Student Derived Marking Criteria in Peer and Self-assessment', *Assessment & Evaluation in Higher Education*, 25(1), pp. 23–38. doi: 10.1080/026029302200000133.

Pallant, J. (2001) *SPSS Survival Manual a step by step guide to data analysis*. Fifth ed. Open University Press.

Panadero, E. (2016) 'Is it safe? Social, interpersonal, and human effects of peer assessment: A review and future directions', in *Handbook of human and social conditions in assessment*. New York: Routledge, pp. 247–266.

Panadero, E. and Brown, G. T. L. (2017) 'Teachers' reasons for using peer assessment: positive experience predicts use', *European Journal of Psychology of Education*, 32(1), pp. 133–156. doi: 10.1007/s10212-015-0282-5.

Papadakis, S. (2018) 'Is Pair Programming More Effective than Solo Programming for Secondary Education Novice Programmers?: A Case Study', *International Journal of Web-Based Learning and Teaching Technologies*, 13(1). doi: 10.4018/IJWLTT.2018010101.

Papinczak, T., Young, L. and Groves, M. (2007) 'Peer assessment in problem-based learning: A qualitative study', *Advances in Health Sciences Education*, 12(2), pp. 169–186. doi: 10.1007/s10459-005-5046-6.

Pappas, I. O., Giannakos, M. N. and L. Jaccheri (2016) 'Investigating factors influencing students' intention to dropout computer science studies', in *Procs of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, pp. 198–203.

Park, J. *et al.* (2017) 'Eliph: Effective Visualization of Code History for Peer Assessment in Programming Education', in *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, pp. 458–467. doi: 10.1145/2998181.2998285.

Park, J. and Williams, K. (2016) 'The Effects of Peer- and Self-assessment on the Assessors', in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education - SIGCSE '16*, pp. 249–254. doi: 10.1145/2839509.2844602.

Passarelli, A. and Kolb, D. (2012) *Using Experiential Learning Theory to Promote Student Learning and Development in Programs of Education Abroad*, *Student learning abroad: What our students are learning, what they're not, and what we can do about it*, pp.137-161.

Patchan, M. M. *et al.* (2013) 'The effects of skill diversity on commenting and revisions',

*Instructional Science*, 41(2), pp. 381–405. doi: 10.1007/s11251-012-9236-3.

Patchan, M. M. and Schunn, C. D. (2016) 'Understanding the effects of receiving peer feedback for text revision: Relations between author and reviewer ability', *Journal of Writing Research*, 8(2), pp. 227–265. doi: 10.17239/jowr-2016.08.02.03.

Petersen, Andrew *et al.* (2016) 'Revisiting why students drop CS1', *ACM International Conference Proceeding Series*, pp. 71–80. doi: 10.1145/2999541.2999552.

Pon-Barry, H., Packard, B. W.-L. and John, A. St. (2017) 'Expanding capacity and promoting inclusion in introductory computer science: a focus on near-peer mentor preparation and code review', *Computer Science Education*. Routledge, 27(1), pp. 54–77. doi: 10.1080/08993408.2017.1333270.

Quality Assurance Agency (QAA) (2003) *Learning from subject review 1993–2001: Sharing good practice*.

Quintana, C. *et al.* (2013) 'Exploring a structured definition for learner-centered design', in *Fourth international conference of the learning sciences*, pp. 256–263.

R (2021) *R-3.6.2 for Windows, R project*. Available at: <https://cran.r-project.org/bin/windows/base/old/3.6.2/> (Accessed: 16 October 2020).

Rahmat, M. *et al.* (2012) 'Major Problems in Basic Programming that Influence Student Performance', *Procedia - Social and Behavioral Sciences*, 59, pp. 287–296. doi: 10.1016/j.sbspro.2012.09.277.

Rawn, C. (2021) *peerScholar, The University of British Columbia*. Available at: <https://isit.arts.ubc.ca/peerscholar/> (Accessed: 19 September 2018).

Reinholz, D. (2016) 'The assessment cycle: a model for learning through peer assessment', *Assessment & Evaluation in Higher Education*. Routledge, 41(2), pp. 301–315. doi: 10.1080/02602938.2015.1008982.

Reschly, A. L. (2020) 'Dropout Prevention and Student Engagement', in *Student Engagement: Effective Academic, Behavioral, Cognitive, and Affective Interventions at School*. Cham: Springer International Publishing, pp. 31–54. doi: 10.1007/978-3-030-37285-9\_2.

Robins, A., Rountree, J. and Rountree, N. (2003) 'Learning and teaching programming: A review and discussion', *International Journal of Phytoremediation*, 21(1), pp. 137–172. doi: 10.1076/csed.13.2.137.14200.

Rogers, Y., Sharp, H. and Preece, J. (2002) *Interaction Design-beyond human computer interaction, jon wiley & sons. Inc.*

Rotsaert, T., Panadero, E. and Schellens, T. (2018) 'Anonymity as an instructional scaffold in peer assessment: its effects on peer feedback quality and evolution in students' perceptions about peer assessment skills', *European Journal of Psychology of Education*, 33(1), pp. 75–99. doi: 10.1007/s10212-017-0339-8.

- Rountree, N. *et al.* (2004) 'Interacting factors that predict success and failure in a CS1 course', in *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, pp. 101–104.
- Sadler, D. R. (1989) 'Formative assessment and the design of instructional systems', *Instructional Science*, 18, pp. 119–144. Available at: [http://michiganassessmentconsortium.org/sites/default/files/Formative Assessment and Design of Instructional Systems.pdf](http://michiganassessmentconsortium.org/sites/default/files/Formative%20Assessment%20and%20Design%20of%20Instructional%20Systems.pdf).
- Sadler, D. R. (2009) 'Transforming holistic assessment and grading into a vehicle for complex learning', *Assessment, Learning and Judgement in Higher Education*, pp. 1–233. doi: 10.1007/978-1-4020-8905-3.
- Sadler, P. M. and Good, E. (2006) 'The Impact of Self- and Peer- Grading on Student Learning', *Educational Assessment*, 11(1), pp. 1–37. doi: 10.1207/s15326977ea1101.
- Saldaña, J. (2016) *The Coding Manual for Qualitative Researchers*. Edited by J. Seaman. SAGE Publications Ltd.
- Sancho-Thomas, P., Fuentes-Fernández, R. and Fernández-Manjón, B. (2009) 'Learning teamwork skills in university programming courses', *Computers and Education*. Elsevier Ltd, 53(2), pp. 517–531. doi: 10.1016/j.compedu.2009.03.010.
- Sarıyalçinkaya, A. D. *et al.* (2021) 'Reflections on Adaptive Learning Analytics: Adaptive Learning Analytics', in *Advancing the Power of Learning Analytics and Big Data in Education*. IGI Global, pp. 61–62. doi: 10.4018/978-1-7998-7103-3.ch003.
- Scager, K. *et al.* (2016) 'Collaborative Learning in Higher Education: Evoking Positive Interdependence', *CBE—Life Sciences Education*, 15(4). doi: 10.1187/cbe.16-07-0219.
- Schmidt, M. *et al.* (2020) 'Methods of User Centered Design and Evaluation for Learning Designers', in *Learner and User Experience Research: An Introduction for the Field of Learning Design & Technology*. EdTech Boo. Available at: [https://edtechbooks.org/ux/ucd\\_methods\\_for\\_lx](https://edtechbooks.org/ux/ucd_methods_for_lx).
- Shaheen, N. (2016) 'International students' critical thinking–related problem areas: UK university teachers' perspectives', *Journal of Research in International Education*, 15(1), pp.18-31. doi: <https://doi.org/10.1177/1475240916635895>.
- Sharma, D. and Potey, M. (2018) 'Effective Learning through Peer Assessment using Peergrade tool', in *IEEE Ninth International Conference on Technology for Education (T4E)*, pp. 114–117. doi: 10.1109/T4E.2018.00031.
- Shatri, K. and Buza, K. (2017) 'The Use of Visualization in Teaching and Learning Process for Developing Critical Thinking of Students', *European Journal of Social Sciences Education and Research*, 9(1), p. 71-74. doi: 10.26417/ejser.v9i1.p71-74.
- Shee, D. Y. and Wang, Y. S. (2008) 'Multi-criteria evaluation of the web-based e-learning system: A methodology based on learner satisfaction and its applications', *Computers and Education*,

50(3), pp. 894–905. doi: 10.1016/j.compedu.2006.09.005.

Shui Ng, W. (2017) 'The Impact of Peer Assessment and Feedback Strategy in Learning Computer Programming in Higher Education', *Issues in Informing Science and Information Technology*, 9, pp. 017–027. doi: 10.28945/1601.

Sithole, A. *et al.* (2017) 'Student Attraction, Persistence and Retention in STEM Programs: Successes and Continuing Challenges', *Higher Education Studies*, 7(1), p. 46. doi: 10.5539/hes.v7n1p46.

Siti Rosminah, M. D. and Ahmad Zamzuri, M. A. (2012) 'Difficulties in learning Programming: Views of students', in *1st International Conference on Current Issues in Education (ICCIE2012)*, pp. 74–78. Available at: [http://rozmiens.com/myarticle/ICCIE2012\\_siti\\_rosminah.pdf](http://rozmiens.com/myarticle/ICCIE2012_siti_rosminah.pdf).

Sitthiworachart, J. and Joy, M. (2003) 'Web-based peer assessment in learning computer programming', in *Proceedings of the 3rd IEEE International Conference on Advanced Learning Technologies (ICALT '03)*, pp. 180–184. Available at: [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=1215052](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1215052).

Sitthiworachart, J. and Joy, M. (2004) 'Effective peer assessment for learning computer programming', in *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*, p. 122-126. doi: 10.1145/1007996.1008030.

Sitthiworachart, J. and Joy, M. (2008) 'Computer support of effective peer assessment in an undergraduate programming class', *Journal of Computer Assisted Learning*, 24(3), pp. 217–231. doi: 10.1111/j.1365-2729.2007.00255.x.

Sondergaard, H. (2009) 'Learning from and with Peers: The Different Roles of Student Peer Reviewing', in *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education*. New York, NY, USA: Association for Computing Machinery (ITiCSE '09), pp. 31–35. doi: 10.1145/1562877.1562893.

Søndergaard, H. and Mulder, R. A. (2012) 'Collaborative learning through formative peer review : pedagogy , programs and potential', *Computer Science Education*, 22(4), pp. 343–367. doi: 10.1080/08993408.2012.728041.

Song, Y., Hu, Z. and Gehringer, E. F. (2015) 'Closing the Circle: Use of Students' Responses for Peer-Assessment Rubric Improvement', in *International Conference on Web-Based Learning*. Springer, Cham., pp. 27–36. doi: 10.1007/978-3-319-25515-6.

Sosu, E. M. and Pheunpha, P. (2019) 'Trajectory of University Dropout: Investigating the Cumulative Effect of Academic Vulnerability and Proximity to Family Support', in *Frontiers in Education*, 4(6). doi: 10.3389/educ.2019.00006.

Spyropoulou, M. *et al.* (2013) 'Evaluating the correspondence of educational software to learning theories', in *ACM International Conference Proceeding Series*, pp. 250–257. doi: 10.1145/2491845.2491882.

Sridharan, B., Tai, J. and Boud, D. (2018) 'Does the use of summative peer assessment in

- collaborative group work inhibit good judgement?', *Higher Education*, 77(5), pp. 1–18. doi: 10.1007/s10734-018-0305-7.
- Stalhane, T. *et al.* (2004) 'Teaching the process of code review', in *2004 Australian Software Engineering Conference. Proceedings*. IEEE, pp. 271–278. doi: 10.1109/ASWEC.2004.129048.
- Stegeman, M. (2014) 'Understanding code quality for introductory courses', in *PPIG*, p. 10. Available at: <http://www.ppig.org/sites/ppig.org/files/2015-PPIG-26th-Stegeman.pdf>.
- Stegeman, M., Barendsen, E. and Smetsers, S. (2014) 'Towards an empirically validated model for assessment of code quality', in *Proceedings of the 14th Koli Calling International Conference on Computing Education Research - Koli Calling '14*, pp. 99–108. doi: 10.1145/2674683.2674702.
- Straub, D. and Gefen, D. (2004) 'Validation Guidelines for IS Positivist Research', *Communications of the Association for Information Systems*, 13(1), p. 24. doi: 10.17705/1cais.01324.
- Sullivan, G. M. and Artino, A. R. (2013) 'Analyzing and interpreting data from likert-type scales', *Journal of graduate medical education*, 5(4), pp. 541–542. doi: 10.4300/JGME-5-4-18.
- Sun, Q. *et al.* (2019) 'Formative Assessment of Programming Language Learning Based on Peer Code Review : Implementation and Experience Report', *Tsinghua Science and Technology*, 24(4), pp. 423–434.
- Sung, Y.-T. *et al.* (2010) 'How many heads are better than one? The reliability and validity of teenagers' self- and peer assessments.', *Journal of Adolescence*, 33, pp. 135–145. doi: 10.1016/j.adolescence.2009.04.004.
- Suzuki, Y. (2016) *Visualizing Peer Assessment*. Available at: <https://www.ischool.berkeley.edu/sites/default/files/projects/finalpaper-yusuzuki-2.pdf>.
- Taherdoost, H. (2018) 'Validity and Reliability of the Research Instrument; How to Test the Validation of a Questionnaire/Survey in a Research', *SSRN Electronic Journal*, 5(3), pp. 28–36. doi: 10.2139/ssrn.3205040.
- Taras, M. (2005) 'Assessment - Summative and formative - Some theoretical reflections', *British Journal of Educational Studies*, 53(4), pp. 466–478. doi: 10.1111/j.1467-8527.2005.00307.x.
- Taylor, E. *et al.* (2013) 'Choosing learning methods suitable for teaching and learning in computer science', in *Proceedings of the International Conference e-Learning 2013*, pp. 74–82.
- The University of New South Wales (2015) *Student Peer Assessment | UNSW Teaching Staff Gateway*. Available at: <https://teaching.unsw.edu.au/peer-assessment>.
- Thyer, B. (2001) *The Handbook of Social Work Research Methods*. Thousand Oaks, CA: Sage.
- To, J. and Panadero, E. (2019) 'Peer assessment effects on the self-assessment process of first-year undergraduates', *Assessment & Evaluation in Higher Education*, 44(6), pp. 920–932.
- Topping, K. (1998) 'Peer assessment between students in colleges and universities', *Review of*



- educational Research*, 68(3), pp. 249–276. doi: 10.3102/00346543068003249.
- Topping, K. J. (2005) 'Trends in Peer Learning', *Educational psychology*, 25(6), pp. 631–645.
- Topping, K. J. (2010) 'Methodological quandaries in studying process and outcomes in peer assessment', *Learning and Instruction*. Elsevier Ltd, 20(4), pp. 339–343. doi: 10.1016/j.learninstruc.2009.08.003.
- Triantaphyllou, E. (2000) *Multi-Criteria Decision Making Methods: A Comparative Study*. Kluwer Academic Publishers. doi: 10.1007/978-1-4757-3157-6.
- Trytten, D. A. (2005) 'A design for team peer code review', in *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education (SIGCSE'05)*. ACM, New York, NY, pp. 455–459. doi: <https://doi.org/10.1145/1047344.1047492>.
- Tseng, J. C. R. *et al.* (2008) 'Development of an adaptive learning system with two sources of personalization information', *Computers & Education*, 51(2), pp. 776–786. doi: <https://doi.org/10.1016/j.compedu.2007.08.002>.
- Turner, S. A. *et al.* (2008) 'Misunderstandings about object-oriented design: Experiences using code reviews', in *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education (SIGCSE'08)*, pp. 97–101. doi: <https://doi.org/10.1145/1352135.1352169>.
- Turner, S. A., Pérez-Quñones, M. A. and Edwards, S. H. (2018) 'Peer review in CS2: Conceptual learning and high-level thinking', *ACM Transactions on Computing Education*, 18(3), pp. 1–37. doi: 10.1145/3152715.
- Ueki, Y. and Ohnishi, K. (2016) 'Visualizing Self- and Peer-assessment Data by a Self-organizing Map for Inducing Awareness in Learners', *Int J Comput Inf Syst Ind Manag Appl (IJCISIM)*, 8, pp. 23–32.
- Van Den Berg, I., Admiraal, W. and Pilot, A. (2006) 'Design principles and outcomes of peer assessment in higher education', *Studies in Higher Education*, 30(03), pp. 341–356.
- Van Der Pol, J. *et al.* (2008) 'The nature, reception, and use of online peer feedback in higher education', *Computers and Education*, 51(4), pp. 1804–1817. doi: 10.1016/j.compedu.2008.06.001.
- Van Schenk Hof, M. *et al.* (2018) 'Peer evaluations within experiential pedagogy: Fairness, objectivity, retaliation safeguarding, constructive feedback, and experiential learning as part of peer assessment', *International Journal of Management Education*. Elsevier, 16(1), pp. 92–104. doi: 10.1016/j.ijme.2017.12.003.
- Venables, A. and Haywood, L. (2003) 'Programming Students NEED Instant Feedback!', in *5th Australasian Computing Education Conference (ACE2003)*, pp. 267–272.
- Vickerman, P. (2009) 'Student perspectives on formative peer assessment: An attempt to deepen learning?', *Assessment and Evaluation in Higher Education*, 34(2), pp. 221–230. doi: 10.1080/02602930801955986.

- Vieira, C., Parsons, P. and Byrd, V. (2018) 'Visual learning analytics of educational data: A systematic literature review and research agenda', *Computers and Education*. Elsevier, 122(1), pp. 119–135. doi: 10.1016/j.compedu.2018.03.018.
- Vygotsky, L. (1980) *Mind in Society: The Development of Higher Psychological Processes*. Harvard University Press.
- Walker, H. M. (2017) 'Retention of students in introductory computing courses: curricular issues and approaches', *ACM Inroads*, 8(4), pp. 14–16.
- Wall, M. B., Ulrich, K. T. and Flowers, W. C. (1992) 'Evaluating Prototyping Technologies for Product Design', *Research in Engineering Design*, 3, pp. 163–177.
- Wang, T. *et al.* (2011) 'Ability-training-oriented automated assessment in introductory programming course', *Computers and Education*. Elsevier Ltd, 56(1), pp. 220–226. doi: 10.1016/j.compedu.2010.08.003.
- Wang, X. M. *et al.* (2017) 'Enhancing students' computer programming performances, critical thinking awareness and attitudes towards programming: An online peerassessment attempt', *Educational Technology and Society*, 20(4), pp. 58–68.
- Wang, Y. *et al.* (2012) 'Assessment of programming language learning based on peer code review model: Implementation and experience report', *Computers and Education*. Elsevier Ltd, 59(2), pp. 412–422. doi: 10.1016/j.compedu.2012.01.007.
- Wang, Y., Liang, Y. and Liu, L. (2012) 'A Motivation Model of Peer Assessment in Programming Language Learning', *Journal of Educational Computing Research*, 52, pp. 1–12. doi: 10.1177/0735633115571303.
- Wanner, T. and Palmer, E. (2018) 'Formative self-and peer assessment for improved student learning: the crucial factors of design, teacher participation and feedback', *Assessment & Evaluation in Higher Education*. Routledge, 43(7), pp. 1032–1047. doi: 10.1080/02602938.2018.1427698.
- Ward, B. A. (1987) 'Instructional Grouping in the Classroom', *School Improvement Research Series*, 24, pp. 1–11.
- Watson, C. and Li, F. W. (2014) 'Failure rates in introductory programming revisited', in *Proceedings of Conference on Innovation and Technology in Computer Science Education, Uppsala, Sweden*, pp. 39–44.
- Wegner, C., Minnaert, L. and Strehlke, F. (2021) 'The importance of learning strategies and how the project "Kolumbus-Kids" promotes them successfully', *European Journal of Science and Mathematics Education*, 1(3), pp. 137–143. doi: 10.30935/scimath/9393.
- Wen, M. L. and Tsai, C. C. (2006) 'University Students' Perceptions of and Attitudes Toward (Online) Peer Assessment', *Higher Education*, 51, pp. 27–44. doi: 10.1007/s10734-004-6375-8.
- Williams, L. and Upchurch, R. L. (2001) 'In support of student pair-programming', *ACM SIGCSE*

*Bulletin*, 33(1), pp. 327–331. doi: 10.1145/366413.364614.

Wilson, M. J., Diao, M. M. and Huang, L. (2015) “‘I’m not here to learn how to mark someone else’s stuff’”: an investigation of an online peer-to-peer review workshop tool’, *Assessment & Evaluation in Higher Education*. Routledge, 40(1), pp. 15–32. doi: 10.1080/02602938.2014.881980.

Wind, D. K., Jørgensen, R. M. and Hansen, S. L. (2018) ‘Peer Feedback with Peergrade’, in *ICEL 2018 13th International Conference on e-Learning*, pp. 184–192.

Wise, A. F. and Vytasek, J. (2017) ‘Learning Analytics Implementation Design’, in *Learning analytics implementation design*, pp. 151–159. doi: 10.18608/hla17.013.

Wood, B. (2020) *Adobe XD Classroom in a Book (2020 release)*. Adobe Press.

Wride, M. (2017) *Guide to Peer-Assessment, Academic Practice*.

Xie, Y., Ke, F. and Sharma, P. (2008) ‘The effect of peer feedback for blogging on college students’ reflective learning processes’, *Internet and Higher Education*, 11(1), pp. 18–25. doi: 10.1016/j.iheduc.2007.11.001.

Yacob, A. and Saman, M. Y. M. (2012) ‘Assessing Level of Motivation in Learning Programming Among Engineering Students’, in *The International Conference on Informatics and Applications (ICIA2012)*, pp. 425–432.

Yang, Y.-F. and Meng, W.-T. (2013) ‘The effects of online feedback training on students text revision’, *Language Learning & Technology*, 17(2), pp. 220–238.

Yen, W.-H. and Chang, C.-C. (2018) ‘Attitude Toward Online Peer-Assessment Activity: Experiential Learning Theory Viewpoint’, in *International Conference on Innovative Technologies and Learning*. Springer, Cham., pp. 61–70.

Yu, Y. H., Hu, Y. N. and Zhang, J. S. (2013) ‘Vygotsky’s Zone of Proximal Development: Instructional Implications and Teachers’ Professional Development’, *Applied Mechanics and Materials*, 411–414(4), pp. 2952–2956. doi: 10.4028/www.scientific.net/AMM.411-414.2952.

Zheng, L. *et al.* (2019) ‘A Systematic Review of Technology-Supported Peer Assessment Research: An Activity Theory Approach’, *International Review of Research in Open and Distributed Learning*, 20(5), pp.168-191. doi: <https://doi.org/10.19173/irrodl.v20i5.4333>.

Zimmerman, D. and Zumbo, B. (1993) ‘Rank transformations and the power of the Student t-test and Welch t’-test for non-normal populations’, *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale*, 47(3), p. 523.

Zoom Website (2022) *Zoom Website*. Available at: <https://zoom.us/> (Accessed: 15 June 2021).

## Appendix A: Ethics approval

5/6/2018

Ethics Form Completed for Project: Improve Computer... - Amal Alkhalifa (PGR)

### Ethics Form Completed for Project: Improve Computer Programming Skills through Automated Peer Assessment

Policy & Information Team, Newcastle University <noreply@limesurvey.org>

Tue 4/10/2018 6:55 PM

To: Amal Alkhalifa (PGR) <A.K.A.Alkhalifa2@newcastle.ac.uk>;

Ref: 4707/2018

Thank you for submitting the ethical approval form for the project 'Improve Computer Programming Skills through Automated Peer Assessment' (Lead Investigator: Amal Khalifa A Alkhalifa). Expected to run from 08/01/2018 to 26/12/2022.

Based on your answers the University Ethics Committee grants its approval for your project to progress. Please be aware that if you make any significant changes to your project then you should complete this form again as further review may be required. If you have any queries please contact [res.policy@ncl.ac.uk](mailto:res.policy@ncl.ac.uk)

Best wishes

Policy & Information Team, Newcastle University Research Office

[res.policy@ncl.ac.uk](mailto:res.policy@ncl.ac.uk)

## Appendix B: Consent form for participation with focus group

I am Amal Alkhalifa, currently studying PhD Computing at Newcastle University. I am currently designing a prototype model of peer assessment for introductory programming course.

Peer assessment is a process in which learners consider the level, quality, or success of their peers with similar statuses, products, or learning outcomes. Peer assessment shows learners how other peers solve problems while also encouraging them to think more critically and deeply and criticize others constructively. Peer assessment is not only a tool that supports constructive feedback for peers, but it is also a learning tool. Therefore, the goal of this focus group to find an appropriate way to design peer assessment for novice programmers. Because your opinion is valued, and you are already finished programming courses, I required your assistance to participate in this focus group. I need to understand what tasks you want and need in this suggested prototype. This will help me make sure that the prototype is designed to meet programmers wants and needs. The information you will provide will help to decide how to improve the current prototype. In addition, this research is confidential and anonymous. All data conquer to data protection act and data policy. Thank you for your interest in taking part in this research.

**Please read the following instructions and confirm consent:**

Please initial box to confirm consent		
1.	I confirm that I have read the information sheet dated 09/2020 for the above study, I have had the opportunity to consider the information, ask questions and I have had any questions answered satisfactorily	
2.	I understand that my participation is voluntary and that I am free to withdraw at any time without giving any reason. I understand that if I decide to withdraw, any data that I have provided up to that point will be omitted.	
3.	I consent to the processing of my personal information [Department, Studying level, Programming level, peer assessment experience] for the purposes of this research study.	
4.	I understand that all information will be treated with confidence and will be disposed of on (1/1/2023).	
5.	I understand that my research data may be published as a report, and my personal information will be anonymised	
6.	I consent to be audio recorded and understand that the recordings will be stored anonymously on the researcher's computer and used for research purposes only.	
7.	I understand that being audio recorded is optional and, therefore, not necessary for my participation in this research.	
8.	I agree to take part in this research project.	

<b>Name of participant:</b> .....	<b>Department:</b> .....	<b>Studying Level:</b> .....
<b>Programming experience:</b> <input type="radio"/> Novice <input type="radio"/> Competence <input type="radio"/> Proficiency	<b>Have you ever done peer assessment before?</b> <input type="radio"/> Yes <input type="radio"/> No	<b>Signature:</b> .....

## Appendix C: An example of students' assessment

**مخطط تصحيح البرنامج**

اسم المعلم الثاني: .....  
 القسم: .....  
 هل قمت بتقييم كود برمجى قبل ذلك؟  نعم  لا

اسم المعلم الاول: .....  
 القسم: .....  
 هل قمت بتقييم كود برمجى قبل ذلك؟  نعم  لا

التعليق الاول: لا حتى تقييم الكود البرمجى بناءا على المعايير القياسية المتكرومة اثناء من خلال الاجابة على جميع الاسئلة التالية ، واسماء التقنية الراجعة في مربع النص التالي.

التعليق الثاني: لا يوجد في البرنامج اخطاء منطقية

التصنيف	المعلم	المعلم الاول		المعلم الثاني	
		نعم	لا	نعم	لا
صحة الكود Correctness	البرنامج ينتج جميع المخرجات بشكل صحيح بناء على المدخلات المطلوبة لا يوجد في البرنامج اخطاء منطقية او منطقية	✓	-	✓	✓
بنية البرنامج Structure	الاختيار الصحيح لارواح المتغيرات وروائل البيانات (مثلا linked list, Array, loop, Case statement) يستخدم الطرق المختصرة في البرنامج بدلا من التكرارات (مثلا loop, statement)	✓		✓	✓
وضوح البرنامج Clarity	اسماء المتغيرات توضح قيم تستخدم وليست مبهمه البرنامج سلس وقابل للفهم، حيث تم استخدام القدرات اللغوية بفعالية		✓	✓	✓
المظهر Layout	استخدمت تعليقات الكود في اجزاء مختلفة وبشكل مناسب (مثلا header, line comments)		✓		✓
الاستعارات Exceptional	استخدمت حالات الخروج من العملية بشكل مناسب وبالوصلا (break; statement)	✓	✓	✓	✓

ماتني فله البرميج بشكل صحيح؟  
 كتابته: .....  
 ماتني فله البرميج بشكل خاطئ؟  
 كتابته: .....  
 كيف تحسن في المستقبل؟  
 .....  
 اذ يستعمل ..... الكود برمجى حسب المطلوب

ماتني فله البرميج بشكل صحيح؟  
 كتابته: .....  
 ماتني فله البرميج بشكل خاطئ؟  
 كتابته: .....  
 كيف تحسن في المستقبل؟  
 .....  
 اذ يستعمل ..... الكود برمجى حسب المطلوب

## Appendix D: An example of entering data into SPSS

Visible: 85 of 85 Variables

Group	Firstassessor	experience	peerassessment	A1	A2	A3	B1	B2	C1	C2	D1
10	العبد الممتحن	Competence	No	No	No	No	Not applicable	Not applicable	Yes	Partly	No
11	رديع العمود	Novice	No	Partly	No	No	Yes	I don't know	Yes	Partly	No
12	روداني الفانيز		No	Yes	Partly	Partly	Not applicable	Not applicable	Yes	Partly	Not applicable
13	سلمى النوسري	Novice	No	No	No	Partly	Partly	No	Partly	No	No
14	ديما النور	Novice	No	No	Partly	Partly	Yes	No	Yes	No	No
15	ديما السبيغ	Competence	No	Partly	I don't know	No	I don't know	Not applicable	Yes	No	Partly
16	رنا الحريسي	Novice	No	No	No	No	Yes	Yes	Yes	Partly	No
17	علاء الحارثي	Novice	No	Partly	Partly	No	Yes	No	Yes	Yes	Partly
18	أسراء البيهسي	Competence	No	No	No	Partly	Not applicable	Yes	Yes	Partly	No
19	عتاة الطيف	Novice	No	No	No	No	Yes	No	Yes	Yes	No
20	لروي الحماد	Novice	No	No	No	Partly	Not applicable	Not applicable	Partly	No	No
21	لين الوكيل	Novice	No	Partly	Partly	I don't know	I don't know	I don't know	Yes	Partly	Not applicable
22	لمى الشريف	Novice	No	I don't know	I don't know	Partly	Yes	Not applicable	Yes	Partly	Not applicable
23	ها المجهي	Novice	No	Partly	No	Not applicable	Yes	Yes	Yes	Partly	Not applicable
24	بنور العتيبي	Competence	No	Partly	Partly	Partly	Yes	Not applicable	Not applicable	No	No
25	هديل السعدون	Competence	No	No	No	No	Yes	Not applicable	Not applicable	Partly	No
26	لمى الأسدي	Novice	No	No	No	No	Yes	Not applicable	Yes	No	No
27	لمى هسيس نصر الله	Novice	No	No	No	No	Yes	Not applicable	Yes	No	No
28	آلاء العبدالله	Competence	No	No	Partly	Not applicable	Partly	I don't know	Yes	Not applicable	Not applicable
29	نادية الوبيطي	Novice	No	No	No	No	Not applicable	Not applicable	Yes	No	No
30	زران السبيبي	Novice	No	No	No	Yes	Partly	Not applicable	Partly	No	Not applicable
31	روان التواني			No	No	No	Not applicable	Not applicable	No	No	No
32	لمى الزهراني	Competence	No	No	No	No	Not applicable	Not applicable	Yes	Yes	No
33	سديم السبيبي	Novice	No	No	No	No	No	Not applicable	Yes	Yes	Not applicable
34	أسيل السعدون	Novice	No	No	No	No	Not applicable	Not applicable	No	No	Not applicable
35	حنى الفوران	Novice	No	Partly	No	No	I don't know	Not applicable	Yes	Partly	No
36	رانيا المقدم	Novice	No	No	Not applicable	No	Not applicable	Not applicable	Yes	Not applicable	Not applicable
37	لمى النوسري	Novice	No	No	No	No	Not applicable	Not applicable	No	No	Not applicable
38	سارة الهويش	Novice	No	No	Partly	No	Not applicable	Not applicable	Yes	No	No
39	أمل الروبيبي	Competence	No	No	Partly	Not applicable	Partly	Not applicable	Partly	Not applicable	No
40	الميرال المشعل	Novice	No	No	No	No	I don't know	Not applicable	Yes	No	No
41	ممد التطفلي	Novice	No	No	No	No	Partly	Not applicable	Yes	No	No
42	مرام العتيبي			No	No	No	I don't know	Partly	Yes	No	No
43	رحمة القسطاني	Competence	No	No	Partly	No	Not applicable	Not applicable	Yes	No	Not applicable
44	خديا الربيعان			No	No	Partly	No	Yes	No	No	No

## Appendix E: An example of students' feedback

### نتيجة تقييم سوسن العتيبي

#### النتيجة الأولى

هل تقييمك على نفس مستوى متوسط تقييم شعبتك؟

Is your assessment at the same level as that of your peers?

Class grade accuracy Average Vs. individual student average

مقارنة بين متوسط درجة الطالبة ومتوسط درجة الشعبة



قومي باختيار الخيارات المناسبة بناءً على الرسم البياني السابق (ملاحظة: يمكنك اختيار أكثر من خيار في كل فقرة)

كيف يمكن أن تتصني في المستقبل؟	مالفات التي قيمتها بشكل أقل من زميلاتك؟	مالفات التي قيمتها بشكل أفضل من زميلاتك؟
<input type="radio"/> Focus on Correctness <input type="radio"/> Focus on Structure <input type="radio"/> Focus on Clarity <input type="radio"/> Focus on Layout <input type="radio"/> Focus on Exceptional <input type="radio"/> None of them	<input type="radio"/> Correctness <input type="radio"/> Structure <input type="radio"/> Clarity <input type="radio"/> Layout <input type="radio"/> Exceptional <input type="radio"/> None of them	<input type="radio"/> Correctness <input type="radio"/> Structure <input type="radio"/> Clarity <input type="radio"/> Layout <input type="radio"/> Exceptional <input type="radio"/> None of them

#### النتيجة الثانية: ماذا عن التعليقات النصية؟ هل تشبه رأي معلمك؟

وجه التشابه بين وجهة نظر المعلم في أخطاء البرنامج ووجهة نظرك	أخطاء البرنامج من وجهة نظر المعلم
<p>Correctness</p> <p>PrintObject</p>	<p>Correctness</p> <p>Layout</p> <p>PrintObject</p> <p>TeacherClass</p> <p>AddComments</p>

قومي باختيار الخيارات المناسبة بناءً على الرسم السابق (ملاحظة: يمكنك اختيار أكثر من خيار في كل فقرة)

كيف يمكن أن تتصني في المستقبل؟	مالفات التي لم تغطيها في ملاحظتك النصية؟	مالفات التي قمتي باكتشاف أخطائها بشكل صحيح؟
<input type="radio"/> Focus on Correctness <input type="radio"/> Focus on Structure <input type="radio"/> Focus on Clarity <input type="radio"/> Focus on Layout <input type="radio"/> Focus on Exceptional <input type="radio"/> None of them	<input type="radio"/> Correctness <input type="radio"/> Structure <input type="radio"/> Clarity <input type="radio"/> Layout <input type="radio"/> Exceptional <input type="radio"/> None of them	<input type="radio"/> Correctness <input type="radio"/> Structure <input type="radio"/> Clarity <input type="radio"/> Layout <input type="radio"/> Exceptional <input type="radio"/> None of them



## Appendix F: Question form that distributed in the first focus group

<b>Pleas identify the characteristics you would like to find it in peer assessment system</b>
Idea no.:
Idea name:
Describe it:
What is the important to you?

<b>Pleas identify the kind of information do you need to represent as <u>a feedback</u> from the ideal peer assessment system?</b>
Idea no.:
Idea name:
Describe it:
What is the important to you?

<b>Please write your attitudes toward these charts as <u>a feedback</u>?</b>
Figure No.      (Like/Dislike) - comment:
Figure No.      (Like/Dislike) - comment:
Figure No.      (Like/Dislike) - comment:
Figure No.      (Like/Dislike) - comment:
Figure No.      (Like/Dislike) - comment:

<b>Pleas identify the tasks you would like to perform with the ideal system</b>
Idea no.:
Idea name:
Describe it:
What is the important to you?

<b>Pleas identify the desired outcomes you may get it from peer assessment system</b>
Idea no.:
Idea name:
Describe it:
What is the important to you?

## Appendix G: Question form that distributed in the third focus group

4

### Questions of the third Focus group

The following questions have been created to catch-up your opinion regarding the prototype. I kindly ask you to participate voluntarily by completing these questions.





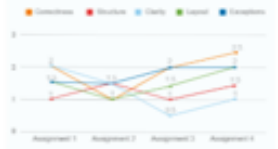
Features	<p>What features most caught your attention?</p> <p>.....</p> <p>.....</p> <p>.....</p>																																			
	<p>What would you expect in terms of skill and ability from someone peer assessing you?</p> <p>.....</p> <p>.....</p> <p>.....</p>																																			
	<p>Do you think the task difficulty level should determine who gets assigned to be your reviewer? (Yes/ No)</p> <p>Who can assign this difficulty level? (Author itself/ Peers/ All of them)</p>																																			
	<p>Would you like the following output of matching: The process of matching depends on an author's <u>need</u>. If you are not confident of your solution, you need at least two proficient reviewers, if you are confident to your solution, one proficient reviewer is enough? (Yes/ No)</p> <p>Please rate the following points for suggested matching process based on your personal value.</p> <table style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th></th> <th>1</th> <th>2</th> <th>3</th> <th>4</th> <th>5</th> <th></th> </tr> </thead> <tbody> <tr> <td>Comfort</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>Not Comfort</td> </tr> <tr> <td>Useful</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>Not Useful</td> </tr> <tr> <td>Important</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>Not Important</td> </tr> <tr> <td>Fair</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>Unfair</td> </tr> </tbody> </table> <p>Are there any changes you would suggest to the matching process? (Yes/ No)</p> <p>If Yes, Outlined these proposed changes.</p> <p>.....</p> <p>.....</p> <p>.....</p>		1	2	3	4	5		Comfort	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Not Comfort	Useful	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Not Useful	Important	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Not Important	Fair	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Unfair
		1	2	3	4	5																														
Comfort	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Not Comfort																														
Useful	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Not Useful																														
Important	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Not Important																														
Fair	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Unfair																														
<p>Please rate figures that displayed in the prototype based on the following aspects.</p> <div style="display: flex; justify-content: space-around;"> <div style="text-align: center;">  <p>Figure 1</p> </div> <div style="text-align: center;">  <p>Figure 2</p> </div> <div style="text-align: center;">  <p>Figure 3</p> </div> </div> <div style="display: flex; justify-content: space-around; margin-top: 20px;"> <div style="text-align: center;">  <p>Figure 4</p> </div> <div style="text-align: center;">  <p>Figure 5</p> </div> </div>																																				

Figure	Importance	Usefulness	comfort
Figure 1	Important <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> Not Important	Useful <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> Not Useful	Comfort <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> Not Comfort
Figure 2	Important <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> Not Important	Useful <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> Not Useful	Comfort <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> Not Comfort
Figure 3	Important <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> Not Important	Useful <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> Not Useful	Comfort <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> Not Comfort
Figure 4	Important <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> Not Important	Useful <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> Not Useful	Comfort <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> Not Comfort
Figure 5	Important <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> Not Important	Useful <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> Not Useful	Comfort <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> Not Comfort

Are there any changes you would suggest to any figure? (Yes/ No)  
If Yes, Outlined number of figure and proposed changes.  
.....  
.....  
.....

<b>Obstacles</b>	What are the prototype's key issues/concerns? ..... .....
	How significant is the issue or concern you have with this prototype? <span style="float: right;">(High / Medium / Low)</span>

<b>Improvement</b>	If you could choose a task of this prototype to eliminate, what would you choose? ..... .....
	If you could choose a task of this prototype to develop further, what would you choose? ..... .....
	If you could add any task to our prototype, what would it be? ..... .....

<b>Usage</b>	Do you expect to use peer assessment with programming courses? Why? <span style="float: right;">(Yes/ No)</span> ..... .....
--------------	--

Thank you for your interest in taking part in this study

## Appendix H : An example of open coding

### Initial Coding

Where we successful in this prototype? What features most caught your attention?

	Coding	Transcripts
Group 1	<p>2 peer assessment</p> <p>1 Matching technique</p> <p>3 Historical Progress</p> <p>1 Clear Criteria</p> <p>4 Charts</p> <p>1 Anonymous</p> <p>1 Satisfaction</p> <p>1 Teacher feedback</p> <p>1 Resubmit</p> <p>1 Data source</p>	<p><b>النورق: 1-</b> صفحة تقدم الطالب، وتصنيف نقاط ضعفه وقوته.</p> <p><b>2-</b> مقارنة درجاته بمعدل درجات أقرانه (الكاتب) تفاصيل دقيقة ومفيدة من جهة كالكود والمقيم عرض الرسم البياني للطالب لمعرفة تفاصيل تقييمه و هذا يساعد الطالب على التطوير.</p> <ul style="list-style-type: none"> <li>• النموذج يحفز ثقة الطالب بنفسه حيث انه من خلاله يلعب دور المقيّم</li> <li>• يوسع مدارك الطالب باطلاعه على شخّيف الطول</li> <li>• يقوم النموذج بقياس مستوى تقدم الطالب مما يجعل الطالب يلمس تقدمه من فترة لفترة ويحفزه ذلك على تحقيق مزيد من التقدم</li> <li>• يوضح النموذج مستوى الطالب مقارنة بقرّنه وهذا أيضاً عامل محفز للتحقيق مزيد من التقدم</li> </ul> <p><b>احتواء التقييم على أغلب عناصر التقييم البرمجية الأساسية،</b> تنوع الرسوم البيانية وشمولها (خاصة رسم ٣ لتوضيح صحة التقييم في حال توافق المقيمين)، ميزة تقييم المقيم، إمكانية التقييم من معلم المقرر، إخفاء هوية المقيمين.</p> <p><b>طريقة الموافقة بين المقيّم والمقيم،</b> إمكانية رؤية تطور المبرمج، إمكانية زيادة عدد المقيمين حسب ثقة المبرمج بجله</p> <p><b>التسجيل:</b> كنت متخيلته بس من جهة المؤلف، لكن أعجبتني أنه ينفع المؤلف والمقيم والمقيمين الباقين، البرنامج ليس من جهة واحدة بل هو من <b>other sides</b> ، المقيمين يحددون بناء على مدى تفتي بالكود لو كنت احتاج مساعدة راح يعطيني داس أكثر فهما مني، أعجبتني <b>progress</b> لأنه يوضح لي التاريخ حتى من اول <b>assignment</b> إلى آخر شي يوضح وين تطورت وبيّن نزل مستواي، المعايير مخطّية جوانب الكود الأساسية ومفهوم وبسيطة، انا أعجبتني <b>progress</b> كذلك مو قاعد بقميني على <b>assignment</b> واحد يظهر مجموعة ويظهر التطوير وفيه تفاصيل كثيرة بالرسومات تفيد الطالبات مثلا كل مقيم ايش أعطاه وهل اتفقوا أو لا، <b>chart</b> المقارنة بين <b>reviewer</b> إذا اتفقوا أو لا، عجبني الرسم البياني رقم ١. وشن فائدة <b>sign in</b> سهل يحفظ <b>history</b> حتى ويجمع بياناتي والا فقط دخولي للكليّة، حلو ان البرنامج فيه <b>data</b> مرررة كثير تناسب تحليل الطالب &lt;مافيه <b>resubmit</b> مرة وحدة زيادة ومايحتاج <b>review</b>، مقارنة درجاته بمعدل درجات أقرانه (الكاتب)</p>
Group 2	<p>1 Clarifying Criteria</p> <p>Visual chart feedback</p> <p>1 progress</p> <p>2 peer assessment</p> <p>4 colors</p> <p>self-assessment</p> <p>3 easy to use</p> <p>3 Design</p> <p>1 Matching technique</p>	<p><b>النورق:</b> وضوح الواجهة – الألوان</p> <ul style="list-style-type: none"> <li>- وضوح النموذج والمعايير</li> <li>- تناسق التصميم والألوان</li> </ul> <p>فكرة المشروع رائعة</p> <p><b>يجمع بين البساطة بالألوان وتناسقها والحدّات بالشكل ، مريح للعين ببساطة ، فكرة التقييم الذاتي ومعرفة مستوى التطور (Progress).</b></p> <p>صفحة <b>feedback</b> جدا واضحة ومفهومة، طريقة التقييم عادلة وسهلة الفكرة جديدة ومفيدة جداً ، وضوح استخدامه والوانه الهادئة ،</p> <p><b>التسجيل:</b> حلو اني اعرف المعايير اللي تقميني عليها الأستاذة لأنني بعض الأحيان احسها عطليتي درجة أقل من مستواي وأوضحت لي المعايير، في البرنامج وضحت المعايير، أنا حبيت <b>feedback</b> ومرة مطلعتها بشكل كويس وواضح، عشانها <b>visual and interactive</b>، حبيت انه يبين <b>progress</b>، وأنا احس انه شي جديد بيخلي الطالب يحتمسون، شي شديد مافد موته لنا معلمة يخليني اتحمس اني اضبط الكود أكثر لأنني بقيم نفسي وودي احط كل شي كامل مو يس أجل الكود واسلمه، الشكل نفسهم يح للعين وبسيط والألوان متناسقة الشكل أعجبتني، حبيت انه فيه رسوم بيانية وملونة وحلو المظهر وأسهل للقراءة من أنه نص، في التقييم الذاتي حبيت مرة السؤال المفتوح وشن أكثر شي حس الطالب انه انجزه بالكود،</p>

**Where we successful in this prototype? What features most caught your attention?**

- 14 Charts (9 general charts, 5 Historical Progress)
- 12 peer assessment
- 9 easy to use (usability)
- 8 Design (4 general design, 4 colors)
- 4 Matching technique
- 4 Clear Criteria
- 4 self-assessment
- 3 Anonymous
- 3 Details
- 2 Satisfaction
- 1 Resubmit
- 1 Data source for student

