Development and Validation of an Instrument for Evaluating Online Coding Tutorial Systems

Ohud Abdullah Alasmari

Submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy

School of Computing Science

College of Science and Engineering
University of Glasgow

2024

Declaration

I, Ms Ohud Abdullah Alasmari, hereby declare that this PhD research, including the primary work, was carried out by me, for the award of a PhD degree in the School of Computing Science, University of Glasgow. It has not been accepted nor currently submitted in the candidature for any other degree.

Dedication

To the memory of my beloved father.....

Acknowledgements

The first and foremost acknowledgement goes to my PhD supervisor, **Dr. Jeremy Singer**. I would like to express my deepest gratitude to him for the invaluable guidance, unlimited support, and continuous encouragement I received throughout my research journey. The constant weekly meetings in person and virtually over Zoom during the COVID-19 pandemic have helped me to learn research and move forward. I am very grateful for the opportunity to study under his supervision and for believing in me to complete my PhD study.

I also extend my gratitude to my second supervisor, **Dr Mireilla Bikanga Ada**, for her comments and aid in shaping the research and producing work with high-quality research.

My deepest thanks go to my husband, **Ali**, and my children, **Saif** and **Alyaa**, for their unaccountable support and patience and for accompanying me to the United Kingdom during my PhD study. I am forever grateful for their unwavering support and love.

I would also like to extend my grateful thanks to my family in Saudi Arabia, including my mother, my sisters, and my brothers, for their support. This work would not have been possible without their unlimited support and constant encouragement during tough times.

I acknowledge the **Saudi Arabian Cultural Bureau** in the UK for providing the financial and logistical support that made this research possible. I am grateful for their support and for the opportunities they have provided me. I am grateful to the School of Computing Science at the University of Glasgow for its support. Furthermore, I would like to thank the academic community for sharing their knowledge and supporting me.

ABSTRACT

In the computing education field, online programming learning platforms have become increasingly popular. There are several reasons for the recent growth in online programming learning systems, including the challenges of learning to code associated with face-to-face learning approaches. However, drill-and-practice-style activities provide one solution that can help overcome this obstacle. Hosting coding exercises, tutorials, and quizzes via online platforms provides a scalable solution to this problem. This can be accomplished with a variety of online programming learning platforms, such as interactive coding platforms and massive open online courses. In this research, the focus is on online coding tutorial systems (OCTSs). A wide variety of free and open online coding tutorial systems provide a basis for interactive programming education at scale. The use of browser-based systems with automated feedback is popular in remote learning scenarios. In addition, these systems facilitate the practical development of software that forms an integral part of the learning process for programming learners. Such systems will only be effective if they address the challenges and learning needs of novice programming learners. **Therefore, the objective of this thesis is to develop and validate an instrument for evaluating online coding tutorial systems to support professional programming educators in evaluating and selecting the appropriate online coding tutorial systems for programming learning to ensure the successful delivery of programming education, effective interactive platform use, and consequent positive impacts on programming novice learners.**

In order to achieve this research goal, a design-based research methodology was employed, and four studies were carried out to develop an evaluation instrument that has been constructed through three stages and four design iterations. Within the exploration phase, three instrument design iterations have been accomplished. During the initial design iteration, a systematic review of literature was undertaken to recognise common challenges in programming learning and to create the first version of the instrument by identifying a series of features of online coding tutorial systems as items of the instrument that could facilitate more robust and efficient programming learning experiences. In the second design iteration, an online survey tool was utilised to gather input on interaction with online coding platforms and to suggest some new items in the instrument. During the third design cycle, an analysis was conducted on seven popular online coding tutorial systems to determine if they

included the identified features and to identify any features that might be present in these systems but are missing as items to be considered in the proposed instrument. Following this, a high-fidelity prototype of an online coding tutorial system was developed in the development phase. Lastly, during the fourth design cycle and the evaluation phase, users of the system prototype tested most of the features discussed in the developed evaluation instrument and suggested new system features as new items in the instrument. Following the instrument development process, a validation study involving experts in teaching programming was carried out to validate the proposed instrument. Additionally, a study to show that programming educators are able to use the developed evaluation instrument effectively in a realistic scenario was conducted.

To summarise, **the main contribution of this research study is the development and validation of an instrument for assessing online coding tutorial systems in the field of computing education. This instrument assists professional programming educators in the evaluation and selection of effective online coding tutorial systems for teaching programming to novice learners.**

Glossary

CSS: Cascading Style Sheets

DBR: Design-Based Research

FDM: Fuzzy Delphi Method

GBL: Game-Based Learning

GMDR: Generic Model for Design Research

HTML: Hypertext Markup Language

IDE: Integrated Development Environment

LMS: Learning Management System

MOOCs: Massive Open Online Courses

OCTSs: Online Coding Tutorial Systems

PBL: Project-Based Learning

REPL: Read-Eval-Print Loop

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Chapter Overview

This chapter is structured as follows: Section 1.2 offers a high-level overview of the background context of this thesis. Section 1.3 discusses the main motivation of this thesis and the research objectives. Section 1.4 provides the thesis statement; Section 1.5 outlines the new research contribution to the field of computing education; Section 1.6 presents the published articles; and finally, Section 1.7 presents an overview of the remaining chapters in this dissertation.

1.2 Introduction

In the digital-native 21st century, computer programming has become an essential skill, and with the increasing reliance on technology in every industry sector, coding skills are in high demand. Versatile developers are required to develop software systems, including web and mobile applications. However, gaining programming skills is not an easy process, it seems to be a challenging task for novice learners across the world [179]. It requires an understanding of logic and good problem-solving skills [150]. Therefore, some novice learners struggle with learning coding, as the high failure rate among those taking programming courses shows [86]. Many challenges have been faced by novice learners from diverse educational backgrounds in learning how to code [35]. For instance, some novices struggle with understanding the syntax of programming languages [177] [115]. Other novice programmers struggle with testing and debugging their code [70][135]. However, according to [155], some of the programming learning difficulties are associated specifically with face-to-face learning modes, such as the lack of facilities for practicing programming. Therefore, the need to provide coding exercises to let novices practice coding by using interactive tutorials

has increased [155] [65]. In response to this, many university students have signed up for extended online learning platforms to practice programming [31] [102]. In addition, during the COVID-19 pandemic, the use of such online platforms that offer interactive practice has increased because of the need to replace face-to-face learning with online learning [133]. This use of online platforms is not only apparent in the programming education field, but also in other disciplines [133].

In the field of programming education, a variety of online learning platform types have been used, such as interactive coding platforms and massive open online courses [108]. These platforms offer a variety of courses and tutorials that cover different programming languages and concepts. Interactive coding platforms, such as Codecademy [49] and FreeCodeCamp [72], provide hands-on coding experience through interactive exercises and projects. In contrast, Massive Open Online Courses, such as Coursera [54] and edX [67], offer courses from top universities and instructors around the world. These courses often include assignments and quizzes to help novice learners master programming concepts. Therefore, online programming learning platforms have become increasingly popular because they are an effective way to learn programming anywhere and anytime [117] [155].

Kim and Ko [108] classify these platforms into five categories: interactive platforms, web reference platforms, educational game systems, creative platforms, and massive open online courses. In this research, the main focus is on **Online Coding Tutorial Systems (OCTSs)**, which adopt many of the features that have been identified in Kim and Ko's first category of interactive platforms, along with some aspects of their creative platforms and MOOCs [108]. In addition, these systems are defined as free and open online coding tutorial systems that provide a basis for interactive programming education on a large scale.

The use of such web-based systems with automated feedback is popular in remote learning scenarios, and there are several current online coding tutorial systems, such as LearnPython [175], TryRuby [94], and TryHaskell [64]. LearnPython [175], for instance, provides a series of interactive coding tutorials that cover the basic concepts of Python programming. The tutorials include explanations of programming concepts, examples of code, and interactive exercises to help novice learners practice their coding. In addition, OCTSs provide a step-by-step approach to learning programming languages and concepts. Nevertheless, such platforms will only be effective and helpful if they address the challenges, problems, and learning needs of novice programmers.

To the best of our knowledge, in the literature of computing education, there is a lack of research that develops and proposes instruments or tools for evaluating online coding tutorial systems based on computing education literature and programming educators and learners perspectives. For instance, Kim and Ko [108] propose a framework to analyse several dimensions of the pedagogical effectiveness of online coding tutorials only from the learning

sciences and education literature. Novice learners and their educators were not involved in this system effectiveness analysis, and the instrument has not been validated.

To address this gap, **the major objective of this research work is to develop and validate an instrument for evaluating arbitrary online coding tutorial systems to support programming educators in selecting systems that meet their needs to ensure the successful delivery of programming learning, effective interactive platform use, and consequent positive impacts on programming novices.** To accomplish this objective, this work follows a design-based research methodology, going through three phases and four design cycles to develop and propose an instrument for evaluating online coding tutorial systems. In the first phase (the analysis phase), there are three design cycles. **In the first design cycle of the instrument,** a list of programming teaching and learning challenges will be identified through a systematic literature review to identify the main components of the instrument, and a set of features will represent the instrument items that could help in evaluating and selecting online coding tutorial systems with richer and more effective programming learning experiences. **In the second design cycle of the instrument,** an online survey will be used to collect feedback from programming educators and novice learners about the online coding platform interaction by using a specific current online coding tutorial system called 'LearnPython', and the initial instrument will be updated. **In the third design cycle of the instrument**, seven current online coding tutorial systems will be analysed to determine whether they offer the identified features in the second version of the instrument. In addition, these selected systems will be analysed to investigate any features that exist in them but are missing in the second version of the instrument for evaluating online coding tutorial systems.

In the second phase (the design and implementation phase), the online coding tutorial system prototype was designed and developed. Finally, in the third phase (the evaluation phase), the fourth design cycle was done. **In the fourth design cycle of the instrument**, an online survey instrument will be used to measure users' satisfaction and collect users' feedback about the system prototype and the elements of the third version of the instrument. Additionally, the users' satisfaction through analysing log data will be used to capture users' interactions with the system prototype. Finally, a validation study was performed to validate the proposed instrument. Additionally, a user case study was conducted to evaluate the use of the instrument as an assessment tool for such systems by the target audience.

1.3 Research Motivation and Objectives

The main motivation of this research is to help novice programming learners easily engage with programming languages through effective online coding tutorial systems that might meet their needs. **This will be accompanied by providing professional programming ed-**

ucators (who adopt online coding tutorial systems in teaching programming to novices) with a validated instrument to evaluate and select effective online coding tutorial systems. and that will be achieved through a list of the following sub-objectives:

- Identifying the main components of the evaluation instrument by identifying the programming learning challenges.
- Identifying the initial evaluation instrument' items by identifying the possible solutions to the identified challenges (the main components).
- Updating the initial evaluation instrument' items by identifying new features of online coding tutorial systems from system stakeholders' perspectives, i.e. novice learners and educators to be evaluated.
- Examine a set of current online coding tutorial systems to determine whether they provide the identified features presented in the instrument and whether any features that exist in the selected current systems are missing in the proposed instrument.
- Developing an interactive online coding tutorial system prototype that provides most of the identified features that exist in the evaluation instrument as items.
- Evaluating the system prototype with real users, capturing their interaction with the system prototype, and collecting their feedback and suggestions on the system to improve the instrument's items.
- Validating the proposed evaluation instrument by experts in the field of computing education.
- Conducting a study to investigate how the target audience (programming educators) find the use of the instrument to evaluate online coding tutorial systems.

1.4 Thesis Statement

The current work focuses on developing and validating an instrument for evaluating online coding tutorial systems that can be used by programming educators who deal with novice programmers to evaluate online coding tutorial systems and select the most effective systems for novices. Thus, the statement defines the scope of the research and motivates a set of specific research questions to guide the study.

Thesis Statement: Online coding tutorial systems should be designed and deployed in such a way that they satisfy novice learners' needs to ensure an effective interactive platform use, leading to the successful delivery of programming teaching by educators and consequent positive impacts for novice learners. Such effectiveness can be achieved by the adoption of a systematically constructed and validated instrument to support programming educators in evaluating and selecting the most appropriate online coding tutorial system for novices and their learning context.

1.5 Thesis Contributions

The primary contribution of this thesis is developing and validating an evaluation instrument for online coding tutorial systems to be used by professional programming educators (who adopt online coding tutorial systems to teach programming) to evaluate and select effective online coding tutorial systems to support novice learners (who use these online coding tutorial systems to learn programming languages).

1.6 Publications

In this research work, several peer-reviewed papers have been published at international conferences.

1. **The work described in Section 4.2 and Section 4.4 has been published in ICETC 2023, in this peer-reviewed article (Alasmari et al. [11]).**

   Ohud Alasmari, Jeremy Singer, and Mireilla Bikanga Ada. 2023. Do Current Online Coding Tutorial Systems Address Novice Programmer Difficulties? In: 15th International Conference on Education Technology and Computers (ICETC 2023), September 26–28, 2023, Barcelona, Spain. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3629296.3629333

2. **The work described in Section 4.5 has been published in EDUNINE 2024, in this peer-reviewed article (Alasmari et al. [10]).**

   Alasmari, O. A. F., Singer, J. and Bikanga Ada, M. (2024) Python OCTS: Design, Implementation, and Evaluation of an Online Coding Tutorial System Prototype. In: VIII IEEE World Engineering Education Conference (EDUNINE2024), Guatemala City, Guatemala, 10-13 March 2024 https://doi.org/10.1109/EDUNINE60625.2024.10500548.

3. **A research paper has been published, in COMPSAC 2024 in this peer-reviewed article (Alasmari et al. [9])**

   Alasmari, Ohud; Singer, Jeremy; Ada, Mireilla Bikanga. Online Coding Tutorial Systems: A New Category of Programming Learning Platforms. In: 2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC). IEEE, 2024. p. 2222-2227, Osaka, Japan, 2-4 July 2024 https://doi.org/10.1109/COMPSAC61105.2024.00356

4. **A research paper has been published, in iSTEM-Ed 2024 in this peer-reviewed article (Alasmari et al. [8]**

   Alasmari, O. A. F., Singer, J. and Bikanga Ada, M. (2024) Analysis of Research into the Teaching and Learning of Programming: An Updated Review In: The 9th International STEM Education Conference 2024 (iSTEM-Ed 2024), Thailand, 31 July- 2 Augest, 2024 https://doi.org/10.1109/iSTEM-Ed62750.2024.10663138

5. **A research paper has been accepted in ICETC 2024.**

   Alasmari, O. A. F., Singer, J. and Bikanga Ada, M. (2024) New Supportive Features for The Online Coding Tutorial Systems: The Learners and Educators Perspectives In: The 16th International Conference on Education Technology and Computers (ICETC 2024), Porto, Portugal, 18-21 September, 2024.

1.7 Thesis Outline

This dissertation contains eight chapters, and this section provides an overview of each chapter as follows:

- **Chapter Two: Background** This chapter will discuss the main concepts of this research, which are programming education, programming learning platforms, online coding tutorial systems, and instruments for such systems. In addition, this chapter examines existing studies in the area of online programming education.

- **Chapter Three: Research Methodology** In this chapter, the five research questions will be discussed, and the qualitative and quantitative research approaches used in each study conducted will be discussed. Moreover, the techniques and data analysis methods that will be implemented will be discussed, along with the analysis of the data and the desired results in the four design cycles, validation study, and case study.

- **Chapter Four: Instrument Development** This chapter will present the four design cycles that are conducted to develop the evaluation instrument in this thesis. Firstly,

an initial draft of the instrument for evaluating online coding tutorial systems was developed from the systematic literature review. Secondary, this chapter discusses programming learners' and educators' feedback on the initial instrument through an online survey instrument that will collect learner and educator feedback about online coding platform interaction. It will also present a revised draft of the instrument (if applicable). Thirdly, this chapter will present the results of the initial fact-finding studies (analysing current online coding systems). It will also present a revised draft of the instrument (if applicable). Lastly, this chapter discusses the development of the Online Coding Tutorial System prototype and the evaluation of the proposed instrument. This chapter is also concerned with the design cycles of the proposed instrument; it will present a revised and final version of the instrument (if applicable).

- **Chapter Five: Instrument Validation** This chapter will present the validation process of the developed evaluation instrument.

- **Chapter Six: Programming Educators' Experiences on the Instrument** This chapter will present a case study that was carried out to investigate how the intended audience found the use of the proposed instrument to evaluate online coding tutorial systems.

- **Chapter Seven: Discussions** This chapter will present the findings and discussion of the work performed in this research and the extent to which the research questions have been answered.

- **Chapter Eight: Conclusion** This chapter will present the conclusions, contribution to knowledge, limitations, and future directions.

Chapter 2

Background

2.1 Chapter Overview

The review in this chapter is to address specific research questions that guide this research study, and these research questions include:

- **What is the current state of programming education and the role of online programming learning systems?**
- **What are the existing instruments and frameworks proposed for online coding tutorial systems?**
- **Has previous research in the field of computing education proposed instruments for evaluating online coding tutorial systems?**
- **What are the research methodologies suitable for investigating online coding tutorial systems?**
- **Are there any identified gaps or challenges in the research on online coding tutorial systems?**

To answer these research questions, this chapter is structured as follows: It begins with an exploration of programming education concepts and online coding tutorial systems in Section 2.2; Section 2.3 highlights the instruments and frameworks proposed for online coding systems; Section 2.4 discusses the research methods used in computing education. Lastly, the chapter concludes with a brief summary in Section 2.5.

2.2 Programming Education

The topic of programming education is one that is expanding quickly and has attracted a lot of interest lately. In addition, effective programming education courses that may assist novice learners in developing the skills and knowledge necessary to thrive in this sector are becoming more and more necessary as the demand for qualified programmers in many industries rises. First, to develop computational thinking skills and empower people to create, comprehend, and manipulate computer programs, programming education is defined as the teaching and learning of computer programming concepts, skills, and problem-solving techniques [219]. This definition emphasises the significance of computational thinking, a fundamental idea in programming education. Computational thinking entails decomposing complicated problems into smaller, more manageable components and applying logic and algorithms to solve them [218]. Nonetheless, it has been observed that learning and teaching coding presents a number of challenges that both educators and beginners in programming face during their coursework [22][79][89]. For example, learning to program requires strong problem-solving abilities, and learning to code in traditional courses is difficult for those with weak problem-solving abilities [19][155].

Nevertheless, many researchers have been investigating the difficulties novice learners face in programming. For example, [156] identified the top programming learning challenges such as functions, error messages, and variables. In addition, [203] reported that the most common programming learning problems are the lack of ability to find errors, develop a program to solve a task, and modularise code using functions and procedures. Moreover, according to [155], functions and procedures were considered the most challenging when learning how to program. In addition, syntax errors are considered one of the barriers for programming novices, delaying the feedback provided to students about the logic of the code developed [60]. According to a study by [106], teaching programming to students through the use of interactive and visual aids increased their motivation and level of engagement. Students who were a part of a community of like-minded beginners were more likely to stick with their programming education, according to a different study [87]. Numerous attempts have been undertaken over the years [209] to address these issues. For instance, over the history of programming languages, a number of scholars have worked to create programming languages whose syntax is appropriate for novices. Programming languages such as Smalltalk, Pascal, Basic, HyperTalk, Logo, and many more are examples [163]. But learning difficulties with programming continue to exist. Even computer languages that are categorised as introductory languages continue to present difficulties for inexperienced programmers to learn. Most programming beginners find it difficult to understand and use many programming principles and structures in their own programming activities, as programming demands a specific way of thinking [118]. There are a few more significant factors that also

affect this state of affairs in programming education. For this reason, it is crucial to understand why beginners in programming choose to study the language [7]. It is also vital to determine what kind of learning style beginners in programming need to grasp particular ideas. To find the appropriate course of action that would address the recurring issues of programming novices, it is necessary to analyse the present ways that are used to teach them programming.

2.2.1 Approaches to programming education

Numerous strategies have been employed to assist educators and inexperienced students in overcoming challenges related to teaching and learning coding. One method, for example, is **Project-Based Learning (PBL)**, in which students of programming work on actual projects to acquire programming ideas. Project-based learning has been found to increase student motivation and involvement in a number of studies[105] [106]. This method also places a strong emphasis on inquiry-based, student-centred learning through the completion of a project or series of projects. It has been demonstrated to be a successful strategy for improving student performance in a range of subject areas, encouraging critical thinking and problem-solving abilities in beginner learners, and [26] [112]. However, studies have indicated that this strategy can be especially useful for advancing student learning in STEM (science, technology, engineering, and mathematics) sectors [30]. Additionally, studies have demonstrated its efficacy in fostering the growth of 21st-century competencies like ingenuity, inventiveness, and self-starting [30].

It can be difficult for educators to execute this strategy, despite all of its advantages. For inexperienced students, it necessitates continual help and direction in addition to a substantial amount of planning and preparation [112]. Teachers also need to be ready to modify their lesson plans to fit the needs of specific first-time students and to give continuous feedback and evaluations all the way through the project. Project-based learning in programming has been the subject of numerous research studies [74] [224] [114] [147]. The Code.org curriculum [51] is one example of an online project-based learning system for programming that is intended for K–12 pupils. Novice learners can work on a variety of projects in the curriculum, including making games, interactive storytelling, and animations. The projects are meant to be interesting and enjoyable, but they also give students a chance to practice their programming abilities.

The literature has also documented the use of **Peer Instruction Learning (PIL)** as a technique. It involves inexperienced programmers teaching basic programming ideas to one another. Peer instruction has been found to enhance student learning and retention in a number of studies [161] [194]. This strategy has been applied to raise student engagement and increase learning outcomes in a variety of educational contexts, not just computer science

education [194]. One of the key advantages of this method is that it encourages active learning and teamwork by allowing inexperienced programmers to share their knowledge and abilities with one another. Furthermore, because they can divide the effort and assist one another in problem-solving activities, it lessens the cognitive strain on individual students [24] [101]. But there are several drawbacks to this strategy as well. For example, it can be challenging to make sure that each student is contributing to the coding work and is equally interested. When there is a sizable skill or knowledge gap between the two learners, this can be especially difficult [24]. This approach's possible drawback is that not all students or learning environments will benefit from it. For instance, certain students can find it difficult to communicate or engage in social situations, or they could prefer to work alone. Despite these difficulties, research indicates that, when done properly, pair instruction can be a useful teaching strategy. The type of task, the instructor's level of support, and the calibre of the pairing are some of the factors that affect how effective it is [24]. Peer education in programming was found to enhance student performance and engagement in an introductory programming course, according to a study by Simon et al. (2010). Additionally, the study discovered that students were more likely to stick with computer science if they took part in peer training. According to a different study by Porter et al. (2011), peer instruction in programming was successful in raising student learning results in a course on data structures and algorithms. Enhancing code quality is one of pair programming's main advantages. Pair programming produced higher-quality code than individual programming, according to research by [217], since pairs could find flaws and problems more quickly. According to a different study by Hanks et al. (2011), pair programming improved code architecture and design because it allowed pairings to communicate and hone their concepts more successfully. Furthermore, one of the main advantages of pair programming is that it can enhance the quality of the code. Pair programming produced higher-quality code than individual programming, according to research by [217], since pairs could find flaws and problems more quickly. According to a different study by Hanks et al. (2011), pair programming improved code architecture and design because it allowed pairings to communicate and hone their concepts more successfully.

Game-Based Learning (GBL) is the third method in this section; it is a novel approach to teaching coding skills that blends game design ideas with coding education[157]. It uses games to teach programming topics. Its promise to boost learner motivation, engagement, and learning outcomes has led to its increasing popularity in recent years [104] [17]. It employs games to engage students and enrich their learning experience. According to a study by [195], game-based learning improved information retention and skill acquisition more than traditional education techniques. Another study by [25] discovered that students' skill development was significantly aided by the use of games in higher education. Furthermore, a study by Hainey (2011) evaluated the effectiveness of game-based learning in raising student

motivation and engagement levels as well as learning results. Furthermore, a number of studies have demonstrated how well coding game-based learning works to increase students' interest and coding proficiency. For instance, research by [104] discovered that students significantly improved their coding skills and demonstrated higher levels of engagement in the learning process when they created games using Scratch [185] as opposed to using standard coding instruction techniques. According to different research by [100], students' motivation and interest in coding increased when they used CodeCombat [50] as a teaching tool. Students' coding skills and problem-solving abilities also improved. A multiplayer game called CodeCombat [50] uses interactive challenges and puzzles to teach coding skills. Another illustration is the visual programming language Scratch [185], which enables pupils to make their own animations and games.

The fourth strategy, called **Online Learning**, teaches programming topics through the use of online resources. Studies have indicated that teaching programming using online means can be successful, particularly when paired with other strategies like project-based learning [20] [24]. Utilising online platforms has grown in popularity recently, especially in the wake of the COVID-19 pandemic [180]. The flexibility and accessibility of those online platforms make them ideal for teaching and studying programming [180]. Students can learn at any time and from any location as long as they have an internet connection [14]. Furthermore, through discussion boards, chat rooms, and other online resources, online platforms can give students greater chances for cooperation and communication [57]. However, when it comes to teaching and learning programming, the quality of the course materials and instructional design is crucial. Effective online learning environments frequently include a variety of assessment techniques, like peer review and formative assessments, which can support students in maintaining motivation and engagement [73]. For teaching particular programming topics, including software design and best practices for coding, online learning can be very useful [119]. Online learning proved to be an effective method of teaching these topics, which can be challenging to teach in traditional classroom settings, according to a study by [61].

Numerous online learning platforms are well-known in the field of programming education. One such platform is Codecademy [49], an interactive website that provides courses in a number of programming languages, including Python, JavaScript, and Ruby. Moreover, the interactive platform TryRuby [94] uses the Ruby programming language. Particularly during the COVID-19 pandemic, the usage of these systems has risen in recent years [5] [62]. Consequently, web-based learning systems are now more crucial than ever [117].

The majority of post-millennial students studying programming have given up on textbooks and other conventional learning materials and rely primarily, if not entirely, on online learning platforms [170]. Furthermore, employing the traditional face-to-face learning style in programming education is more expensive than using the virtual learning style, as per [144]. As a result, online solutions for self-learning have taken the place of in-person programming

courses at several universities [144]. Furthermore, as per [127], learners with varying educational backgrounds and programming experience can become programmers in a shorter amount of time and with less effort by utilising virtual coding environments, which is another advantage of utilising online platforms in programming education. In addition, such platforms support learners by facilitating the coding learning process by providing practicing programming problem-solving activities such as interactive tutorials, coding exercises, or quizzes [19][155]. According to a study by [19], for example, some struggling programmers require more extensive practical learning resources to help them through the learning process. These challenges are frequently surmountable through honing problem-solving techniques in programming through simple, appropriate, and accessible online programming learning systems that facilitate successful learning strategies, like immediate feedback or deep programming learning motivation [19][155]. Stated differently, the availability of efficient online learning platforms for programmeming that cater to the needs of learners and make it simple and convenient for them to learn and practice programmeming is crucial for learning to programme effectively [28] [108] [170].

2.2.2 A view of online programming learning systems

According to Kim and Ko [108], online systems in the programming education field can be divided into several categories. The first category is **Interactive Learning Platforms**; these platforms provide learners with interactive coding environments where they can practice coding and receive instant feedback. In addition, they are designed to make learning to code more engaging and interactive. One of the most popular interactive learning platforms for programming is Codecademy [49]. In addition, FreeCodeCamp [72] is a non-profit organisation that offers interactive coding challenges and projects to help learners build their coding skills. The platform offers courses in various programming languages, including HTML, CSS, and JavaScript. Moreover, Khan Academy [107] is another example of this category. It is an online learning platform that offers courses in various subjects, including computer programming.

The second category is **Web Reference Programming Learning Platforms**, these platforms are online platforms that provide learners with access to a wide range of programming resources, including tutorials, documentation, and code examples. The most popular web reference programming learning platform is W3Schools [211].

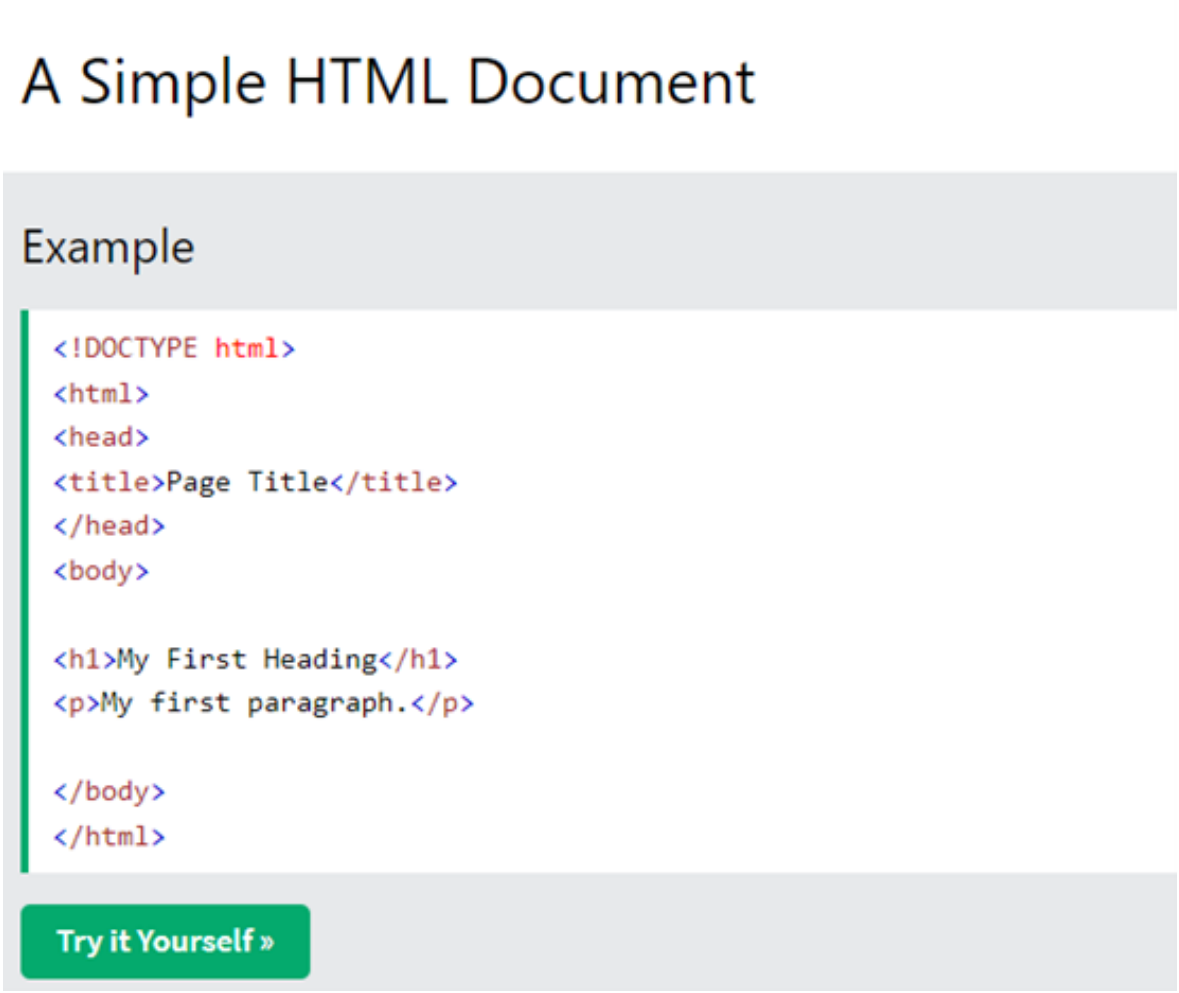

Figure 2.1: A screenshot of W3Schools' code editor

As shown in Figure 2.1, W3Schools is a web reference programming learning platform that provides learners with access to tutorials, documentation, and code examples for a wide range of web development technologies, including HTML, CSS, JavaScript, and SQL. The platform is free to use and is widely regarded as one of the best resources for learning web development. Several studies have investigated the effectiveness of W3Schools as a learning platform.

The third category is **Programming Educational Games Systems**; these systems are designed to teach programming concepts and skills through interactive games. They are becoming increasingly popular as they provide a fun and engaging way for learners to develop their programming skills [152]. There are several types of these systems; for instance, Puzzle-Based Games. This type requires learners to solve programming puzzles by writing code to complete a task. Examples of puzzle-based games include Lightbot, Code.org [51], and Blockly. In addition, Role-Playing Games are another type that allow learners to play a role in a virtual world where they must use programming skills to complete tasks and solve problems, such as CodeCombat [50].

The fourth category is **Programming Creative Platforms**, which are tools that allow users to create interactive and dynamic digital content using programming languages. These platforms are designed to make programming more accessible and engaging for users with different levels of programming experience. One of the most common programming creative platforms is Scratch [185], a visual programming language and online community developed by the MIT Media Lab.

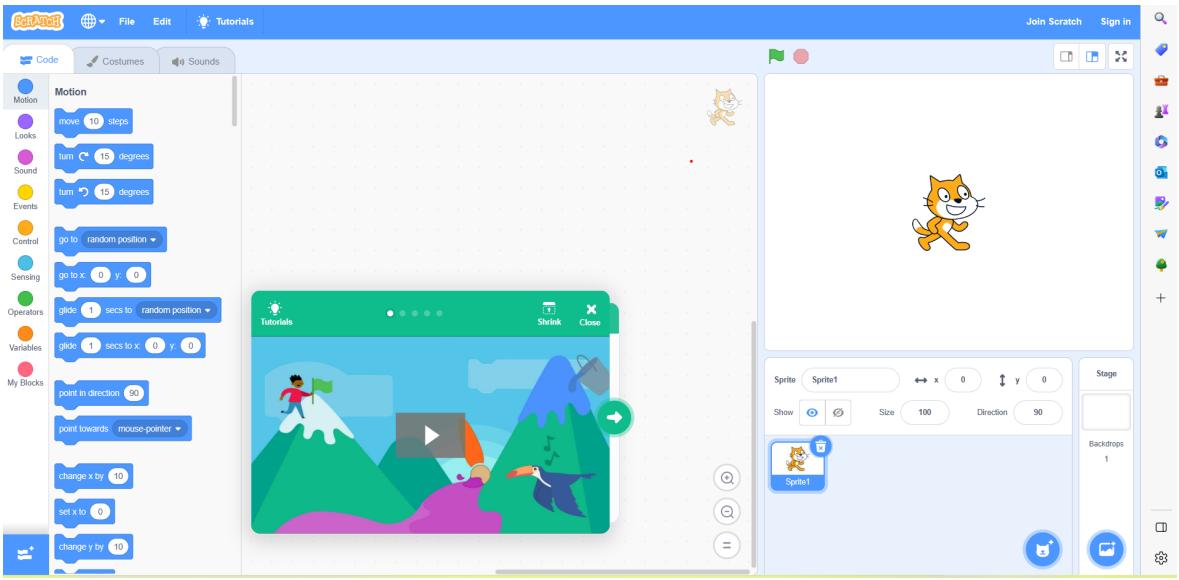

Figure 2.2: A screenshot of Scratch platform

As shown in Figure 2.2, it allows users to create interactive stories, games, and animations using drag-and-drop blocks [176]. In addition, Processing [162] is an open-source programming language and development environment designed for artists and designers. It allows users to create interactive graphics, animations, and visualisations using Java-based syntax.

The last category is **Massive Open Online Courses**, which are designed to be accessible to anyone with an internet connection. They are typically free or low-cost and offer a wide range of courses, including programming courses. One of the most common MOOCs for programming learning is edX [67].

2.2.3 Online coding tutorial systems

As discussed in the previous section, several online programming learning platforms were developed to facilitate learning and teaching programming. In this research study, the main focus is **Online Coding Tutorial Systems**, which adopt many of the features that have been identified in Kim and Ko's first category of interactive platforms, along with some aspects of their creative platforms and MOOCs [108].

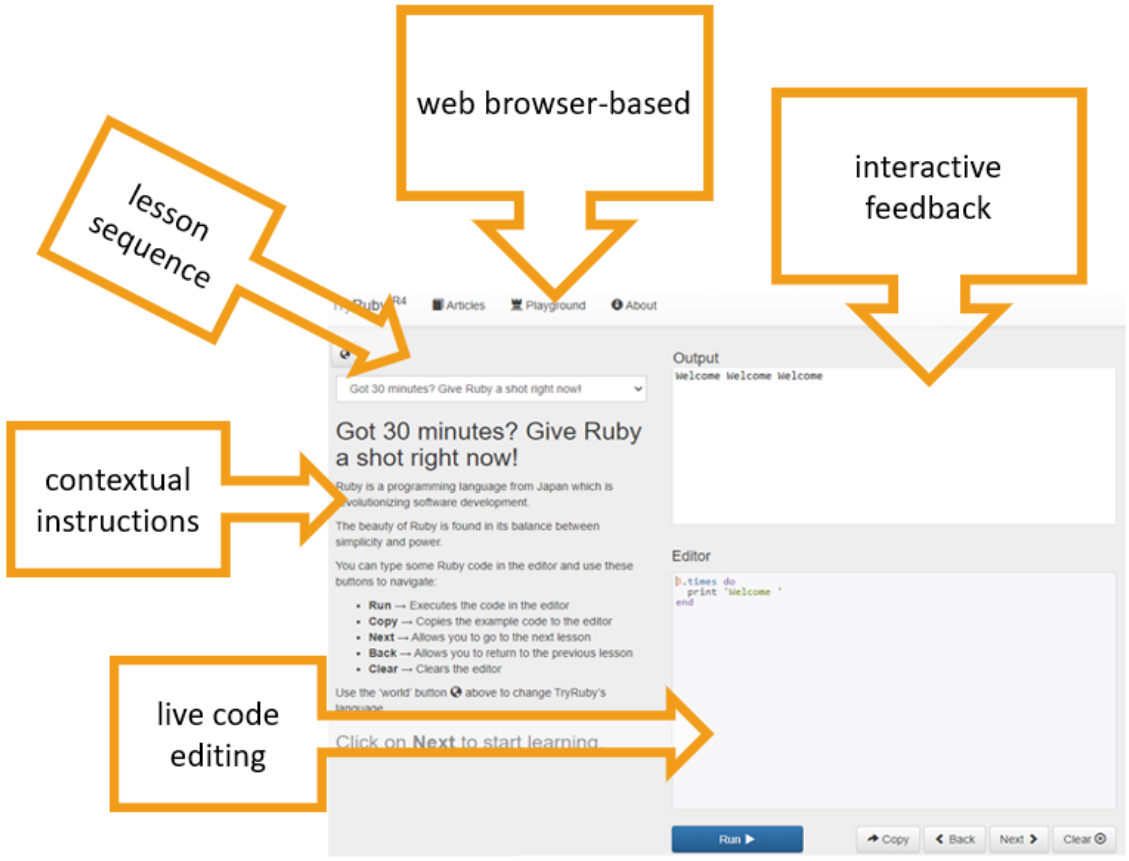

Figure 2.3: The components of an online coding tutorial system called “TryRuby”

There are several current online coding tutorial systems, such as LearnPython[175], Try-Haskell [64], TryJavaScript [159], and TryRuby [94]. As shown in Figure 2.3, such systems incorporate a set of online interactive services that provide learners with tools to support their programming learning journey.

An online interactive programming learning system teaches programming through a series of coding tutorials, and generally provides an interactive shell known as a Read-Eval-Print loop (REPL), where learners engage by typing fragments of source code into the text editor, running them, evaluating them, seeing the output, and compiling errors when appropriate [95]. These interactive systems provide a useful learning approach that allows learners to gain familiarity with a programming language very quickly [222]. A typical instance of the online interactive programming learning system concept is “TryRuby”. Figure 2.3 shows the interface of this system, which incorporates many of the characteristic features of online interactive programming learning systems, including:

- **Live code editing:** It is a text box for direct source code entry with syntax highlighting support. In addition, a toolbar with a ‘Run’ button facilitates the immediate execution of input source code.

- **Contextual instructions:** They are scaffolded, hyperlinked tutorials or reference guides visible on the main screen.
- **Lesson sequence:** It is a series of coding tutorials.
- **Web browser-based:** It is a web-based system that can be used through web browsers.
- **Interactive feedback:** It is an output box for interactive result display.

To define Online Coding Tutorial Systems (OCTSs), some features were adopted that have been identified in Kim and Ko's first category of interactive platforms [108], and some aspects of the fourth category creative platforms and the fifth category MOOC were integrated. Accordingly, online coding tutorial systems are defined as systems that teach programming through a series of structured coding tutorials as well as provide a useful programming learning approach [222]. Similar to the components of TryRuby shown in Figure 2.3, the most significant feature that an online coding tutorial system should provide is an interactive interpreter shell known as a Read-Eval-Print loop (REPL). Learners can practice coding by typing fragments of source code into the text editor, running them, evaluating them, seeing output, and getting errors when appropriate [95].

To summarise:

- Learning systems that provide novice learners with interactive feedback-driven tools to support their programming learning journey, provide a useful learning approach that allows learners to gain familiarity with a programming language quickly [222]. Therefore, we consider the pedagogical dimension in developing OCTS' evaluation instruments since these systems are considered learning systems.
- Web-based systems that must teach programming through a series of structured coding tutorials. Therefore, we consider the technical dimension in developing OCTS' evaluation instruments since these systems are considered web-based platforms.
- Web-based programming evaluation instrument that gives learners around the world that have different cultural backgrounds, the opportunity to take advantage of free, effective, and accessible programming education opportunities. Therefore, we consider cultural dimension in developing OCTS' evaluation instruments since these systems provide open programming educational resources and tools in the virtual world that enable learners to learn and practice programming easily and conveniently anywhere [28].
- Programming learning environments that should be analysed from a cognitive perspective rather than traditional aspects that are not sufficient to analyse programming learning environments [81].

2.2.4 Importance of online coding tutorial systems

The importance of these systems has increased in recent years, as they provide a convenient and easily accessible means for individuals at all skill levels to acquire coding proficiency. They offer learners a diverse range of resources, such as interactive tutorials, practice exercises, and discussion forums. In addition, interactive programming platforms have gained popularity as they provide hands-on coding experiences and immediate feedback to learners. Several studies have been conducted to investigate the effectiveness, user experiences, and learning outcomes of interactive programming platforms. For example, [151] examines the usability of interactive programming systems and identifies key design considerations for supporting novice programmers. Another study explores the impact of an interactive programming platform on student learning outcomes, engagement, and the effectiveness of automated assessment and immediate feedback in programming courses [154].

A growing body of research suggests that online coding tutorial systems can effectively facilitate programming learning. For instance, [15] conducted a study that demonstrated that due to the progress in cognitive psychology, artificial intelligence, and computer technology, it has become feasible to design computer systems that are comparable in effectiveness to human tutors. In addition, these systems bring numerous potential benefits for learners, including flexibility, affordability, personalisation, and collaboration. Most of these systems provide interactive features such as coding editors, instant feedback, and debugging tools. These features allow learners to practice coding in real-time, receive immediate feedback on their code, and debug errors. The interactive nature of these systems enhances learner engagement, reinforces understanding, and helps bridge the gap between theory and practical application [131]. Moreover, such systems offer a wide range of learning resources, including tutorials, videos, code examples, and documentation. Learners can access diverse learning materials that suit their preferred learning styles and needs. This variety of resources provides learners with multiple perspectives, explanations, and approaches to programming concepts, enhancing their understanding and learning outcomes [41]. Moreover, such systems support learners with different learning styles, they offer a variety of instructional materials, such as text-based tutorials, video lectures, interactive coding exercises, and visual demonstrations [178]. To summarise, online coding tutorial systems play a crucial role in facilitating programming learning and have the potential to have a significant impact on novice learners.

2.2.5 Users of online coding tutorial systems

Online coding tutorial systems have been used by novice programmers, students in programming courses, and learners seeking to learn coding independently. Firstly, **novice program-**

mers refer to learners who have limited or no prior experience in coding. They may be beginners who are exploring programming for the first time or learners transitioning from other domains [106]. These systems were designed for them since they emphasise foundational programming concepts, basic syntax, and problem-solving skills. Additionally, these online coding tutorials typically provide step-by-step guidance and interactive exercises to support learners in their initial coding journey. In addition, these systems are tailored for **students** enrolled in programming courses, both at the school and university levels. They provide them with additional resources, practice exercises, and opportunities for self-paced learning. On the other hand, these systems are also popular among **learners who learn coding for professional development**, these self-learners seek flexible learning resources that allow them to acquire coding skills at their own pace and convenience [106].

2.3 Instruments for Evaluating Online Programming Learning Systems

In terms of online programming learning systems, some studies have been conducted to analyse the design, features, and characteristics of such systems. For instance, Zinovieva et al. [230] published an article that discusses a comparative analysis of different online programming learning platforms for teaching programming according to specific criteria. However, they analysed different types of programming systems, not particularly online coding tutorial platforms or interactive coding systems. Moreover, Sim et al.[193] reviewed research on supporting novice programming, focusing on the implementation of programming environments that might be solutions for programming learning problems. Their study specifically focused on tools that support block programming and intelligent tutoring systems. However, their analysis of the systems was general; no systems' features were discussed.

In addition, [85], focusses on the Online Python Tutor, a web-based program visualisation tool for Python. This research explores the popularity of Python as a language for teaching introductory computer science courses. The tool enables users to write Python programs directly in a web browser and provides features for step-by-step execution visualisation and sharing program visualisations online. Addressing sociological barriers to programming is the primary objective of [106], which identifies the lack of a social context and compelling learning contexts as challenges in programming education. The study proposes solutions to address these barriers and highlights the importance of creating engaging environments for learning programming.

In [128], authors evaluate an instructional design for teaching Python and Java within the context of mobile application development. The study sheds light on effective pedagogical strategies that can be incorporated into online coding tutorial systems, providing valuable

insights for improving the instructional design of such systems. Although not directly focused on online coding tutorial systems,[4] investigates the patterns of debugging among novice computer science students. Debugging is a critical skill in programming and is often integrated into online coding tutorial systems. This study offers insights into how novices debug their code, which can inform the development of effective debugging features in online coding tutorial systems. Moreover, a study conducted by [108] and the current research on "Online Coding Tutorial System" differ not only in their methodologies but also in their specific areas of focus and objectives. [108]'s research primarily aimed to evaluate online coding tutorials using an analytical approach. Their focus was on analysing the content and instructional methods employed in these tutorials, assessing how they aligned with curriculum design dimensions. By taking this approach, they were able to comprehensively assess a wide range of tutorials and delve into aspects that may not be easily quantifiable.

Overall, while all the above studies contribute to the field of online coding education, they differ in their research objectives, methodologies, and areas of emphasis. However, to the best of our knowledge, no specific instrument or framework for evaluating online coding tutorial systems has been developed and validated in the literature of the computing education field. Therefore, it is significant to propose an evaluation instrument that could support professional programming educators in evaluating and selecting effective online coding tutorial systems for novices in this thesis.

2.4 Research Methodologies

Selecting research methodologies depends on the research questions, objectives, and nature of the study. Common research methodologies used in the computing education field include qualitative, quantitative, and mixed-methods approaches. For instance, qualitative research methodologies are frequently used in computing education to explore subjective experiences, perceptions, and social contexts. They involve collecting and analysing non-numerical data to gain in-depth insights into phenomena. For example, a qualitative study conducted by [213] explored students' experiences with pair programming in an introductory computer science course by conducting interviews and observations to understand the benefits, challenges, and impact of pair programming on student learning.

Another methodology has been used in the computing education field, which is a quantitative research methodology. In using this method, statistical and numerical analyses are employed to gather and analyse data in computing education. These approaches involve collecting structured data through surveys, tests, or experiments, allowing for statistical analysis to uncover patterns, relationships, and generalizability. For instance, a quantitative study conducted by [115] evaluated different approaches to teaching introductory programming.

In addition, mixed-methods research is another methodology that combines qualitative and quantitative approaches to provide a comprehensive understanding of research questions in computing education. It involves integrating data collection and analysis techniques from both qualitative and quantitative methodologies to address research objectives and provide a more complete picture. For example, a study was conducted by [228] to evaluate programming tools and measure the performance of the users. Another methodology that has been frequently used in the computing education field is design-based research [1] which is the research methodology that has been adapted in this research to develop the instrument for evaluating online coding tutorial systems. This research methodology involves iterative cycles of design, implementation, and evaluation of any learning interventions or technologies [16]. It emphasises the development of practical solutions while simultaneously generating theoretical insights.

2.5 Chapter Summary

This chapter has covered the introduction of the research, including the problem background and important concepts. Moreover, an investigation has been done to define those terms that have been mentioned above.

Chapter 3

Research Methodologies

3.1 Chapter Overview

The purpose of this chapter is to identify and justify the research methods that will be used in this research study to develop and validate the evaluation instrument for online coding tutorial systems. This chapter is structured as follows: Section 3.2 outlines the research questions addressed in this thesis. Section 3.3 presents the definitions of the research methodology; Section 3.4 discusses the design-based research methodology and why it has been used in this thesis; Section 3.5 discusses all the research methodologies that have been used and why they were used in this thesis to develop the instrument. Section 3.6 presents the timeline of this research project, concluding with a brief summary in Section 3.7.

3.2 Research Questions

In this thesis, several research questions will be answered to address the main claim, which is developing and validating an instrument to evaluate online coding tutorial systems that is formulated into one main research question **What instrument(s) are appropriate to evaluate any online coding tutorial systems?**

Firstly, **in terms of an evaluation instrument development, four research questions are addressed in Chapter 4:**

- **RQ1:** What are common programming learning difficulties for novices? And which supportive features are potential solutions for these identified difficulties?
- **RQ2:** What are appropriate supportive features in online coding tutorial systems from learner and educator perspectives?

- **RQ3:** What are the supportive features that exist in current deployed online coding tutorial systems but are absent from the instrument? Do the identified supportive features in the evaluation instrument exist in these systems?
- **RQ4:** Building on our research findings, what would an online coding tutorial system look like? Based on a prototype implementation, to what extent are typical learners satisfied with the features of such an online coding tutorial system?

Secondly, **in terms of instrument validation, one research question is addressed in Chapter 5**

- **RQ5:** To what extent is it applicable for programming educators to use the proposed instrument for evaluating online coding tutorial systems?

Thirdly, **in terms of the use of the instrument, one research question is addressed in Chapter 6**

- **RQ6:** What are the attitudes of programming educators toward using the instrument to evaluate online coding tutorial systems?

3.3 Research Methodologies– An overview

Research methodologies are the methodical approaches and strategies that scientists can use to gather, examine, and evaluate data in order to address a research question or go further into a certain topic [82]. A methodology provides an organised framework for the execution of research findings and ensures their validity, applicability, and reliability [82]. It also contains a variety of components, such as research design, data gathering tactics, data analysis techniques, and ethical issues [192]. First, the term ”research design” refers to the approach or blueprint that guides the entire research process [82]. It outlines the steps that must be taken to address the research question and accomplish the goals. It may employ mixed, quantitative, or qualitative approaches, depending on the nature of the research topic and the data required [164]. Second, in terms of data collection methods, these involve gathering information or data from relevant sources; examples of such methods are surveys, observations, interviews, experiments, and document analysis [21]. The choice of data collection strategy is influenced by the research question, the type of data needed, and the resources available [33]. Data analysis, which entails arranging, coding, and interpreting the data using suitable statistical or qualitative analytic techniques, can be used to examine the data after it has been acquired in order to derive meaningful conclusions [82]. Finally, ethical considerations that guarantee the secrecy, privacy, and protection of participant rights are also incorporated

into the research process [33]. Participants must give their informed consent before research begins, confidentiality must be maintained, and ethical standards established by pertinent organisations or bodies must be followed [116] [33].

3.4 Design-Based Research Methodology

The current thesis develops and suggests the intended evaluation instrument for online coding tutorial systems using a design-based research (DBR) methodology. Design-based research is defined as a methodology that integrates theory development and iterative design procedures to address any complex educational challenge [23] [172]. This methodology is especially prevalent in the education sciences since it is challenging to supervise educational interventions in real-world settings [172]. Additionally, these methods often demonstrate five key characteristics that characterise their approach to research and development in educational settings [16] [172]:

- **Firstly**, it focuses on real-world problems within authentic learning environments. In addition, it emphasises the relevance and applicability of research findings to the actual challenges faced by educators, students, and other stakeholders in the educational context [16] [172] [212].
- **Secondly**, it follows an iterative design process, which involves multiple cycles of design, implementation, and evaluation. Each iteration in the cycle allows for the refinement and improvement of the educational intervention based on feedback, data analysis, and insights gained from previous iterations [16] [172] [39].
- **Thirdly**, it emphasises collaboration between researchers and practitioners, such as teachers, instructional designers, or administrators. This collaboration ensures that the research is informed by practical expertise and that the developed interventions are relevant and feasible in real-world educational settings [16] [172].
- **Fourthly**, it contributes to the development and refinement of theoretical frameworks or models that explain the relationship between the designed interventions and their impact on teaching and learning. It combines theory development with practical application to create evidence-based solutions that can be implemented and tested in educational contexts [16] [172].
- **Lastly**, it employs a mixed methods approach, combining qualitative and quantitative data collection and analysis techniques. This allows for a comprehensive understanding of the educational intervention's effects, the underlying processes, and the

contextual factors influencing outcomes [158]. Therefore, this approach has been increasingly adopted and refined over the past few decades [16] [158].

In the field of computing education, this approach has been extensively utilised to develop either systems, frameworks, instruments, or models that might help in enhancing programming learning. For example, **in terms of developing programming educational systems**, Sengupta et al. [186] have used a design-based research approach to integrate computational thinking into middle school science curricula through the use of agent-based modelling tools. The authors in this study iteratively designed and refined both the software tools and the instructional strategies. Sengupta et al. [186] reported enhanced student engagement and improved understanding of complex scientific concepts. The findings show that the students were able to relate computational models to real-world phenomena, indicating a deep integration of computational thinking into their learning processes. While the work of Sengupta et al. [186] demonstrated significant benefits of using this approach to design educational tools, one limitation of the design-based research approach was the heavy reliance on researcher involvement in the classroom, which may affect the scalability and sustainability of the intervention. The context-specific nature of the findings also poses challenges for generalising the results to other settings without similar iterative refinements.

Additionally, this method was used in a different study by Kolling et al. [113] to create BlueJ, an integrated development environment (IDE) made especially to help inexperienced programmers. In order to improve the system, this study's iterative design method incorporated input from teachers and students. According to the authors, BlueJ improved students' motivation and involvement in learning programming by assisting them in comprehending object-orientated programming ideas more thoroughly. Although the system was highly accepted, the repeated cycles of the study were mostly carried out in controlled educational environments, which might have limited the findings' wider applicability. Additionally, the needs of students studying more sophisticated programming may not be immediately met by the tool's customisation for beginning learners.

Another study by Kelleher et al. [106] covered the application of design-based research methodology to create "Alice," a novel programming environment that uses drag-and-drop 3D graphics programming to make programming more approachable and entertaining for beginners. This method was used to build and improve the environment iteratively in response to user input and learning objectives. It was discovered that Alice greatly reduces the initial obstacles that new programmers face. The platform made it possible for new students to make games and animations, which gave them an engaging introduction to logic and programming ideas. Nevertheless, while Alice proved effective in engaging beginners and helping them understand basic programming constructs, its deviation from standard textual programming might not prepare students for more traditional programming environments

used in later educational stages or professional settings. The study also faced challenges in measuring long-term learning outcomes.

Using a design-based research methodology, Weintrop et al. [215] concentrated on creating and improving Blockly, an online visual programming editor, through iterative refinement. The tool created for this study used drag-and-drop coding blocks to visually depict programming ideas, making it easier for inexperienced programmers to learn. According to the study's findings, Blockly greatly decreased the syntax errors that are frequently made in traditional text-based programming, freeing up students' time to concentrate more on understanding logic and concepts. The tool was particularly effective for younger students and those new to programming. Nevertheless, despite Blockly helping to reduce the cognitive load associated with learning programming syntax, there are concerns about its long-term benefits. The simplified environment may hinder the transition to text-based programming, potentially limiting learners' preparedness for more complex programming tasks. Additionally, the study's focus was limited to initial learning phases, with less attention given to how skills transfer to more traditional programming environments.

On the other hand, **several studies have used design-based research to develop educational frameworks and instruments in programming education literature**. For instance, Brennan and Resnick [38] have employed a design-based approach to create a comprehensive framework for understanding and assessing computational thinking (CT) in educational settings. The study involved multiple iterations of designing educational activities using the Scratch programming environment, evaluating student work, and refining the activities based on observations and student feedback. The framework outlined dimensions of computational thinking, including concepts, practices, and perspectives, and provided a structured approach to assessing students' CT skills. This framework has been instrumental in guiding curriculum development and assessment strategies in programming education. In spite of providing a robust model for embedding CT into programming education, its reliance on the Scratch environment may limit its applicability to other programming languages and environments. Additionally, the framework's focus on younger learners may not directly transfer to older or more advanced students.

In addition, Guzdial's work [88] uses this approach to explore methods for making computing education accessible to a broader audience, focussing on integrative approaches that combine computing with other disciplines. The research involved developing educational interventions that were iteratively tested and refined in diverse classroom settings. The findings of this study demonstrated that integrative approaches could significantly enhance student engagement and learning in computing, particularly for students not primarily focused on STEM fields. It also provided a framework for curriculum design that supports a wide range of educational goals and student backgrounds. While promising, the framework's broad applicability might dilute the depth of computing knowledge conveyed, potentially under-

preparing students for specialised roles in computing. The broad scope of the interventions also makes them more challenging to implement consistently across different educational settings.

To conclude, design-based research has been proven valuable in developing innovative educational tools, instruments, and frameworks in computing education literature, particularly in programming learning, where there is a high cognitive load associated with mastering complex concepts. The reviewed studies above highlight some challenges, such as ensuring the transferability of skills to more traditional environments, maintaining the balance between simplification and real-world applicability, and the resource intensity of developing sophisticated adaptive systems. However, they highlight the effectiveness of using this approach in creating educational interventions that are responsive to learner needs and educational contexts. In addition, the reviewed studies show that involving close collaboration between researchers and practitioners, such as educators, programming learners, or instructional designers, ensures that the research is grounded in real-world contexts, addresses practical challenges, and creates, implements, and refines educational interventions in authentic settings. [23] [212]. **Therefore, in this research**, this approach is appropriate to develop an evaluation instrument for online coding tutorial systems in several aspects:

- **Integration of theory and practice in programming education**

  This approach ensures that the instrument for evaluating online coding tutorial systems is rooted in established programming educational theories and best programming practices. By incorporating theoretical foundations, the instrument can effectively address novice programming learning needs and challenges [16] [23]. This helps in creating a more robust instrument that not only evaluates online coding tutorial systems but also enhances programming educational outcomes.

- **Iterative development and refinement of the instrument**

  This approach involves a cyclical process of design, implementation, analysis, and redesign. This iterative nature allows for the continual refinement of designs based on empirical evidence collected from actual use scenarios [23]. In the context of online coding tutorial systems, this means that an evaluation instrument can be continually adjusted to better address the specific challenges and needs of programming novices and educators encountered during the instructional design and delivery process [16]. For example, the initial instrument might be tested and found lacking in supporting certain types of interactive exercises, leading to design revisions that better incorporate these vital elements [52] [23].

- **Collaboration with programming educators and novices**

This approach allows collaboration with novice programming learners themselves [212]. This collaboration ensures that the instrument developed is not only academically robust but also practically viable and relevant to the needs of those who will implement and use it (educators and novices). For example, insights from educators who have extensive experience in teaching programming can be invaluable in identifying the key components that make online tutorial systems effective [212].

- **Addressing several programming learning problems**

  This approach allows to consider multiple dimensions simultaneously, such as technical and content dimensions [47]. Moreover, a model developed through this approach can thus effectively address complex programming educational problems by considering all relevant factors, including technological interfaces, instructional strategies, and learner feedback mechanisms [47] [212].

The five DBR characteristics by [16]	In this current research
1- Real educational context	Two case studies were conducted in a real educational context. Learners and educators involved
2- Involving iterative design process	There were four design cycles in this research. .
3- Involving a collaborative partnership between researchers and practitioners	The system prototype also went through cycles of iterations where users were asked to provide feedback.
4- Design and Testing of an Intervention	Design and evaluate a system prototype
5- Using mixed methods	A systematic literature review, survey, case studies were used.

Table 3.1: How these characteristics are specified in this research. (The DBR characteristics adapted from [16])

In addition, as shown in Table 3.1, the five characteristics of design-based research that were identified by [16] work together to create a research approach that is collaborative, contextually grounded, and focused on producing practical solutions to educational challenges in this thesis.

3.4.1 A cyclic process and phases of design-based research

According to [132], design-based research follows a cyclic process containing cycles of analysis, design, evaluation, and revision. Some authors discussed these phases, for instance,

[171] discussed four main phases. The first phase is concerned with the analysis of practical problems; the second phase involves the development of solutions; and the third phase is about the iterative cycle of implementing and evaluating the interventions. The last phase is about reflection. However, in this thesis, the model of Bikanga [34] and [1] that was based on the original model identified by Mckenney [132] is adopted. They have created a new model that contains only three main phases instead of four, which are: analysis and exploration, design and construction, and evaluation and reflection. Below, these three phases are discussed, and how they are presented in this thesis is explained:

- **Analysis and Exploration Phase** This phase needs context analysis, a review of literature, and the development of a conceptual or theoretical framework for the study [132]. In addition, it is when the researcher uses results from both quantitative and qualitative data analysis to understand a specific phenomenon and validate one set of findings with the other. This is presented in the first three cycles of this thesis (Section 4.2, Section 4.3 and Section 4.4). For instance, in the first study in Section 4.2, the computing education literature was reviewed in order to develop the initial instrument. In Section 4.3, the initial instrument was developed by exploring and analysing learners and educators feedback. Lastly, in Section 4.4, the instrument has been updated based on the findings of the analysis of a list of selected current systems.

- **Design and Construction Phase** This phase is when researchers develop the intervention to evaluate and validate one set of findings with the other. This is presented in the system prototype development phase of this thesis in Section 4.5.

- **Evaluation and Reflection Phase)** This phase is where the summative evaluation is done to conclude whether the solution or intervention meets the predetermined specifications. As this phase often results in recommendations for improvement of the intervention, this phase is called semi-summative. It is when the researcher evaluates the findings with others. This is presented in the system prototype evaluation study in this thesis in Section 4.5.

Throughout these design-based research phases, there is an emphasis on reflection and critical analysis, and researchers and practitioners reflect on the data collected, refine the intervention, and make informed decisions for subsequent iterations [158]. The following Figure 3.1 highlights the design cycles in this study and how the study positions itself within the design-based research phases of GMDR [132]. In addition, the following four versions of the evaluation instrument for OCTSs are shown:

- **The first version** of the evaluation instrument emerges from the literature review.

- **The second version** of the evaluation instrument emerges from the initial (explorative) fact-finding study.
- **The third version** of the evaluation instrument emerges from a competitive analysis study (case study).
- **The fourth version** of the evaluation instrument comes from the development of the system prototype, which corresponds to the design and construction of the GMDR [1] [132]. Also, version four of the evaluation instrument comes from design cycle four, which corresponds to the evaluation and reflection phases of the GMDR. In this design cycle, an evaluation study was conducted to evaluate the system prototype.

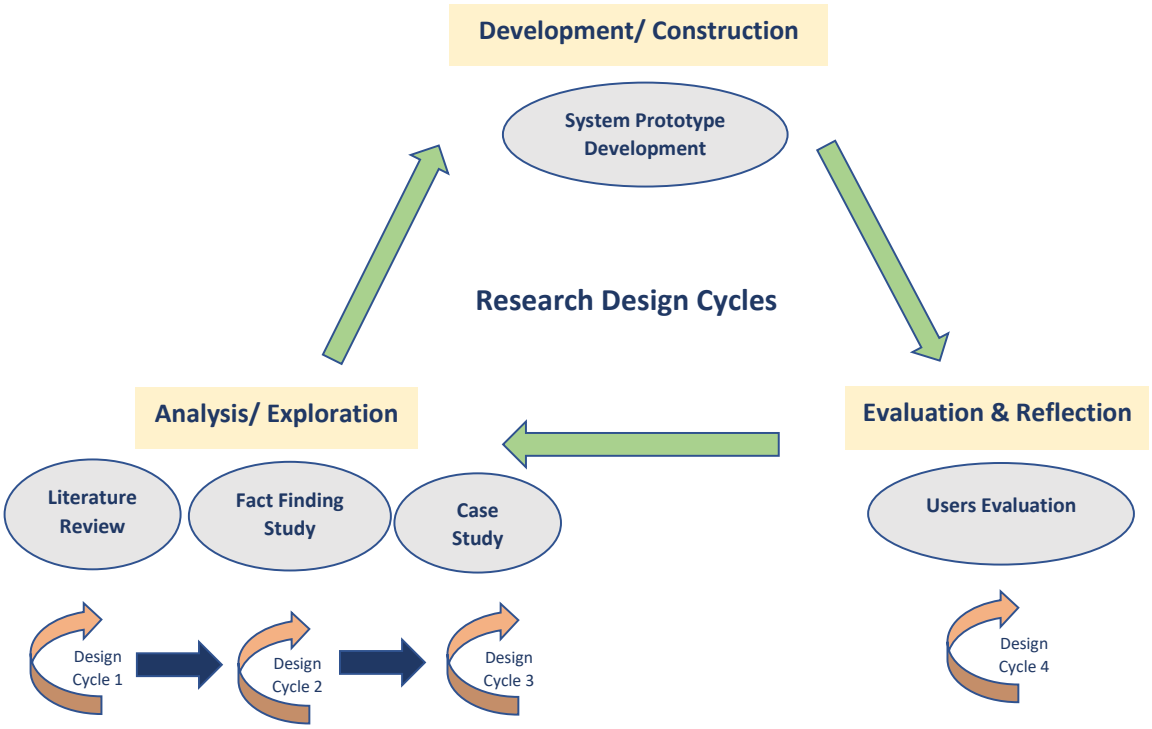

Figure 3.1: This cyclic process is adapted from the DBR model created by [34] and [1], which was based on the original GMDR from Mckenney [132]

3.5 Data Collection Methods for Each Research Question

In this research work on adapting the design-based research methodology, mixed methods were used during the four design cycles of developing the evaluation instrument for online coding tutorial systems. As shown in Figure 3.1, in the analysis, development, and evaluation phases, various methods were used, such as **Systematic review for RQ1, online questionnaire for RQ2, RQ4 and RQ6, case study for RQ3, fuzzy Delphi method for RQ5**, as shown in Table 3.2. In the below sections, each research method for each research question in this thesis is discussed, and the reasons why they have been selected are demonstrated:

Thesis Contributions	Research Questions	Research Methods	Chapters /Sections No
Instrument development	RQ1: What are common programming learning difficulties for novices? And which supportive features are potential solutions for these identified difficulties?	Systematic review	Chapter 4 - Section 4.2
Instrument development	RQ2: What are appropriate supportive features in online coding tutorial systems from learner and educator perspectives?	Online questionnaire	Chapter 4 - Section 4.3
Instrument development	RQ3: What are the supportive features that exist in current deployed online coding tutorial systems and absent in the instrument ? Do the identified supportive features in the evaluation instrument exists in these systems?	Comparative study	Chapter 4 - Section 4.4
Instrument development	RQ4: Building on our research findings, what would an online coding tutorial system look like? Based on a prototype implementation, to what extent are typical learners satisfied with the features of such an online coding tutorial system?	Online questionnaire	Chapter 4 - Section 4.5
Instrument validation	RQ5: To what extent is it applicable to use the proposed instrument for OCTSs as an tool to evaluate any online coding tutorial systems?	Fuzzy Delphi method	Chapter 5
Instrument use in practice	RQ6: What are the attitudes of programming educators toward using the instrument to evaluate online coding tutorial systems?	Online questionnaire	Chapter 6

Table 3.2: Research methods for each research question in this thesis

3.5.1 Systematic review to answer RQ1

In this thesis, the research method used to answer the first research question is: *What are common programming learning difficulties for novices? Which supportive features are potential solutions for these identified difficulties?* is **a systematic literature review process** that was used to explore common programming learning challenges and possible solutions [199] in Chapter 4 in Section 4.2. This method is a structured approach to identifying, evaluating, and synthesising existing research studies on a specific research question or topic [109]. It aims to provide a comprehensive summary of the available evidence and identify any gaps or inconsistencies in the literature [42]. In addition, it follows a predetermined protocol and employs rigorous methods to minimise bias and ensure transparency [109]. This predefined protocol specifies the criteria for including and excluding studies, minimises selection and publication biases, and offers a more objective assessment of the evidence than traditional literature reviews [109]. It can reveal areas where the evidence is conflicting or lacking, thus highlighting the need for further research. This is particularly valuable in computing education, where rapid technological advancements outpace the ability of individual studies to provide conclusive evidence [42]. However, this approach may quickly become outdated, as new studies could significantly alter the understanding of effective educational practices [110]

In the field of computing education, systematic literature reviews serve as a pivotal methodology for synthesising existing research findings, identifying gaps in the knowledge base, and determining prevalent challenges in programming education [42]. For instance, **in terms of identifying programming learning challenges,** Pears et al. [153] used systematic review to analyse research studies related to the teaching of introductory programming. This review aimed to consolidate understanding of pedagogical challenges and effective practices in this area. The methodology involved selecting studies based on predefined criteria, focussing on peer-reviewed articles and conference papers that discussed introductory programming courses. The review highlighted several key challenges, such as high failure rates, difficulties in learning programming concepts, and the significant cognitive load imposed by the syntax and semantics of programming languages. It also identified various effective teaching approaches, including problem-based learning and the use of visual programming environments. However, the study's scope was limited to introductory programming, potentially overlooking challenges experienced by intermediate or advanced learners. Additionally, as the field evolves, some of the findings might need updating to reflect new programming languages and evolving educational technologies.

In addition, Lister et al. [123] employed a systematic literature review coupled with empirical research. This study examined the reading and tracing skills of novice programmers across multiple countries. The review targeted studies that explicitly assessed these skills through

tracing exercises and code comprehension tests. The results revealed widespread difficulties among novices in understanding basic programming constructs and following logical flow in code. It also underscored the importance of teaching code-tracing skills early in the education process to improve overall programming competence. This review was valuable for its international perspective and empirical approach. However, its focus on early programming skills may not capture the full spectrum of educational challenges, particularly those related to more complex programming tasks or advanced concepts.

Moreover, Luxton-Reilly et al. [124] provided a comprehensive systematic literature review focused on the challenges of teaching and learning programming in higher education. The methodology involved detailed protocols for article selection, data extraction, and synthesis, ensuring a thorough examination of current challenges and teaching practices. The review identified consistent themes such as the struggle with abstract programming concepts, difficulty in debugging, and issues with motivation and confidence among learners. It also emphasised the variability in success rates across different educational and cultural contexts, suggesting the need for more personalised approaches to programming education. Although this systematic literature review provides an extensive overview of current issues in programming education, the variability in study designs and contexts across the included articles may impact the generalisability of the findings. The focus on introductory courses also leaves out challenges pertinent to higher-level programming education. Another study by Ahmad et al. [3] also published a review of literature focussing on programming teaching and learning by addressing issues and challenges in the context of introductory programming at the tertiary level. The main objective of this article was to propose the categorisation of programming challenges and highlight the key issues in programming teaching and learning in higher education for further research and improvement.

To conclude, despite this approach' challenges, these systematic literature reviews are instrumental in collating widespread evidence on the challenges faced by learners and educators in the field of programming. Each review brings valuable insights into specific aspects of programming education, but they are often constrained by their focus on certain educational levels or skills. Future reviews could benefit from incorporating a broader range of studies, including those that address recent developments in programming languages, educational technologies, and hybrid teaching modalities, to provide a more comprehensive picture of the current educational landscape in computing [111].

Therefore, in this research, this approach is appropriate to identify programming learning difficulties and possible solutions to develop the initial evaluation instrument for several reasons [103]:

- **Firstly,** in this study, systematic review helps in providing a comprehensive synthesis of existing research in the programming education field. It helps in structuring the

initial base of the instrument by identifying programming learning challenges and possible solutions. In other words, it helps gather data from a wide array of programming education studies to offer a complete picture of the current knowledge base in the field. This is crucial in fields like programming education, where diverse factors, including cognitive, pedagogical, and technological, impact learning outcomes [111].

- **Secondary,** by using this approach, the specific programming learning challenges that novice learners face can be captured. It addresses novice learners' challenges by synthesising research on learners' difficulties with programming and by using specific key-terms [103].
- **Thirdly,** using this approach helps in identifying what works and what does not, which allows for a more strategic design of online coding tutorial systems, integrating pedagogical tools and approaches that have been empirically validated in the programming education field [103].

3.5.2 Online questionnaire to answer RQ2, RQ4 and RQ6

The research method used to answer both the second research question and the fourth research question is:

RQ2: What are appropriate supportive features in online coding tutorial systems from learner and educator perspectives?,

RQ4: To what extent are typical learners satisfied with the features of such an online coding tutorial system?,

RQ6: What are the attitudes of programming educators toward using the instrument to evaluate online coding tutorial systems? is **an online questionnaire**.

For RQ2 in Chapter 4 Section 4.3, it was used to gather quantitative and qualitative data from a sample of individuals (programming novice learners and educators) through structured questionnaires to improve the first version of the instrument. Moreover, for RQ4 in Chapter 4 Section 4.5, it was conducted to test the online coding tutorial system prototype (PythonOCTS [165]) and to collect quantitative and qualitative data from a sample of individual users. In addition, in this study [10], the fourth version of the evaluation instrument is introduced. Both of those studies aim to evaluate educational technologies used by real users to capture their thoughts and measure their satisfaction.

In addition, for RQ6 in Chapter 6, it was used to gather quantitative and qualitative data from the target audience of the instrument (programming educators) through structured questionnaires to investigate their attitudes toward the use of the instrument to evaluate online coding tutorial systems.

The online survey method has become a prevalent tool in several disciplines, offering a convenient and efficient means for data collection across diverse populations [37]. This method employs digital questionnaires distributed through the internet, allowing researchers to gather data on a wide range of topics from large groups of people [37]. It has been used to collect both qualitative and quantitative data [208]. The data collection and processing are significantly faster as responses are gathered and analysed digitally [53]. They can reach a geographically dispersed audience more easily than traditional methods [220]. In addition, respondents can complete surveys at their convenience, which can potentially lead to higher response rates [63].

However, using this approach has some challenges; for example, the proliferation of online surveys can lead to survey fatigue, resulting in lower response rates compared to other methods [68]. Moreover, the absence of the researchers in the process of filling out the online survey might lead to misinterpretation of questions or disengaged responses [80].

In the field of computing education, the use of online surveys within a mixed-methods instrument can provide nuanced insights into the programming of educational technologies, programming pedagogical approaches, and programming learner behaviours [63]. By combining quantitative and qualitative data, researchers can gain a comprehensive understanding of the multifaceted issues within this field. Studies using mixed methods, including online surveys, allow for a more holistic view. For instance, Venkatesh, Brown, and Bala [56] highlight the ability to triangulate data, enhancing the validity of the findings by correlating quantitative statistical results with qualitative insights. In computing education, where the impact of technologies often varies significantly by context, this can elucidate how specific tools support or hinder learning. Online surveys facilitate large-scale data collection across diverse geographical locations and demographics, which is essential in computing education, which often involves diverse international audiences such as programming learners. A study by Dillman et al. [63] exemplifies this by efficiently collecting data from a broad spectrum of participants, highlighting differing educational needs and responses to the programming education technology.

However, one major critique is that the anonymity of online surveys can lead to issues with response quality. According to Gosling et al. [80], participants might provide superficial answers or exhibit social desirability bias in their responses. In computing education, where technical subjects are discussed, this might lead to oversimplified answers that lack depth or critical technical insights. For example, [34] conducted evaluation studies and used an online questionnaire in the process of developing a mobile application for assessment feedback to enhance student motivation, engagement in tertiary education. Moreover, a study was conducted to investigate novice learners' and educators' attitudes towards mobile learning in higher education using an online survey [6]. This study shows the effectiveness of using this approach to collect both quantitative and qualitative data [6].

On the other hand, online surveys, especially when unsolicited, suffer from low response rates, as noted by Sheehan [189]. In the context of computing education, where educators and students are often bombarded with digital communications, engagement with surveys can be particularly challenging. This may lead to response biases, where only certain types of individuals—perhaps those with strong opinions or more time—choose to respond [189].

In this research, for both RQ2 and RQ4, this method is appropriate to be used to improve the development process of the evaluation instrument because it helps in gathering quantitative and qualitative data on a large scale, efficiently capturing the perspectives and preferences of a diverse range of stakeholders, such as programming learners and programming educators [63]. By using this research approach, quantitative data can be collected from structured questionnaire responses, and qualitative data from open-ended responses that provide in-depth insights into appropriate supportive features in an online coding tutorial system from real users (learners and educators) [225].

3.5.3 Comparative study to answer RQ3

The research method used to answer the third research question: *What are the supportive features that exist in current deployed online coding tutorial systems and are absent in the instrument? Do the identified supportive features in the instrument exist in these systems?* is **a comparative study** that was conducted to do a comparative analysis across the features in the second version of the instrument and seven selected current online coding tutorial systems. This study has been published and presented in this paper [11].

Comparative studies are typically used across various disciplines to explore the similarities and differences between groups, conditions, or time periods [83]. This approach can be particularly powerful for identifying the impacts of different interventions or understanding variations across different contexts [83].

In the field of computing education, comparative studies are instrumental in understanding different educational interventions, tools, and pedagogical approaches. By directly comparing two or more methods or groups, these studies aim to identify which factors most effectively promote learning in computer science education. They often provide clear evidence about the effectiveness of different teaching methods or tools by measuring their impact on student learning outcomes directly against each other. For instance, Al-Zoubi et al. [91] demonstrated the effectiveness of using gamification compared to traditional teaching methods in enhancing students' programming skills. Moreover, when properly designed, comparative studies that include diverse educational settings, tools, and demographics can offer findings with high external validity, allowing for broader generalizations. For example, Beaubouef and Mason [27] explored gender differences in computer programming courses across different institutions.

One significant limitation of using this approach in computing education is that ensuring that learning experiences are equivalent across different study conditions can be challenging. Differences in how instructors interpret and implement interventions may lead to inconsistencies. This issue was highlighted by Lahtinen et al. [118], who noted variability in teaching approaches when comparing pedagogical techniques in computer science education.

In this research, this approach has been found appropriate to compare online coding tutorial systems in order to improve the development of the evaluation instrument for online coding tutorial systems and to identify new features because this research method involves in-depth examination of a particular case [71]. In the context of online coding tutorial systems, this approach helps examine current systems to identify new features. It allows to systematically examine multiple existing systems, draw out key insights, and identify best practices and areas for improvement [71]. In addition, it helps to identify which features are common across various successful online coding tutorial systems, as well as those that are unique to specific systems. This can shed light on essential elements that should be included in any online coding tutorial and those that can be tailored to specific educational goals or target audiences.

3.5.4 Fuzzy Delphi method to answer RQ5

The research method used to answer the fifth research question: *To what extent is it applicable to use the instrument for OCTSs to evaluate any online coding tutorial systems?* that is answered in Chapter 5 is: **fuzzy delphi method**. The main aim of this study is to gather qualitative data in order to validate the proposed instrument that has been developed in Chapter 4 to evaluate online coding tutorial systems by experts. The Fuzzy Delphi Method (FDM) is a refined version of the traditional Delphi method, incorporating principles of fuzzy logic to handle uncertainties and subjective judgements in the decision-making process [99].

This study followed a number of the steps in the Fuzzy Delphi Method, as shown in Figure 3.2. First, gather a team of subject-matter experts, convert their opinions into imprecise numerical values, and draft a questionnaire outlining the primary issues or elements that need to be assessed. This usually means using verbal variables, like ”very important,” ”important,” and ”less important,” that are converted into imprecise numerical scales. Then group the vague opinions from all the experts together. This can be done by merging fuzzy numbers and applying a range of mathematical approaches to arrive at a consensus value. Finally, convert the combined fuzzy findings back into a crisp number to make them understandable and practical [99].

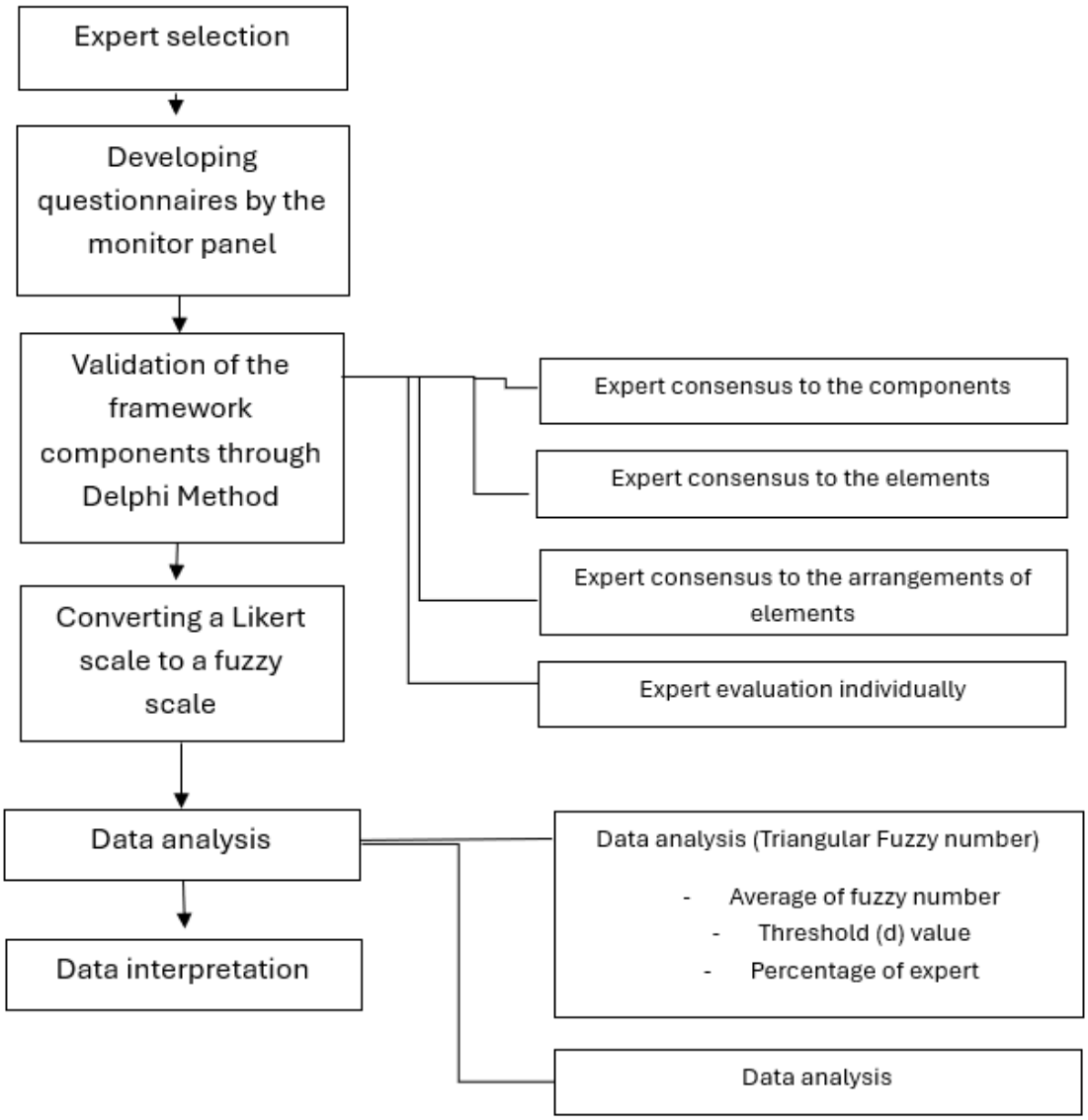

Figure 3.2: Validation procedures for the components and the items of the instrument for evaluating online coding tutorial systems

In computing education, fuzzy Delphi method was used to obtain expert consensus on the topics and skills that ought to be covered in the most relevant computing curricula. For instance, as technology advances, there is a continuing need to update educational materials. Experts can provide guidance on emerging trends and essential skills, and fuzzy Delphi method helps integrate these recommendations to produce a coherent curriculum. Chang, Hsu, and Chang [221] used fuzzy Delphi method to identify important competences for information technology workers in order to keep training programs in line with market demands. Moreover, the dynamic character of the field poses challenges in developing assessment techniques that precisely gauge students' proficiency in computer-related learning. Fuzzy

Delphi method can help with the development and validation of assessment instruments by including professional opinions on different assessment processes and their effectiveness. In a study by [98], evaluation criteria for e-learning systems were created using fuzzy Delphi method, ensuring comprehensive and relevant assessments.

In this research, this approach has been found appropriate to validate the instrument through a consultation process to get the final outcome must be interpreted as a statement of expert consensus. Consequently, the fuzzy Delphi technique is considered a reliable and effective tool [99]. The Delphi method is a widely acknowledged and employed methodology with various applications in several disciplines, including education [46]. Furthermore, employing the fuzzy Delphi approach can help prevent misrepresentation or the loss of crucial data that could happen when utilising the Delphi approach [130]. Fuzzy Delphi approach can provide precise and timely input, but it has several drawbacks as well. In addition to interacting with eager research subjects, researchers need to have background information relevant to the study's context, which can be found in the literature review. Other research that calls for consensus and expert assessment can also use the fuzzy Delphi method. The study and assessment phase resulted in the development of models, modules, frameworks, and products. In particular, the fuzzy Delphi method has recently been adopted in the context of computing education as a valuable tool to develop a computing curriculum [120], to integrate technology into teacher education [97], or to validate elements of computational thinking for solving problems in programming [227], or to create learning-orientated rubrics for Computer Science Principles teachers using the Beauty and Joy of Computing curriculum [169]. Furthermore, the fuzzy Delphi approach can prevent misrepresentation or the loss of crucial data that can happen when applying the Delphi approach [130]. Even though fuzzy Delphi method can provide prompt and accurate feedback, there are certain drawbacks. In addition to interacting with study specialists who are eager to take part in the research, researchers also need to possess prior knowledge relevant to the study's environment, which can be found in the literature review.

3.6 Research Timeline

The four design cycles in this research work were conducted over three years, from 2020 to 2023. As shown in Figure 3.3, this research has gone through several phases and iterations during the last three years to propose the evaluation instrument for online coding tutorial systems, which is the main objective of this work.

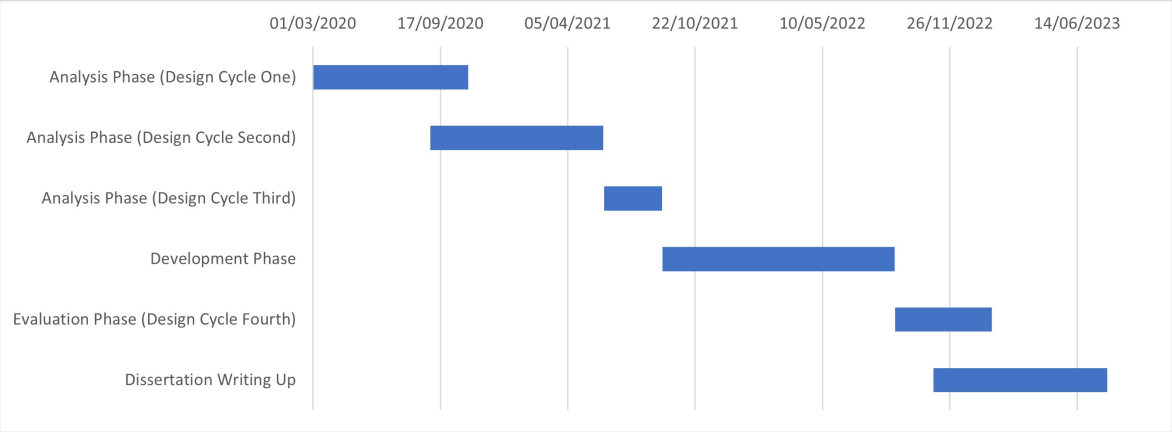

Figure 3.3: Research timeline

3.7 Chapter Summary

This chapter proposes the research design and methodologies used in this research work. Each research question and each study have been clarified, and the method that has been used for each study has been discussed.

Chapter 4

Instrument Development

4.1 Chapter Overview

This chapter describes the process of creating the evaluation instrument for online coding tutorial systems: iteration. It comprises four design cycles, the purpose of which is to modify the instrument in accordance with empirical data, the opinion of experts, and rational analysis. The chapter is structured as follows: Section 4.2 describes the first design cycle which covered the development of the instrument as described in this paper; Section 4.3 deals with the second design cycle in which the input of educators and learners was incorporated; Section 4.4 outlines the third design cycle, which is the analysis of current online coding systems; Section 4.5 describes the last cycle of the design process, which entails the assessment and the finalisation of the instrument and concluding with a brief chapter summary in Section 4.6.

4.2 Instrument Design Cycle One

This section presents the first design cycle in the analysis/exploration phase, which comprises a literature review study. Moreover, it proposes the first version of the instrument for evaluating online coding tutorial systems. The **first design cycle** in this research aims to identify common programming learning challenges by conducting a systematic literature review to propose the main components for the instrument and a set of features as solutions to develop the first version of the instrument's items (**draft one**).

4.2.1 Research question 1

In this study to develop an initial instrument for evaluating online coding tutorial systems, a systematic literature review and semi-systematic review were conducted to address RQ1: **What are common programming learning difficulties for novices? Which supportive features are potential solutions for these identified difficulties? This study contributes to** the knowledge by proposing an initial instrument for evaluating online coding tutorial systems. In addition, it contributes to knowledge by focusing on understanding the common difficulties related to programming learning, particularly for novice programmers (i.e. those with little or no prior coding experience). From personal observation, knowledge about programming learning difficulties is scattered across the literature, and there is little exploration of possible solutions for these problems. We observed that the majority of relevant research predominantly relied on quantitative, questionnaire-based methodology. Although such work has uncovered novice learner difficulties, an in-depth understanding of the supportive system features that can overcome these challenges is limited.

4.2.2 Study method

In this research work, the novice problems in learning programming and possible solutions need to be investigated in order to develop a instrument for evaluating online coding tutorial systems based on novice learners' and educators' needs. Therefore, firstly, this section presents the programming learning difficulties that are identified from programming education literature.

4.2.2.1 Novices' problems

A systematic literature review of programming education has been performed to identify programming learning difficulties for novices. This systematic review process was done to gather data [111, 199] to explore common programming learning challenges. An initial search of articles published between 1980 and 2023 was carried out. Searches were done in the ACM and IEEE.

- **Search databases:** The databases searched are mainly from the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE). The reason for focusing only on these two databases is that most research related to computing education can be found in the ACM and IEEE databases, where most of the well-known computing education conference proceedings and journals can be found. For instance, Koli Calling, ICER, UKICER, and SIGCSE.

- **Search terms:** A narrow search was done in order to select and review the papers that only focused on identifying programming learning problems. In addition, the strategy that has been used is to search the papers only with the titles of programming, teaching, or learning problems. Therefore, the keywords used were boolean combinators, as follows:: **"Programming" OR "Coding" OR "Computer Programming" AND "Learning" AND "Difficulties" OR "Issues" OR "Challenges" OR "Problems"**.

- **Search process:** The search process was done by selecting the "Title" option in both databases in the advanced search. Therefore, the number of returned papers seems to be small.

- **Publication date:** The initial search was for articles published between 1980 and 2023. This selected period of time has been chosen in order to include most of the published studies in the programming education field.

Source	# Studies Found	# Candidates	# Selected
IEEE	36	15	5
ACM	16	10	2
Total	52	25	7

Table 4.1: Search summary for each database

- **Inclusion and exclusion criteria:**As shown in Table 4.1, 52 relevant articles were found. Then, an initial inclusion screening was done based on title and abstract to get a subset of candidate studies that only focus on programming learning challenges, and the article numbers were filtered to 25 after removing duplicates and out-of-focus papers. From these 25 remaining articles, further screening was performed by considering full-text content, excluding articles that did not discuss programming learning difficulties, and removing duplicate and non-English articles. The final number of selected articles was 7.

Reference	Paper Title
[138]	Difficulties in learning and teaching programming—views of students and tutors
[35]	Why is programming so difficult to learn? Patterns of Difficulties Related to Programming Learning Mid-Stage
[167]	Using data to under- stand difficulties of learning to program: A study with Chinese middle school students
[155]	Learning computer program- ming: study of difficulties in learning programming
[78]	A teacher's view about introductory programming teaching and learning: Difficulties, strategies and motivations
[92]	Difficulties in Learning Structured Programming: A Case Study in UTP
[203]	Learning difficulties in programming courses: undergraduates' perspective and perception

Table 4.2: List of seven selected papers

From these 25 remaining articles, further screening was performed by considering full-text content, excluding articles that did not discuss programming learning difficulties, removing duplicate articles and non-English articles. The final number of the selected articles was 7 (see Table 4.3).

Programming Learning Difficulties	[138]	[35]	[167]	[155]	[78]	[92]	[203]
Syntax of programming languages		✓	✓	✓	✓	✓	✓
Structure of code	✓	✓	✓	✓		✓	✓
Understanding basic concepts	✓		✓	✓		✓	✓
Debugging		✓		✓		✓	✓
Dividing functionality into procedures				✓		✓	✓
Transferring algorithm to concrete implementation					✓		✓

Table 4.3: List of programming learning difficulties identified in the literature

4.2.3 Systematic literature review findings

The finding of this systematic review is shown in Table 4.3 that presents a list of programming learning difficulties identified from the programming education literature. The rows are

ordered from the highest number of papers mentioning a problem to the lowest number, with the requirement that at least two papers must corroborate a difficulty before we include it in our list. Three problems were observed (syntax, structure and basic concept understanding) that are inherent in reading and understanding code. The remaining three problems (debugging, proceduralization and algorithm implementation) involve writing and executing code activities.

In this section, each programming learning problem identified from the literature is defined and discussed.

- **Syntax of programming languages**: Syntax refers to the specific rules and structure that define how programming languages are written [148] . Novices may struggle with understanding and applying these syntax rules correctly. For example, they might have difficulty using the proper syntax for declaring variables, defining functions, or writing conditional statements. Syntax errors can lead to programs that fail to compile or run correctly. To overcome this challenge, novices need to familiarize themselves with the syntax of the programming language they are learning and practice writing code to reinforce their understanding [196] [35][167] [155] [78][92] [203].

- **Structure of code:** The structure of code refers to how different parts of a program are organized and interconnected. Novices may find it challenging to design code with a clear and logical structure [216]. Novice learners may struggle with concepts like breaking down a problem into smaller components, deciding when to use functions or classes, or understanding how different sections of code interact. Developing a good understanding of program organization and modular design principles can help novices create code that is easier to read, maintain, and debug [138][35][167][155][92][203].

- **Understanding basic concepts:** Novices often encounter difficulties in grasping the fundamental concepts of programming. These concepts include variables, loops, conditionals, and data types [190] . For example, understanding how variables store and manipulate data, or how loops allow for repeated execution of code, can be challenging for beginners. It is crucial for novices to invest time in studying and practicing these basic concepts to build a solid foundation in programming [138][167][155][92][203].

- **Debugging:** Debugging is the process of identifying and fixing errors or bugs in a program [13]. Novices may struggle with finding and resolving errors due to their limited experience. Debugging requires skills such as reading and understanding error messages, analyzing code logic, and systematically testing different parts of the program. Novices can improve their debugging skills by using debugging tools provided by integrated development environments (IDEs), studying error messages, and employing systematic approaches to locate and fix issues in their code [35] [155] [92] [203] .

- **Dividing functionality into procedures:** Novices may find it challenging to break down a complex problem into smaller tasks or procedures. This skill, known as procedural decomposition, involves identifying the different functionalities required to solve a problem and dividing them into separate, manageable procedures or functions. Novices may struggle with determining the appropriate level of granularity for these procedures or understanding how to pass data between them. Developing a problem-solving mindset and practicing algorithmic thinking can help novices improve their ability to divide functionality effectively.

- **Transferring algorithm to concrete implementation:** Translating an algorithm or problem-solving approach into actual code can be daunting for novices. They may struggle with understanding how to represent their ideas and logical steps using the syntax and constructs of a programming language. This challenge requires both an understanding of the problem-solving process and a solid grasp of programming concepts. Novices can improve this skill by breaking down problems into smaller steps, pseudocode or diagramming their algorithmic approach, and gradually translating those steps into code [118].

4.2.3.1 Potential solutions

After identifying some of programming learning problems from programming education literature in previous section, a further, semi-systematic review was conducted in order to investigate possible solutions for these problems. These possible solutions are supportive features that might be worthwhile to have in online coding tutorial systems to help students to learn coding more effectively. Therefore, this section presents the identified supportive features for the identified difficulties in previous section to form an initial instrument for evaluating online coding tutorial systems which is the main purpose of this research work. The search strategy used in this study to discover possible solutions for the set of common programming learning difficulties identified in the previous Section was a semi-systematic review [199]. Usually the aim of a semi-systematic review approach is to review every single article that could be relevant to the research topic. Therefore, we use this approach to cover different types of studies from programming education literature to identify support features that are helpful to overcome programming novice learners' difficulties.

The research strategy employed is as follows: First, we looked at the possible solutions described in the selected articles in the previous section. Some supportive features were identified to assist novices based on these articles. In addition, the specific difficulties were used as keywords for further research. Searches were done in the ACM and IEEE databases.

4.2.4 Semi-systematic literature review findings

This section presents the initial list of supportive features that were identified from the programming education literature. This initial list of features form the initial instrument for evaluating online coding tutorial systems. In this section, the list of features were grouped according to the problems that were identified in the above section. Note this section only discusses the supportive features that could be incorporated in a software-based programming environment since this research focus on online coding tutorial systems.

Syntax of programming languages

According to [115], syntax difficulties are the overhead of learning the syntax and semantics of a language at the same time. In addition, [177] mentioned that understanding syntax problem is simply the challenges that are faced not only by novices but also by learners who have adequate problem-solving skills and manage to phrase a solution to a programming problem in terms of informal code, but find it difficult to turn such code into a syntactically correct computer program. Moreover, according to [177], syntax understanding is not the main difficulty. Novice learners may know the syntax and semantics of individual statements, but they do not know how to combine those elements in order to produce valid programs [200]. However, three helpful supportive features have been discussed in previous studies, these features are; syntax error messages, underlining syntax errors, and syntax source code highlighting or coloring.

- **Syntax error messages**: Reporting syntax errors in a programming environment helps novice learners reduce mistakes in spelling, punctuation and order of keywords in their programs [59, 96]. For instance, SyntaxTrain parses a student's source code and, if it detects a syntax error, displays an error message and a diagram illustrating the required syntax [143]. Moreover, [96] mentions that syntax error messages do not necessarily point the learners in the right direction needed to fix the code but providing this technique in programming environments help learners to understand syntax and semantic errors in their code. Providing comprehensible syntax error messages is often motivated by the need to better serve novice programmers [182].

- **Underlining syntax errors:** Source code errors in modern integrated development environments are highlighted interactively with red underlines below problematic lines of code [29]. This is often accompanied by hints or error message pop-ups.

- **Syntax highlighting:** Highlighting helps novice learners to identify keywords and become familiar with language-specific concrete syntax. Researchers have examined the influence of syntax highlighting on novice comprehension of source code [90, 166]. For instance, [90] conducted a controlled experiment with 390 undergraduate students

in an introductory Java programming course, and the authors examined how syntax highlighting improves novices' ability to comprehend source code by measured the correctness with which they solved small coding tasks.

Structure of code Novice learner earners may struggle to understand how to build blocks of the code and syntax constructs and commands that perform actions [135]. However, a supportive feature has been identified, this feature is the visual map.

- **Visual map:** Natural visual learners are recommended to begin by learning a graphical programming language, which provides a visual map as a bridge to learning textual programming languages [197]. In general, program visualization systems are developed to help beginners understand fundamental programming concepts, structure and execution [201].

Understanding basic concepts

According to [155] [92], most novice programmers have problems with understanding basic concepts, which are encountered at the beginning of their learning journey—for instance, variables, arrays and loops. However, four supportive features have been identified as helpful to understand basic concepts of programming languages. These features are lesson content, reference materials, worked solutions and quizzes.

- **Lesson content:** Programming learning environments can provide contextually relevant, structured lesson content that teaches different concepts [55].
- **Reference materials:** These authoritative resources can promote understanding of programming concepts since learners can browse and request these materials at any time. Such reference information might be organised as a digital textbook [55].
- **Worked solutions:** The availability of complete example programs and worked solutions was used in some programming learning environments to demonstrate programming concepts [55].
- **Quizzes:** Basic assessment activities and quizzes allow learners to test their understanding. In addition, providing questions along with interactive exercises can help to reinforce concepts and measure novice learner achievement [55].

Debugging

Novice programmers need to know how to test and analyze their code to identify and correct problems, and this is called debugging [70]. In addition, novice learners face some difficulties in understanding the problem domain, finding the bugs and errors and resolving bugs

[135] [203] [70]. To help novice learners overcome this difficulty, three supportive features have been identified. These features are detailed error messages, identifying errors locations and customized hints.

- **Detailed error messages:** Raw error messages are often uninformative and sometimes misleading for novices [166]. Researchers have created tools to provide enhanced error messages. For instance, CS1 students who saw detailed error messages made significantly fewer errors than those who only saw raw Java error messages; more detailed error messages helped students debug their programs [28].
- **Identifying error locations:** Novice programmers get frustrated by errors, and they try to debug their code in the hope that they discover the location of the error to make debugging easier [40, 182]. Therefore, novices persistently seek assistance for problems with basic syntactic details; identifying errors locations can help novice learners debug their code [40].
- **Customized hints:** Providing specific hints based on novice learner errors is a beneficial addition to any teaching programming platform [18][129]. When a novice types a segment of code and the editor shows an error, the interactive hints feature should provide some guidance to help the developer find and fix the bug [18, 166].

Dividing functionality into procedures

- **Auto-completion:** The code editor should predict whatever the programmer wants to type. In addition, auto-completion is considered a helpful feature for supporting users in making procedure calls and developing additional procedures that can be used exactly the same way as the built-in library functions [223]. Moreover, this technique saves programmer time by helping them to complete keywords rather than typing every character.

Transferring algorithm to concrete implementation

- **Syntax-directed editor:** To help novice programmers to use a programming language to implement an algorithm for solving a specific problem without concern for syntactic detail, some programming learning tools support templates and menus with syntactically correct choices for every incomplete part of a program [223]. Such templates can be based on textual representations or graphical representations.

4.2.5 First version of the instrument

The list of identified supportive features that was discussed in the previous section is synthesized to create the initial instrument for evaluating online coding tutorial systems. The aim of developing and validating this instrument to provide educators with systems that might help them in teaching and learning programming in more effective ways since this instrument is developed based on a systematic approach. In this chapter, the first design cycle of the evaluation instrument for OCTSs is presented. The main aim of this design cycle is to identify answers from the available literature and to develop an initial solution to the problems, and this initial instrument focuses on the identification of the supportive features to help novice learners to overcome programming learning difficulties.

Components	Items
Syntax of programming languages	Syntax error messages
	Underlining syntax errors
	Syntax highlighting
Structure of code	Visual map
Understanding basic concepts	Lesson content
	Reference materials
	Worked solutions
	Quizzes
Debugging	Detailed error messages
	Identifying error locations
	Customized hints
Dividing functionality into procedures	Auto-completion
Transferring algorithm to concrete implementation	Syntax-directed editor

Figure 4.1: The initial evaluation instrument for online coding tutorial systems based on the systematic literature review (design cycle one)

As shown in Figure 4.1, based on the identified programming learning challenges found in the literature [167] [155] [78] [92], this first draft for the evaluation instrument for online coding tutorial systems has shown a number of pragmatic suggestions from the literature for learner-assistive features that can be provided by coding environments.

4.3 Instrument Design Cycle Two

The online coding tutorial systems offer learners a flexible and accessible platform to acquire coding skills, while educators benefit from the ability to monitor progress, provide personalized feedback, and facilitate effective instruction at scale. However, to develop truly effective and engaging online coding tutorial systems, it is crucial to understand the dual perspectives of both learners and educators. In the previous Section (Section4.2), a comprehensive exploration of existing literature on programming education was conducted. The findings revealed valuable insights on a list of supportive features that had were used to synthesize the first version of the evaluation instrument of OCTSs in Section 4.1.

However, this section presents the second version of the instrument which was developed in **the second design cycle** in the analysis/exploration phase [1]. The objective of this second design cycle is to improve the initial instrument based on the perspectives of both learners and educators. Therefore, in this study, an online survey was distributed among learners and educators to investigate their needs and to collect their feedback and suggestions on a current online coding tutorial system called LearnPython [175].

Conducting this fact-finding study is significant because involving learners and educators in developing the evaluation instrument for online coding tutorial systems will fill the gap of lacking of educational needs in the instrument. This goal was achieved by investigating whether the features in the first version of the evaluation instrument of online coding tutorial systems that proposed in Section 4.2.5 found helpful by learners and educators. Moreover, learners and educators were suggesting some new features that were added to the proposed instrument. Evantually, this chapter proposes the second version of the evaluation instrument for Online Coding Tutorial Systems (DFOCT) in Section 4.3.6 and presents **design cycle two** stage of the research study to explore educators' and learners'views on the features and characteristics of a selected online coding tutorial system called (LearnPython [175]).

4.3.1 Research question 2

In this study to improve the first version of the evaluation instrument a second design cycle was done. In this second design cycle, an online survey instrument was designed in Appendix

A and distributed to address RQ2: **What are the appropriate supportive features in online coding tutorial systems from learner and educator perspectives?**

This study contributes to the knowledge area by improving the development of the evaluation instrument for online coding tutorial systems that is the main purpose of this research work. This development of the instrument was done through identifying more supportive features for online coding tutorial systems from learners and educators by using an online survey instrument. In addition, this online survey aims to measure the satisfaction of the learners and educators toward the initial instrument and also update the first draft of the evaluation instrument and produce the second version.

4.3.2 Study method

Th data collection method used in this study to collect the programming educators and learners' feedback on the proposed features and characteristics of the first version of evaluation instrument for online coding tutorials systems and update the instrument in Section 4.2.5 was an online survey as presented in Appendix A. This online survey was distributed among participants for 2 months from March 2021 to May 2021 after receiving the ethical approval from the College of Science and Engineering and the number of the application is 300200206.

4.3.2.1 Procedure

This online questionnaire was distributed randomly to fellow educators and learners participants are asked to fill in a questionnaire on their opinions with the list of system features in the selected online coding tutorial system called LearnPython [175]. This online survey was distributed among programming learners and educators by sending an invitation email to the School of Computing Science mailing list in University of Glasgow. In addition, this online survey was designed based on the initial instrument elements discussed in Section 4.2, and it contains four sections as presented in Appendix A, and below is a list of sections of the online questionnaire.

- The first section is a consent form.
- The second section contains pre-testing questions (demographics questions).
- The third section contains a set of testing instructions.
- The last section contains post-testing questions, i.e. a list of statements about the features of online coding tutorial system. This list is designed totally based on the features in the first version of the evaluation instrument (Section 4.1. These questions gave a

scale of scores between Strongly Disagree and Strongly Agree while some statements ask whether the learners and educators are satisfied with the proposed features in the initial version of the evaluation instrument of OCTSs proposed in Section 4.2.5.this section contains three parts; **Part 1** contains questions about features and characteristics that tested system has. **Part 2** contains questions about features and characteristics that tested system does not has, and **Part 3** contains two open ended questions to collect suggestions about other important features of online coding tutorials systems from participants.

4.3.2.2 Pilot study

Before distributing this online survey, a pilot study was conducted in order to test the clarity and the validity of the content of the survey. The pilot study was performed between January and February 2021. Three PhD researchers from Computing Science School in University of Glasgow participated in this study, and the small number was because, most of the PhD researchers in the school were busy on their studies. Nevertheless, the information gathered was enough to enhance the clarity of the questions of the online survey. Some changes had been made to update the online survey questions based on the participants' feedback collected in this pilot study.

4.3.2.3 Participants

The target participants in this study were novice, intermediate, and expert programmers (learners and educators). By distributing the online survey through various platforms such as WhatsApp and email, the main aim was to attract participants with diverse programming backgrounds. Initially, the target sample size was set at around 200 participants to gather a wide range of feedback and suggestions. However, the actual sample size for this study turned out to be only 37 participants. The smaller than expected number of participants could be attributed to several factors. Firstly, it is possible that some of the participants who received the survey were not interested in the subject matter or did not find it relevant to their current activities or research focus. As a result, they might have chosen not to participate in the survey, leading to a lower response rate.

Additionally, the timing of the survey distribution could have influenced the participation rate. If the survey was sent during a period when potential participants were preoccupied with other academic or personal commitments, it might have impacted their willingness or ability to complete the questionnaire. Although the sample size of 37 participants is relatively small, it is important to note that the study still managed to collect valuable insights and feedback from learners and educators.

4.3.2.4 Demographic

After distributing the online survey, the data collected from the second section, as shown in Appendix A, covers participant demographic information. This characterizes the programming learners and educators that were included in the data collection process. The demographic data revealed that participants were from Saudi Arabia and the UK, that the total sample was n=37, out of this sample the learner sample was n = 24 and the educator sample was n = 13. Therefore, in this study, the feedback reflects mostly the feedback given by learners. Moreover, participants ranged in age from 18 to 60, and the coding experience data showed that 37% (n = 14) have been coding for three years or more, 31% (n = 11) participants have been coding for two to three years, and also 32% (n = 12) participants have no long-term experience of coding (less than one year of coding).

4.3.3 Data analysis techniques

In this study, to improve the first version of the design instrument for OCTSs quantitative and qualitative data were collected through the online survey. As presented in Appendix A, each participant went through three scenarios and subsequently, they completed the questionnaire items related to both categories of features identified in Section 4.2. The responses to these Likert scale questions were analyzed using **a descriptive statistical technique** [69]. For the quantitative data analysis in this part of the study, Excel was utilized as the chosen analysis tool [145]. The results of these quantitative questions were presented in the form of histogram graphs, which depicted the frequency distribution of each response option.

For the qualitative data, a **thematic analysis** of participant responses to the two open-ended survey questions [205]. Several phases had been followed as described by [36]; gaining familiarity with data, generating initial codes or labels, searching for themes or main ideas, reviewing themes or main ideas, defining and naming themes or main ideas, and producing the report. For this study, the coding was implemented by hand. The codes chosen aimed to identify the elements that participants noted as important to them in their responses. Out of 37 participants, only 27 answered the first open-ended question, '***What other helpful and usable features or characteristics of online coding tutorial systems would you suggest?*** and 10 participants responded to the second open-ended question ***Any other suggestions?***

4.3.4 Study finding

4.3.4.1 Quantitative findings

After each participant went through three scenarios described in Appendix A, each participants had filled the set of items in the questionnaire related to the features that was identified

in Section 4.2. In this section, data analysis is processed using quantitative data analysis techniques. In the implementation of this research using the type or form of descriptive research that is carried out through data collection in the field. The data analysis technique used in this part of the study for quantitative data analysis is Excel. Below is a list of the quantitative findings for each features discussed in the online survey:

As presented in Appendix A, the online survey contains a list of questions about each feature identify in the design cycle one 4.1. In this section, the responses of the survey participants are presented:

- **List of coding lessons** The responses to this feature show that 83.7% of the participants felt that offering coding lessons and tutorials in online coding tutorials systems could enhance the understanding of the coding topics. It is believed that starting with basic concepts of programming language in learning coding would be easier for learners to understand the concepts particularly for learners which are classified as visualize learners.

- **Reference materials** The responses to this feature indicate 17.2% of learners and educators disagree on offering other coding learning material rather than coding tutorials, they found this feature unhelpful feature in their learning journey. However, most of the participants (71.4%) of learners and educators agree on offering other coding learning material such as providing other books and finding extra learning materials might enhance of learning programming.

- **Worked solutions** The responses to this feature show 85.7% majority of the learners and educators felt that providing solutions for quizzes questions allows the learner to evaluate his/her skills at the end of each section is an importance feature of online coding tutorials systems.

- **Quizzes** The responses to this feature show 86.1% majority of the learners and educators felt that the assessment activity such as quizzes that allows the learner to evaluate his/her skills at the end of each section is an importance feature of online coding tutorials systems. This is because, the aim of the assessment activity is to evaluate learners coding skills at the end of each section, and this eventually enhance the thinking and problem solving skill because they need to quickly think for the answer and solutions for the given problems. Only 2.8% of participants disagree

On the other hand, the survey contains a list of questions about the system features. In this section, the responses of the survey participants to the features will be presented:

- **Syntax error message** The responses to this feature show 83% of the participants felt that providing a REPL interpreters that print the result as message that indicates errors

that might code has could enhance learners in understanding of the coding exercises. It is believed that with the error messages that REPL print would be easier for learners to understand the error reason particularly for learners which are classified as practical learners.

- **Underlining syntax error** The responses to this feature show 57.2% of the learners and educators felt that providing a REPL interpreters that underlining syntax error might code has could enhance learners in understanding of the coding exercises. In addition, only 14.3% of the participants felt that providing this feature has no affect on their learning programming.

- **Syntax highlighting** The responses to this feature show 85.9% of of the learners and educators felt that providing a REPL interpreter that would highlight syntax could assist learners in understanding of the coding exercises.

- **Detailed error message** All the participants felt that offering detailed error message would help learners to avoid careless mistakes.This means providing detailed error messages found helpful in learning coding.

- **Identifying error locations** All the learners and educators that participated in this study agree that all online coding tutorials systems must have a code editor that offer identifying error locations.

- **Customized hints** The responses to this feature show 82.8% of the learners and educators agree that Customized hints should be applied in all online coding tutorials systems.

- **Visual map** The responses to this feature show 71.4% of the participants felt that having visualizing code editor will help them to learn coding.

- **Auto-completion** 100% of the participants agree that having coding editor that offers auto-completion that allow learners to write code more quickly and precisely.

- **Syntax-detailed editor** The responses to this feature show 57.1% majority of the learners and educators felt that providing code editor with syntax detailed editor in online coding tutorials system would enhance the learning journey. Only 14.3% of the participants felt no.

To summarize, the findings from the Likert scale items revealed that a significant majority of participants recognized the value of incorporating coding lessons and tutorials into these systems, as it was believed to enhance learners' understanding of coding concepts. This finding was particularly relevant for visual learners, who found it easier to grasp the basics

of programming language through this approach. Furthermore, the participants expressed mixed opinions regarding the inclusion of additional coding learning materials. While the majority agreed that offering supplementary resources such as books and extra learning materials would enhance their learning experience, a notable portion disagreed, considering this feature unhelpful in their educational journey. This discrepancy suggests the importance of considering individual preferences and needs when designing online coding tutorial systems.

The findings also highlighted the significance of assessment activities and quizzes. A large majority of participants viewed them as valuable tools for evaluating their coding skills and promoting critical thinking and problem-solving abilities. The ability to assess one's progress and knowledge through quizzes was seen as beneficial for self-evaluation and growth. Regarding the features of the code editor, participants expressed a strong preference for error messages and syntax highlighting. They believed that error messages generated by REPL interpreters, specifically those indicating syntax errors, facilitated a better understanding of coding exercises. Syntax highlighting in code editors was also recognized as a useful feature, aiding learners in identifying and differentiating coding elements more effectively. Furthermore, participants unanimously agreed on the importance of providing detailed error messages in the online coding tutorial systems. These messages were seen as valuable in helping learners avoid careless mistakes and fostering a deeper understanding of coding principles. Other features, such as identifying error locations in code editors, customized hints, visual maps, and syntax-detailed editors, received varying degrees of support from participants. The majority expressed their preference for these features, emphasizing their potential to enhance the learning journey and improve overall comprehension of coding concepts.

To conclude, the responses from educators' and learners' surveys were analysed. Findings showed that the vast majority of educators and learners were satisfied with the features presented in the first version of the evaluation instrument. On a more positive note, only few negative responses received showed that of the participants are not satisfied with some of the proposed features. For example, Quizzes, References materials, syntax-detailed editor and underlining error messages.

4.3.4.2 Qualitative findings

After testing the given system (LearnPython [175]), two follow-up open-ended questions were asked as shown in Appendix A to capture learners and educators thoughts about appropriate features to be in online coding tutorial systems. These questions are; the first open-ended question, '***What other helpful and usable features or characteristics of online coding tutorial systems would you suggest?*** and 10 participants responded to the second open-ended question ***Any other suggestions?*** This section presents a qualitative data gathered from these two open-ended questions which involved some interesting features from

participants. This qualitative data give in-depth understanding of the learners and educators needs in online coding tutorial systems.

Figure 4.2: Word cloud visualization of responses to question one (larger size words indicate more frequently repeated words and smaller size words indicate less frequently repeated words found in the participants' responses)

4.3.4.3 Identified themes from Q1

As shown in Figure 4.2, 27 responses had been received from the first open-ended question in the online survey. However , as presented in Table 4.4 and Table 4.5, only 24 responses were suggesting some suggestions, and they had been analysed based on themes and codes.

Theme1: Extra content lessons Adding more advance coding lessons was the main theme found from the participants' responses. For instance, one participant mentioned *"providing more 'optional' exercises so the learner can code different problems. The systems tested only provide one exercise for each section"*. This response indicated the need more coding problems to be provided to the learners who are using online coding systems. Another participant also suggested the same feature , he/she said *"additional practise exercises with several difficulty levels"*. This response indicates the need for additional coding exercises to the current coding exercises in LearnPython [175].

Theme2: Reflection notes Another supportive feature has been suggested from the participants in this study is providing reflection note as one participant describes reflecting on learning as: *" I think providing reflection box to write learning achievements might be a very helpful feature"*. Not only one participant suggested this feature, also another participant responded to the first open ended question and said *"reflection box-learning notes"*. These responses indicate the importance of providing reflection note box in online coding tutorial systems where the learners can record their learning reflection thoughts.

Theme3: Content organization Another suggestions has been noted from the participants responses is providing an organized coding lessons. As one participant mentioned *"Organizing content lessons"*. This response to the first open-ended question was short but it seems that this participant prefer to have an online coding tutorial system that has an organized coding lessons list that is organized from basic to advanced coding topics.

Participant responses	Codes highlighting participant ideas: Significant ideas	Themes: Main idea
” providing more ’optional’ exercises so the learner can code different problems. The systems tested only provide one exercise for each section”	Adding more coding exercises	Extra content lessons
”provide links to other websites so the learner can have easy access to other related content instead of searching the web again (e.g., links to GitHub, stackoverflow, geeks for geeks).”	Adding other similar websites	Extra resources
” provide other materials such as video or written steps about how to code a specific problem”	Adding other materials	Extra resources
” May be provide/highlight subsections, so the learner can see that there are different techniques under the section. For example, in loop section the subsection could be (for, while, etc).”	Adding more subsections	Extra content lessons
” Might be interesting to have a system that’s capable of exploring mistakes/misunderstandings /with/ a user - allow the user to reflect on their misconceptions.”	Ability to reflect on their understanding	Reflection note
” I believe that most of features you discussed in this survey are interesting.”	All of the Features	All of the Features
” You have covered all, thanks”	All of the Features	All of the Features
” additional practise exercises with several difficulty levels ”	Adding more exercises	Extra content lessons
”Providing the solution of each coding exercise.”	Providing solutions	Worked solutions

Table 4.4: Participants responses to the first open-ended question-with gray cells indicating the new features participants suggested (Part 1)

Participant responses	Codes highlighting participant ideas: Significant ideas	Themes: Main idea
All usable features have already been mentioned	All of the Features	All of the Features
”you have covered most of important features”	All of the Features	All of the Features
” I think covering basic concept of programming is an important characteristic to help beginners”	Basic concept	Extra content lessons
”The features mentioned are interesting”	All of the Features	All of the Features
” Provide many solutions for each problem, so that the learner could know multiple ways to solve a problem or even use specific line code for other problems”	Adding more exercises	Extra content lessons
” I think what mentioned is already a perfect way . Thank you ”.	All of the Features	All of the Features
”You have covered most of the helpful features”	All of the Features	All of the Features
”Providing solution for each coding exercise”	Providing solutions	Worked solutions
”Reflection box”	Ability to reflect on their understanding	Reflection note
” I think providing reflection box to write learning achievements might be a very helpful feature”	Ability to reflect on their understanding	Reflection note
” All the features provided are useful ”	All of the Features	All of the Features
”Organizing content lessons”	Organizing the content	content lessons organization
”reflection box-learning notes”	Ability to reflect on their understanding	Reflection note
”Content organization- extra lessons”	Organizing the content	Content organization
”more content”	Adding more lessons	Extra content lessons

Table 4.5: Participants responses to the first open-ended question - with gray cells indicating the new features participants suggested (Part 2)

4.3.4.4 Identified themes from Q2

From the second open-ended question only 10 responses had been received. However, as presented in Table 4.6, only three responses were shown some feedback that had been analysed based on themes and codes. However, there is no new supportive features can be added to the initial instrument from this question.

Participant responses	Codes highlighting participant ideas: Significant ideas	Themes: Main idea
”Improve the content itself.”	Improving content	Content lessons
”you have mentioned interesting features, good luck”	All of the Features	All of the Features
”Interesting study ”	All of the Features	All of the Features

Table 4.6: Participants responses to the second open-ended question

The main aim of providing the two open-ended questions in this online survey is to identify new supportive features from learner and educator perspectives that can be added to the first version of the evaluation instrument for online coding tutorial systems. The qualitative findings from this online survey are consistent with the earlier literature survey in Section 4.2, they strongly imply that the learners and educators are satisfied with the features proposed in Section 4.1. In addition, it is interesting to note that some the responses to both open-ended questions indicate satisfaction with most of the features.

Interestingly, three new features have been identified from the responses to the two open-ended questions. These are providing reflection notes, organizing content lessons, and adding extra content lessons. The main findings from the learner and educator perspectives in the study emphasize the importance of incorporating user perspectives in the design of the evaluation instrument for online coding tutorial systems. Participants from both groups suggested several enhancements for these systems, including the addition of a ”Reflection note” feature, improved content organization, and more coding lessons. These suggestions highlight the need to cater to learners’ individual needs, enhance their learning experience, and provide comprehensive and user-friendly resources. The incorporation of user perspectives in the design process is crucial as it ensures that the resulting tutorial systems align with the expectations and requirements of the target audience. By involving learners and educators in the decision-making process, the systems can be tailored to address their specific challenges and preferences, leading to more effective and engaging learning experiences.

These findings contribute to the overall research goals and objectives by providing valuable insights into the needs and expectations of learners and educators in the context of online

coding tutorials. By understanding their perspectives and incorporating their suggestions, researchers can develop more user-centered and effective educational tools. The study highlights the significance of user feedback in guiding the design and improvement of online coding tutorial systems, ultimately aiming to enhance the learning outcomes and experiences of programming learners.

4.3.5 The changes in version one of the instrument

In this study, the learners and educators suggested some suggestions and recommendations for improving online coding tutorial systems. Based on these recommendations, the initial instrument presented in Section 4.2 has been updated, with the updates in both categories discussed below.

- **Reflection notes**

  The responses revealed that some of the participants felt the need for reflecting on their learning journey. This could be achieved based on providing a reflection note feature. Reflection notes might help to enhance the learning experience and promote deeper understanding of the learning material. It provides an opportunity for learners to review their progress, assess their understanding, and think critically about their learning journey. Both educators and learners emphasized the importance of adding a ”Reflection note” feature to online coding tutorial systems. This feature would provide a space for learners to write about their learning achievements, reflect on their progress, and document their understanding of coding concepts. This suggestion indicates a desire for self-reflection and metacognitive learning, allowing learners to consolidate their knowledge and track their growth. The inclusion of a reflection note feature aligns with the broader educational research that highlights the benefits of reflection in promoting deeper learning and self-awareness.

- **Extra list of lessons**

  Another feature has been mentioned in the participant responses: adding more lists of coding lessons. Providing this feature might help learners to understand in depth programming concepts. Both educators and learners expressed the desire for an increased number of coding lessons within the online tutorial systems. They emphasized the importance of having a variety of coding exercises, scenarios, and practice opportunities. By offering a broader range of coding lessons, learners can gain exposure to different programming concepts, improve their problem-solving skills, and enhance their overall coding proficiency. This recommendation reflects the recognition that hands-on practice and ample coding exercises are essential for effective programming learning.

- **Content organization**

  Moreover, another feature has been mentioned in the participant responses: organizing coding lessons. This might help learners to understand programming concepts from the basic to more advanced concepts. Both educators and learners would like improved content organization within online coding tutorial systems. Participants expressed the desire for clear and structured lesson plans or modules, highlighting the importance of organizing coding lessons in a logical and coherent manner. By providing a well-structured instrument, learners can navigate through the content more easily and educators can ensure a comprehensive coverage of coding topics. This suggestion emphasizes the significance of effective instructional design and curriculum development in online coding education.

4.3.6 Second version of the instrument

As shown in Figure 4.3, the second version of the evaluation instrument for Coding Tutorial Systems is proposed. Conducting an exploration study by using fact-finding study (online survey) helped to improve the evaluation instrument in this research work. It also enabled adding new supportive features for online coding tutorial systems.

Components	Items
Syntax of programming languages	Syntax error messages
	Underlining syntax errors
	Syntax highlighting
Structure of code	Visual map
Understanding basic concepts	Lesson content
	Reference materials
	Worked solutions
	Quizzes
	Extra content lessons
	Content organization
Debugging	Detailed error messages
	Identifying error locations
	Customized hints
Dividing functionality into procedures	Auto-completion
Transferring algorithm to concrete implementation	Syntax-directed editor
Improving learning experiences	Reflection notes

Figure 4.3: The evaluation instrument for online coding tutorial systems version two based on initial facts finding study (design cycle two), red text indicates new feature added

This study fits in the analysis/exploration phase of the Design-Based Research approach used to structure this research project [1] [34]. It contributed to a better understanding of the supportive features for OCTSs that were identified in the review of the literature. This study presented the results of the online survey and a detailed exploration of participants' experiences (learners' and educators' experiences and feedback) on the identified features in the initial instrument presented in Section 4.1.

In addition, this section concludes the second design cycle of developing the evaluation instrument in this research. Nevertheless, there are still concerns about whether these identified features already exist in current online coding tutorials systems. Therefore, in the next chapter an analysis study will be conducted on selected OCTSs.

4.4 Instrument Design Cycle Three

This chapter presents the last study of the analysis/exploration phase in the design-based research approach [1] [34] that has been followed in this research work. The purpose of this study is to gain a comprehensive understanding of existing online coding tutorial systems and identify their strengths and weaknesses by examining the current state of these platforms. This section presents **the third design cycle** in this research, which is a second fact finding study (comparative study). The main purpose of this comparative study is to assess current online coding tutorials systems for the presence or absence of the supportive features that were identified in the version two of the instrument of online coding tutorial systems (as presented in previous chapter, Section 4.3).

This is an important study because looking at the existing online coding tutorial systems will give this research work more reliability regarding the needs of proposing evaluation instrument for online coding tutorial systems. Moreover, the importance of this comparative study lays behind checking whither there are features that current systems have and are not exist in the instrument and based on the finding from this comparative study the instrument will be updated. Lastly, this chapter proposes the third version (**draft three**) of the evaluation instrument for Online Coding Tutorial Systems.

4.4.1 Research question 3

This third study in the analysis/exploration phase addresses RQ3: **What are the supportive features that exist in current deployed online coding tutorial systems and absent in the instrument? Do the identified supportive features in the instrument exist in these system?** The aim is to improve the evaluation instrument for OCTSs based on a comparative analysis among the second instrument in Section 4.3 and seven selected current online coding

tutorial systems. In addition, from this comparative analysis new supportive features for online coding tutorial systems might be identified.

This study contributes to the knowledge area by investigating whether the features in the instrument version two presented in Section 4.3 exist or missing in current online coding tutorial systems, also whether they provide any of supportive features that are not mentioned in the instrument.

4.4.2 Study method

The approach used to improve the second version of the evaluation instrument for OCTSs was conducting a comparative analysis study [23]. Using this method, seven current online coding tutorial systems were selected and analyzed.

4.4.2.1 Procedure

The criteria for selecting these systems were drawn up on a systematic basis. Firstly, to ensure a representative sample of online coding tutorial systems, the selection process involved looking at the top ten programming languages by popularity on GitHub. These languages were identified as Python, JavaScript, Java, TypeScript, Go, C++, Ruby, PHP, C#, and C. Compiler-based toolchains were excluded from consideration as they are not particularly suitable for deployment in online coding environments. From the remaining high-level interpretive/scripting languages, namely Python, JavaScript, Java, TypeScript, Go, Ruby, and PHP, online coding tutorial systems were identified through a Google search using the query ”interactive tutorial X,” where X represented each programming language. The highest ranked links on Google that led to online coding tutorial systems were followed for further analysis.

Based on this method, the following online coding tutorial systems were selected for the software survey:

- LearnPython [175]
- TryJavaScript [159]
- LearnJava [173]
- Codecademy LearnTypeScript [48]
- Tour of Go [75]
- RubyMonk [160]

- LearnPHP [174]

These systems were chosen as representative examples within their respective programming languages and were deemed appropriate for novice programmers. By analyzing these selected online coding tutorial systems, new supportive features for OCTSs might be identified.

4.4.3 Data analysis techniques

The process of data analysis employed to evaluate and analyze the current online coding tutorial systems involved the selection of several systems, followed by an assessment of the presence or absence of identified supportive features. The evaluation criteria utilized in assessing the seven selected systems were based on a comparison between the list of features identified in the second version of the evaluation instrument and the features offered by each system. Each system underwent an examination in which it was reviewed and checked for the provision of each feature outlined in the list. Furthermore, in addition to evaluating the presence of the identified features, each system was also checked to determine if it offered any features that were not initially included in the proposed evaluation instrument. This comprehensive analysis aimed to identify both the adherence of the selected systems to the existing instrument and any potential additional features that could enhance the overall design of online coding tutorial systems.

By conducting this evaluation and analysis, the study sought to gain insights into how well the current systems align with the identified features and determine if there are any notable variations or innovative features not previously considered. This approach allows for a thorough understanding of the strengths and weaknesses of the existing systems and provides valuable information to inform the design of improved online coding tutorial systems. Through this rigorous evaluation process, the study aimed to contribute to the ongoing improvement and advancement of online coding tutorial systems by identifying the features that effectively support novice learners and uncovering potential areas for further enhancement. By incorporating these findings into the design of future systems, developers and educators can create more effective and comprehensive platforms that cater to the evolving needs of novice learners in the coding community.

4.4.4 Study findings

4.4.4.1 Assessment of deployed online coding tutorial systems

Features that are not exist in the deployed systems

Supportive Features	*LearnPython*	*TryJavaScript*	*LearnJava*	*Learn-TypeScript*	*Tour of Go*	*RubyMonk*	*LearnPHP*
Syntax error messages	✓		✓	✓	✓	✓	✓
Underlining syntax errors							
Syntax highlighting	✓		✓	✓		✓	✓
Visual map							
Lesson content	✓	✓	✓	✓	✓	✓	✓
Reference materials	✓	✓	✓	✓			✓
Worked solutions				✓		✓	✓
Quizzes	✓						
Detailed error messages	✓	✓	✓	✓	✓	✓	✓
Identifying error locations	✓		✓		✓		✓
Customized hints						✓	
Auto-completion							
Syntax-directed editor	✓	✓	✓	✓	✓	✓	✓
Reflection notes							
Extra lesson content	✓	✓	✓	✓	✓	✓	✓
Content organization	✓	✓	✓	✓	✓	✓	✓

Table 4.7: Comparative analysis of inclusion of supportive features across seven tutorial systems (grey row indicates complete absence of feature in all systems)

As shown in Table 4.7, seven systems were analysed across 16 supportive features have been identified in the second version of the evaluation instrument for OCTSs that is presented in Section 4.3. It is obvious that most of the identified features exist in some systems. For instance, all seven selected systems provide a syntax-directed editor that helps novice learners be guided when to use a particular syntax by providing templates. In addition, we find that five of the selected systems have parsers that help novice learners to detect syntax errors by highlighting errors in the REPL. For example, LearnPython [175], LearnJava [173], Learn-TypeScript [48], RubyMonk [160] and LearnPHP [174] provide syntax highlighting as the user enters text. Moreover, we find that TryJavaScript runs a custom parser that provides extended, beginner-friendly feedback when errors occur.

However, highlighted rows in Table 4.7 indicate four supportive features that are not provided by any of the studied systems. For example, underlining syntax error, reflection notes, visual map and auto-completion. In addition, we also note that two features, quizzes and customized hints are only provided by a single system under study. In summary, as shown in Table 4.7, the findings of this analysis provide supporting evidence that the current online coding tutorial systems do not fully address novice programming difficulties. These findings

represent the first direct demonstration that online coding tutorial systems are not ideal for novice learners since they lack significant supportive features that have proved to be helpful for novice programmers to overcome specific programming learning difficulties. or not. One interpretation of these findings is that current online coding tutorial systems missing some features For instance, to help novice learners to understand how to divide functionality into Procedures and support them to develop additional functions/procedures that can be used to call in exactly the same way as the built-in functions/procedures[223]. In addition, according to [55], to help novice learners to understand programming concept, basics assessment activities and quizzes are useful technique to let novice learners test their understanding. In addition, providing questions along with interactive exercises can help to reinforce concepts and measure the novice learner'understanding. Moreover, providing customized hints based on specific learner errors can help learners to repairing bugs found in their programs [166]. In other words, when a novice programmer types a segment of code and the editor shows an error, the hints feature will provide some hints to help the learner to find and fix the bug in the code[166]. According to [197], these graphs make learning programming languages are easy to understand and allowed novices to learn the structure of code without making syntax errors. In addition, [214] mentioned that using graphical representation can help novices to understand the code structure by visualizing how the code structured. According to [166], visualization tools that illustrate code structure and program execution have been helpful for novice programmers to develop their understandings of how computer programs function and the notional machine. While visualization tools are helpful to make the notional machine and code execution visible, they may also increase learners' cognitive load and so make things more complex [166]. Regarding to underlining syntax errors, this feature has been proved as an important feature for novice learners to show them error messages to avoid any syntax errors [29].

One implication of these findings is the urgent need for enhanced support for novice programmers through a more considerate design process for online coding tutorial systems. Therefore, in this research, an evaluation instrument will be developed based on novice learners perspectives. In a follow-up phase, we contacted developers of the five selected systems and explained our research findings, including the identification of 'must-have' features to support novice learners. We asked each developer directly why they did not incorporate such supportive features into their platforms.

Our study reinforces an important point. Malmi et al. [126] identify three sets of stakeholders for programming learning tools, namely learners, educators and developers. Problems for learners occur due to communication difficulties and priority mismatches between educators and developers. It appears that many online coding tutorial systems are created by programming language developers (whether original language designers or enthusiastic advocates) with minimal input from computing education experts. Anecdotally, this is the case with the

TryHaskell platform.

Features that are not present in the second version of the instrument

The seven selected online coding tutorial systems were assessed with regard to the supportive features in the second version of the evaluation instrument. These seven systems were analysed and checked in order to identify any feature in these systems that is not provided in the second version of the evaluation instrument. As result of this analysis study, one supportive feature was identified that is present in two of the current systems and is not covered in the second version of the instrument (see Section 4.3.6). This new feature is offering several options of programming learning systems. For example, the candidate online coding tutorial system called LearnPython [175] gives the option for the novice learner to choose from several options of programming languages such as Java, Ruby and JavaScript.

4.4.5 The changes in version two of the instrument

After conducting this comparative analysis study, only one supportive feature has been explored in this section. In this section, the update on the version two of the instrument presented in Section 4.3.6 will be discussed. The findings from this second fact finding study in the analysis/exploration phase enabled the identification of new feature in the instrument. This new feature is as follow:

- **Offering several programming languages**

  As shown in Figure 4.4, LearnPython [175] provide several options of other programming languages. It is important to provide a diverse range of options to cater to different novice learner preferences. This supportive feature might be helpful for learners to have effective programming learning journey since there are several options of programming languages.

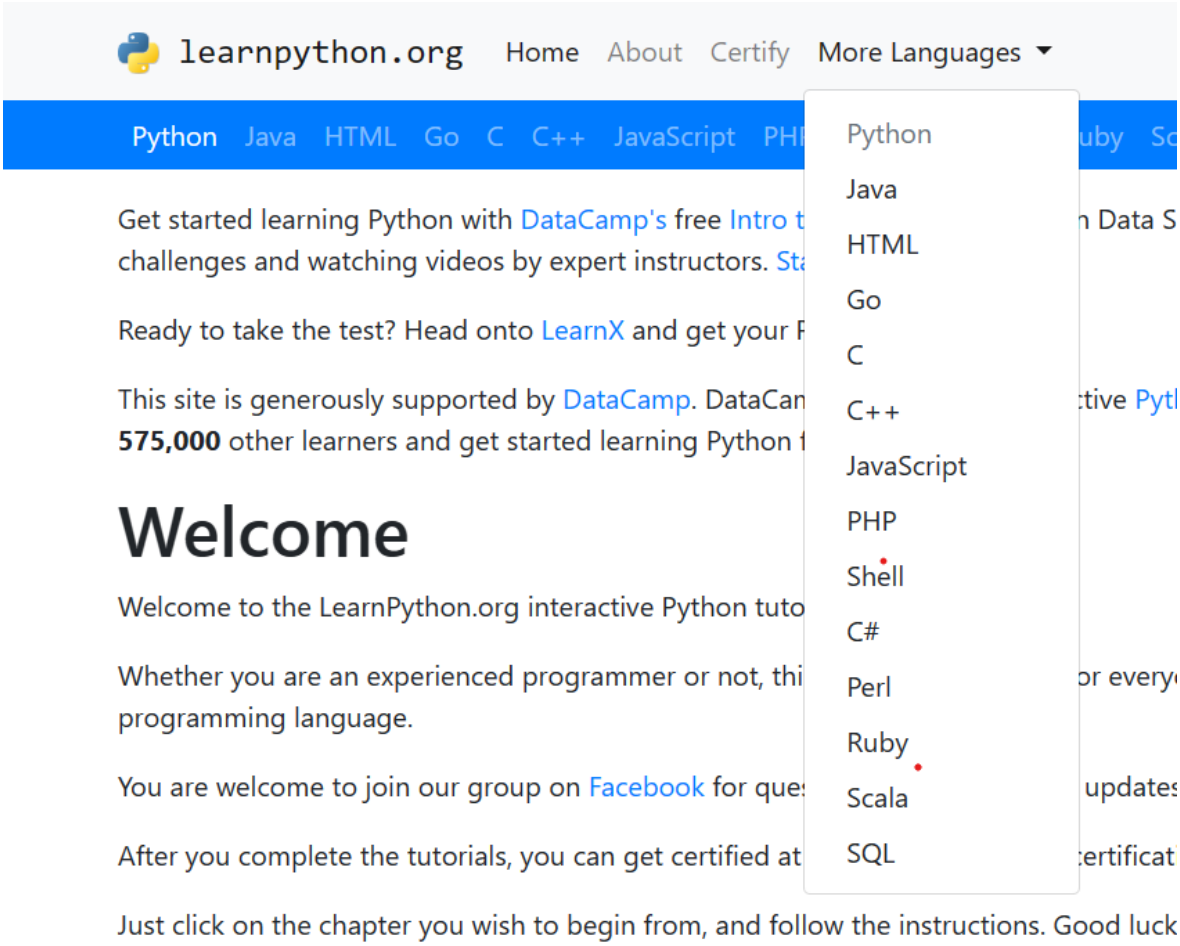

Figure 4.4: A screenshot of an online coding tutorial system that provides several programming languages

4.4.6 Third version of the instrument

As a result of the case study in this section, the second version of the evaluation instrument of online coding tutorial systems that is presented in Section 4.3 has been updated. Figure 4.5 presents the third version of the evaluation instrument for online coding tutorial systems that is the main aim of this research work.

Components	Items
Syntax of programming languages	Syntax error messages
	Underlining syntax errors
	Syntax highlighting
Structure of code	Visual map
Understanding basic concepts	Lesson content
	Reference materials
	Worked solutions
	Quizzes
	Extra content lessons
	Content organization
Debugging	Detailed error messages
	Identifying error locations
	Customized hints
Dividing functionality into procedures	Auto-completion
Transferring algorithm to concrete implementation	Syntax-directed editor
Improving learning experiences	Reflection notes
Offering several programming languages	Offering several programming languages

Figure 4.5: The evaluation instrument for online coding tutorial systems version three based on systems analysis-case study (design cycle three), red text indicates new feature added

This study reported in this chapter fits in the analysis/exploration phase of the Design-Based Research approach [1][34]. It contributed to investigate the conformance of the current systems to the identified requirements. A comparative study presented the results of analysing the current systems. The comparative study results contributed to the second draft of the evaluation instrument for Online Coding Tutorial System in Section 4.3 by updating the instrument and adding one new feature. This section concludes design cycle three of the research. It also concludes Phase three: Analysis and Exploration of GMDR [1][34]. To address this gap, the research aims to develop an evaluation instrument based on novice learners' perspectives. Additionally, the chapter discussed the intention to contact the developers of the selected systems to share the research findings and discuss the reasons behind the absence of these supportive features.

4.5 Instrument Design Cycle Four

In this research work, the approach followed is the design-based research model that is created by [1] and [34] that is originally adopted by [125]. This design-based approach has three phases, analysis/exploration phase, design and implementation, evaluation/revision phases. This section presents the design and the implementation phase for developing an Online Coding Tutorial System Prototype called Python OCTS [165]. The system prototype development is totally based on the third version of the evaluation instrument that presented in Section 4.4 and that was produced in the third design cycle. Developing an online coding tutorial system prototype that contains most of the identified features contributed to the third draft of the evaluation instrument, enabling the understanding of real users needs and subsequently generating the features specifications for designing Online Coding tutorials Systems. It is important to develop a system prototype and to conduct an evaluation study. This because in design-based research, a design must be implemented to achieve a goal. Providing a prototype for an invention can answer questions about whether the goal of the research has been achieved [66]. Therefore, conducting this study contributes to the the objective of this research work by providing a higher fidelity implementation to test if the goal of developing efficient educational intervention in this work has been achieved or not.

4.5.1 Design and development of "Python OCTS"- an online coding tutorial system prototype

The purpose of this research is to investigate the supportive features that can online coding tutorial systems provide in order to support programming novices in their learning journey. Firstly, in this research work a systematic literature review of supportive features was carried

out to help to identify the initial list of features and to develop the first version of the evaluation instrument that is presented in Section 4.2. In addition to that, user research techniques were applied throughout this research life cycle to better understand learners' and educators' needs because the focus should be on them to understand whether users are satisfied with the identified features of online coding tutorial systems. An initial fact-finding survey for both learners and educators was distributed (see Section 4.3) and both educators and learners were given the opportunity to input through this survey. This data gathering of relevant information enabled a clear understanding the supportive features for the Python OCTS prototype. Additionally, the qualitative data collected from this survey contributes to the development of the evaluation instrument in this research work.

Moreover, the analysis of the seven current online coding tutorial systems also enabled the identification of the issues with existing systems and how is developing Python OCTS [165] prototype could help overcome these limitations and offers a prototype for the missing features in current systems. For instance, Learnpython [175], does not incorporate a feature that enables novice learners to leave their learning notes while they learning through the coding lessons. TryRuby [94] is one popular Online Coding Tutorial System that is widely used. Although this case study contributed to the development of the evaluation instrument of OCTSs by adding new features.

This section presents the list of features necessary to design Python OCTS (system prototype). Following the design process approach in this research, the result of the initial systematic review of the literature (Section 4.2), the results of the fact-finding study (Section 4.3), and the results of the comparative study (Section 4.4) and the third draft of the evaluation instrument for online coding tutorial systems were used in order to design and develop the system prototype. Therefore, the list of features used in this section to design and develop the online coding tutorial system prototype is the third version of the instrument presented in Section 4.4.

4.5.1.1 Defining Python OCTS

The computing education literature previously highlighted the importance of providing novice learners with interactive coding platforms that help them in learning programming (Chapter 2), it is necessary to look at how ensure the effectiveness of such systems. In this research work, a set of features have been identified to be in online coding tutorial systems to have effective systems that support novice learners in their programming learning journey. This design and implementation phase in the instrument design cycle aims to develop system called Python Online Coding Tutorial System (Python OCTS) to evaluate after that the features provided in this system prototype. Python OCTS [165] consists most of the features that have been identified during the design process of the instrument in this work. These

include content and technical-based features. The proposed online coding tutorial system contains all the above-mentioned features is illustrated as in Figure 4.6.

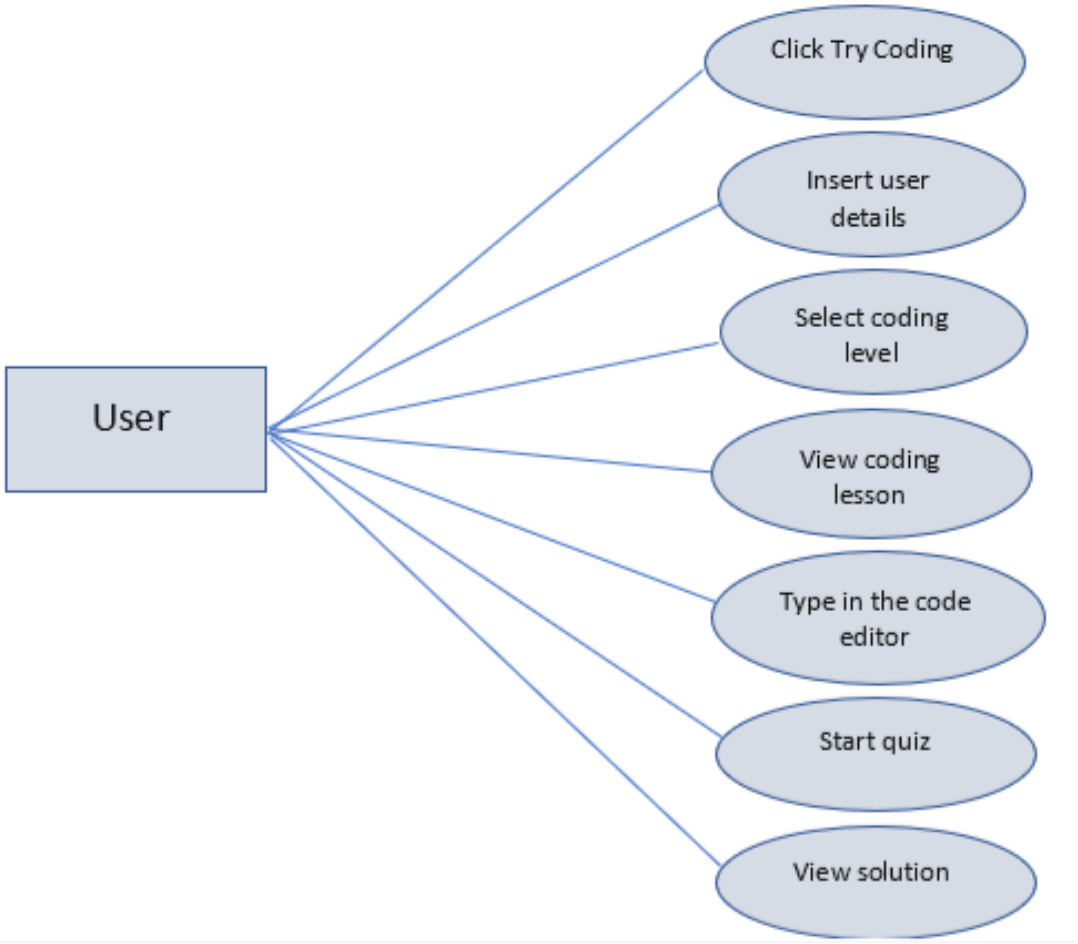

Figure 4.6: Proposed Python OCTS

4.5.1.2 Python OCTS architecture

The underlying objective was to develop a high -fidelity online coding tutorial system. This section explains the Python OCTS architecture that enables linking the basic structure of the components that the system comprise and the communications between these components, and describing the different features that comprise it. According to [183], system architecture is a description of how a software system is organised. To access the system URL via the internet using their browser in order to log in to the system (https://python-octs.herokuapp.com/). Each time a user logs into the web interface of the Python OCTS, the user is authenticated with the server. The system itself is hosted by a cloud platform called Heroku [93]. This section only provides a brief overview of the system prototype. It is designed to test the third version of the evaluation instrument in Section 4.4. It is also designed to gives real users the possibility to try most features proposed in this research work. Fig-

ure 4.7 illustrates the general architecture of the Python OCTS that shows the data transfer between the users and the system.

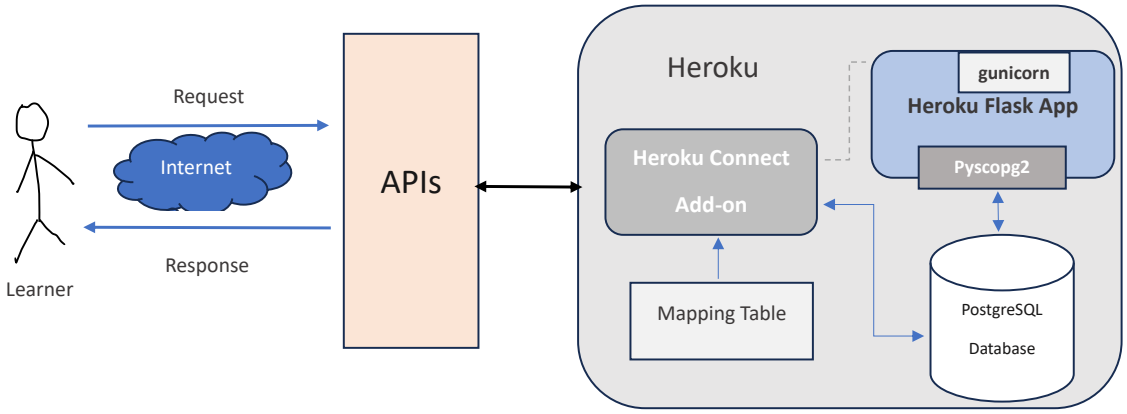

Figure 4.7: Python OCTS architecture

4.5.1.3 User interface design

The tool used to design and develop the Online Coding Tutorial System prototype' user interface [165] is **Visual Studio Code (VS Code) [210]** . Visual Studio Code platform is a free and open-source source code editor developed by Microsoft [136]. It is widely used by developers for various programming languages and platforms, however, in this system development Python programming language was used. Additionally, VS Code [210] provides a lightweight coding environment with powerful features and a customizable user interface. Additionally, it is a popular code editor that is widely used for web-based system development, and it offers a range of features and extensions that make it easy to write, debug, and deploy code for web applications. The reasons why this tool was chosen in this research work to develop the system prototype, is because it has very helpful features for intermediate developers. For instance, the code editor in this platform offers several helpful features such as syntax highlighting, code completion, and intelligent suggestions, making it easier for developers to write code efficiently and with fewer errors. In addition, it supports a wide range of extensions, which are add-ons that enhance its functionality. For instance, debugging capabilities for multiple programming languages. Developers can set breakpoints, inspect variables, step through code, and analyze run time behavior to identify and fix issues in their code. Therefore, VS Code was a great choice for developing this web-based system prototype (Python-OCTS [165]) with using Python programming language.

4.5.1.4 Hosting technology

The hosting technology used in this research work to host the online coding tutorial system (PythonOCTS [165]) is Heroku [93]. It is a cloud platform that offers hosting and deployment services for web-based systems. It provides a platform as a service (PaaS) model, which means that it abstracts away the underlying infrastructure and allows developers to focus on building and deploying their applications without worrying about server management or configuration. This hosting platform was selected because it provides a scalable and reliable infrastructure to ensure that the system prototype is accessible to users in the evaluation study that will be discussed in this section. In addition, it simplifies the deployment process by providing easy-to-use tools and integrations. In addition, the database used in this work to store the users interactions with the system prototype is Heroku Postgres that Heroku [93] provides. It is an open source database as a trusted, secure, and scalable service that is optimized for developers. Developers can build engaging, data-driven apps while relying on Heroku's expertise and fully managed platform to build, operate, secure, and validate compliance for their data stack.

4.5.1.5 Development of the Python OCTS

After designing the system prototype and selecting the appropriate development platforms and technologies, the system prototype was developed based on the supportive features presented in the third version of instrument. This section only provides a summary and graphical representation of the supportive features in Python OCTS. Figure 4.8 shows the homepage of the system prototype that acts as a starting point for new and returning users/novice learners, providing an overview of everything it offers. On the left side, the pink bottom moves the user to the registration page.

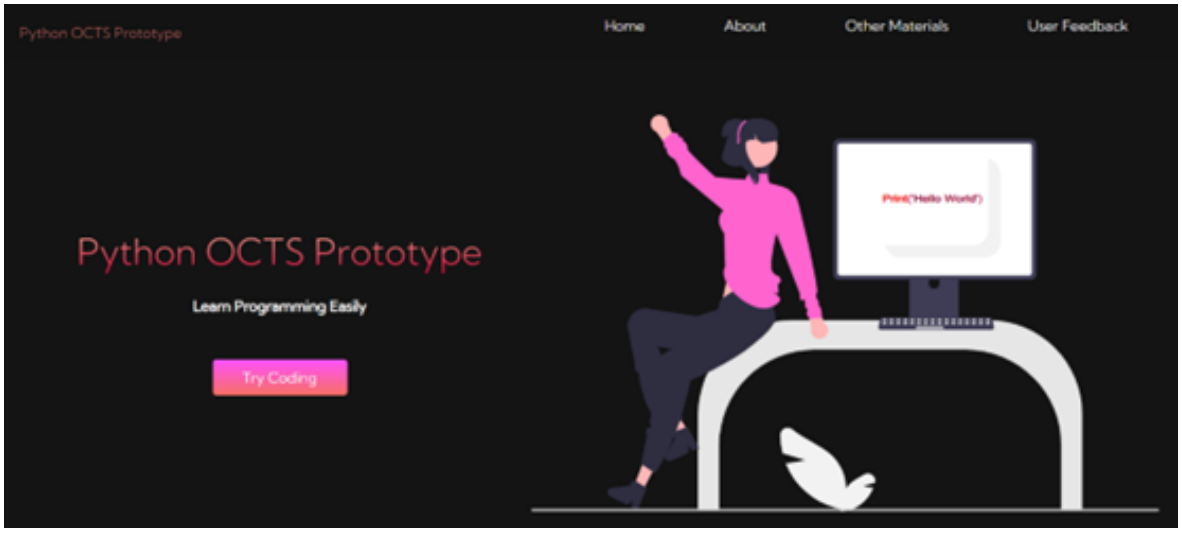

Figure 4.8: The home page of the system prototype

In the registration page or the sign-up page users can request themselves and login anytime without need to start from scratch as shown in Figure 4.9.

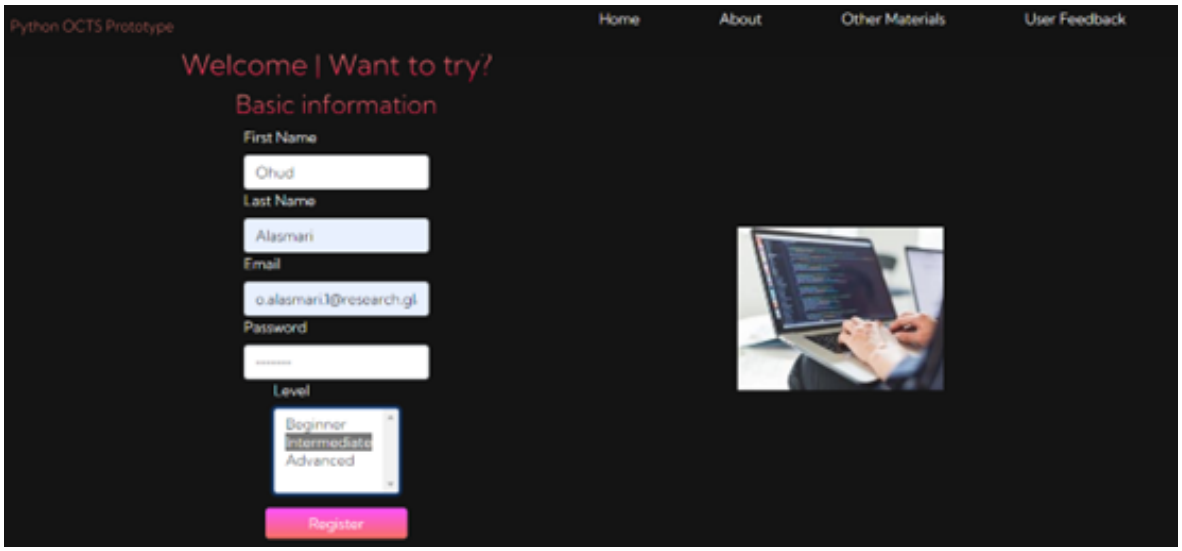

Figure 4.9: The registration page in the system prototype

After the user login the list of lessons of coding will be shown as shown in Figure 4.10. In this system prototype, only four coding lessons were added to the system to give the users the ability to know how the online coding tutorial system should look like. In addition, as shown in the top of the page there are four button that allow users to moves to other pages.

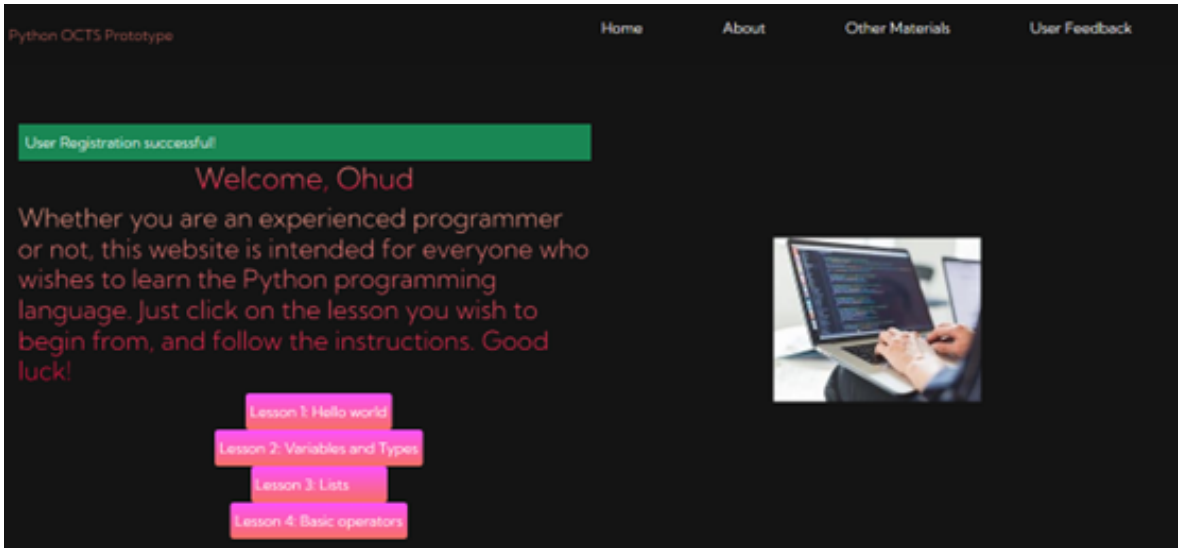

Figure 4.10: The coding lessons list in the system prototype

After clicking on ”Other Materials” button, this page will be shown as shown in Figure 4.11. It contains a list of helpful resources such as books and other learning materials.

Figure 4.11: The other materials page in the system prototype

The other button goes to a page containing the user feedback survey as shown in Figure 4.12. This survey related to the evaluation study that was conducted to test the satisfaction of the users toward the system prototype .

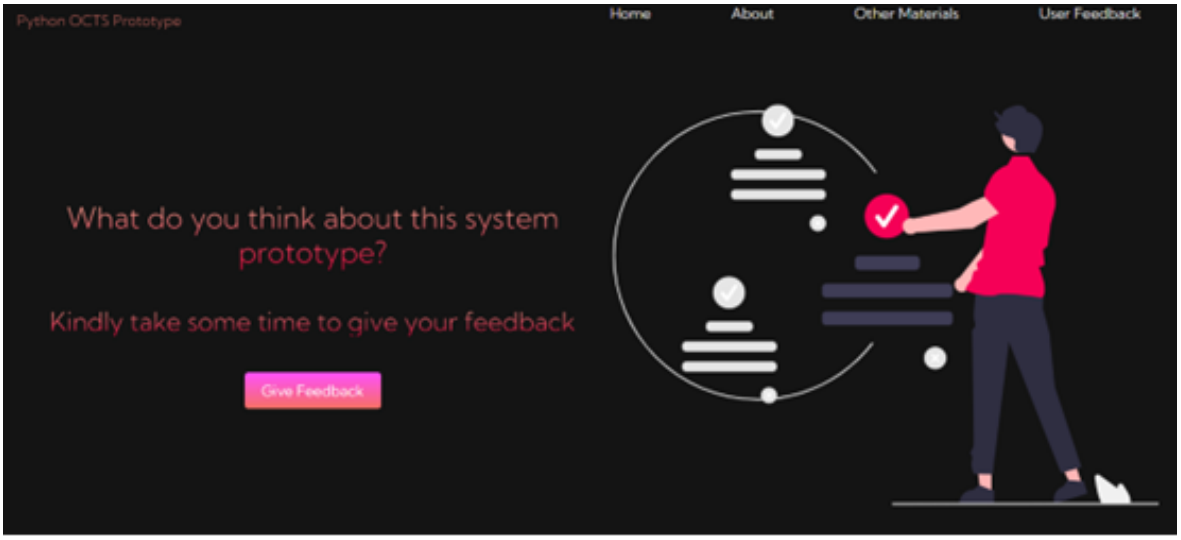

Figure 4.12: The users feedback page

As shown in Figure 4.13, the first lesson page contains the first programming learning lesson where the learners learn how to develop the first program in Python called “Hello, World!”. In addition, as shown in Figure 4.13, the main feature that is developed to provide other features is the built-in source code editor. In this system prototype, the trinket code editor was embedded in order to give the system prototype users the ability to test the other features such as syntax highlighting, syntax error message and underlining syntax errors.

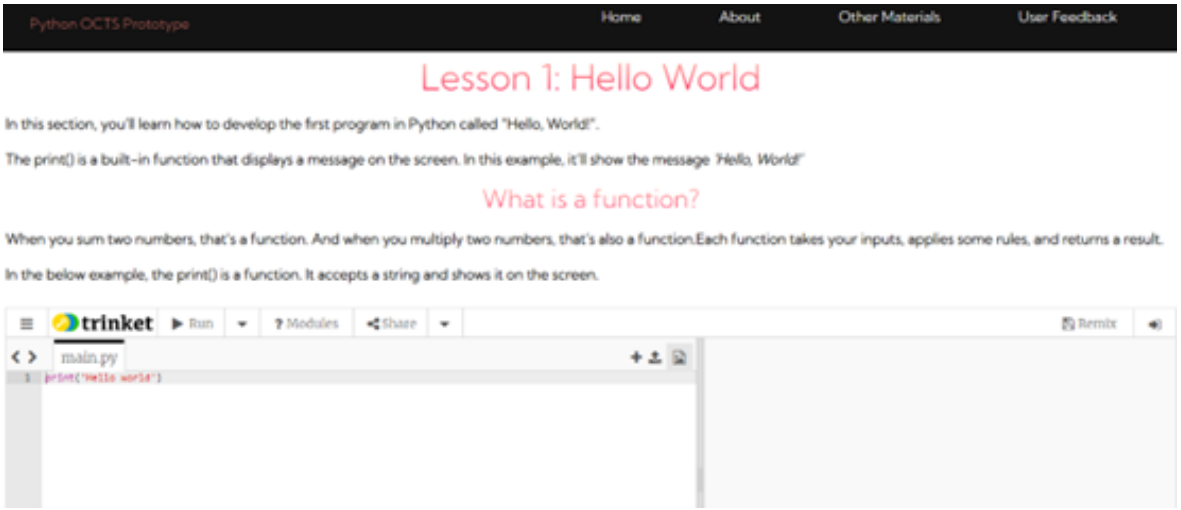

Figure 4.13: The embedded code editor

As shown in Figure 4.14, the system offers quiz feature at the end of each coding lesson to test the user coding skill.

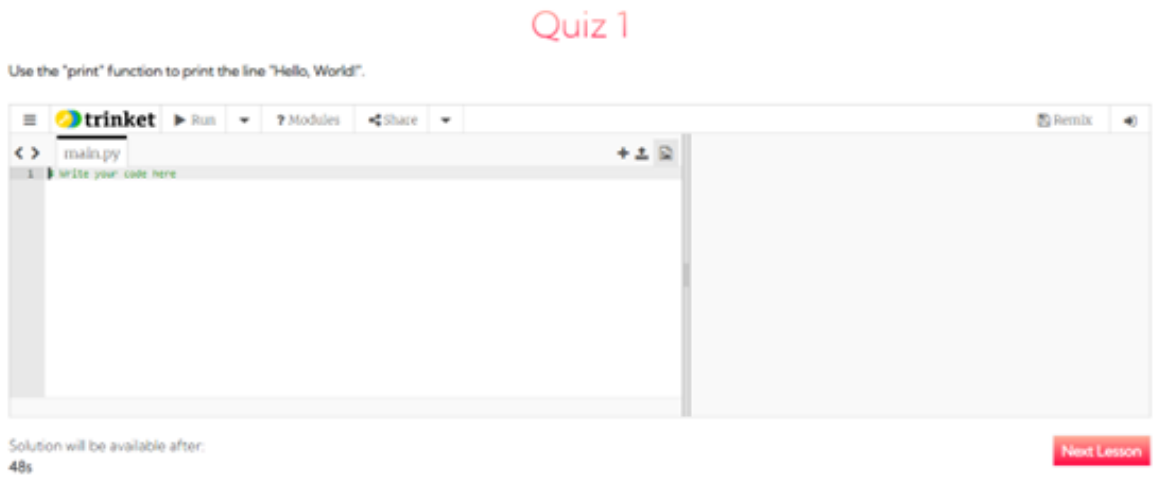

Figure 4.14: Quiz on the first lesson

After that as shown in Figure 4.15, the system offers solution feature at the end of each coding lesson below the quizzes to help users to understand each quizzes gaven.

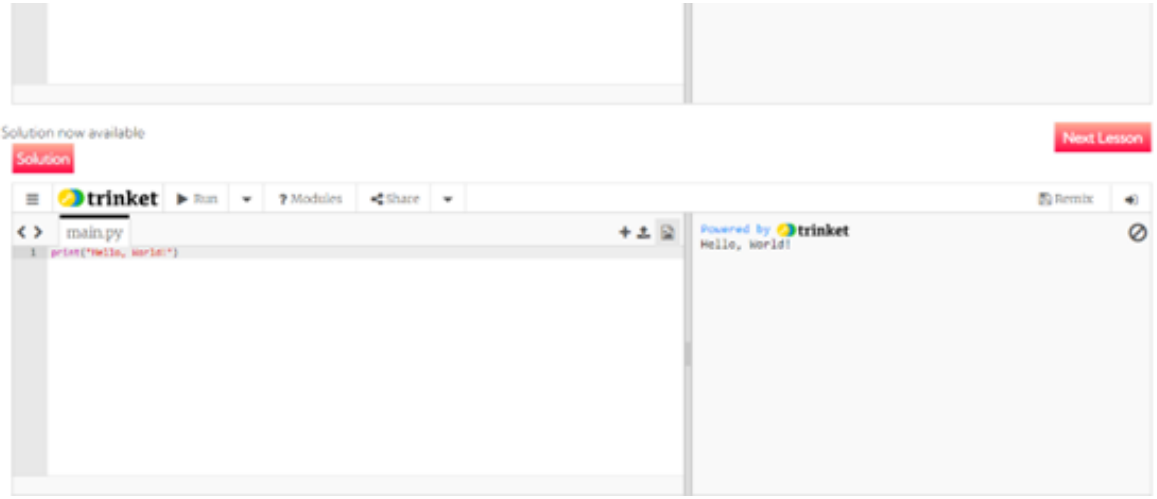

Figure 4.15: Solution of the quiz

In addition, as shown Figure 4.16 the Python OCTS provides a visual map feature that might help the users to see how the program code fits together without reading through files and lines of code. With these visual maps, the organization and relationships in the code, including its structure and its dependencies will be shown.

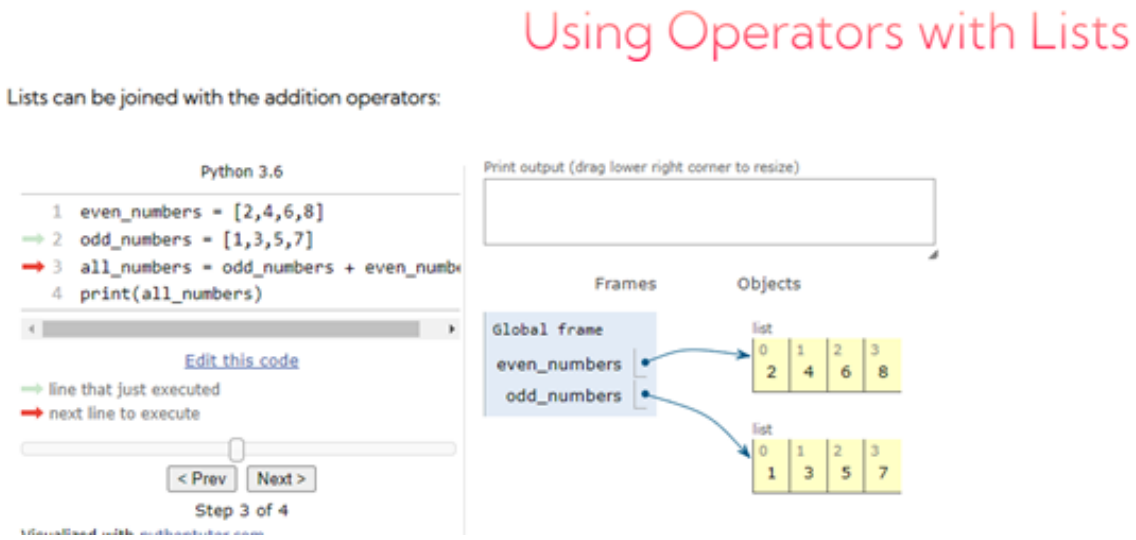

Figure 4.16: The visual map feature

As shown in Figure 4.17, in the end of each coding lesson page there is a reflective learning box. In this box, each learner can write down their reflections on their coding learning journey.

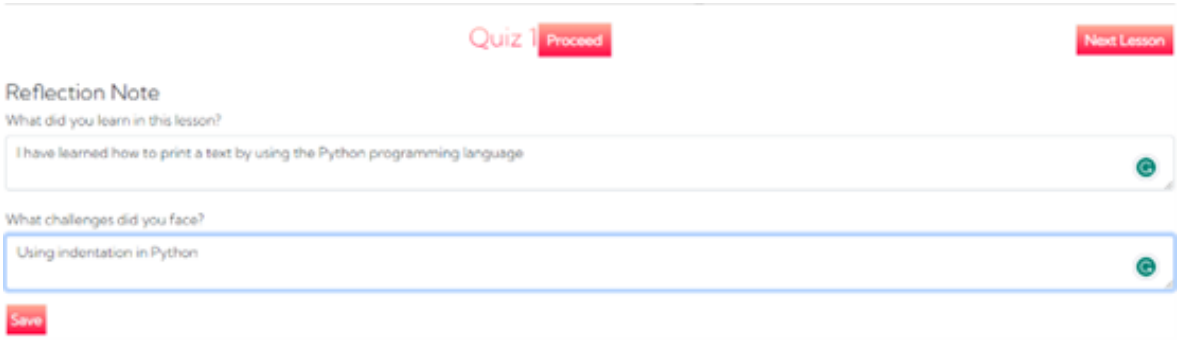

Figure 4.17: The reflection note box

In the Python OCTS, a code editor has been embedded to allow the users the ability to practice coding. This code editor provides several supportive features, such as shown in Figure 4.18, the syntax errors will be shown when the user type a wrong syntax.

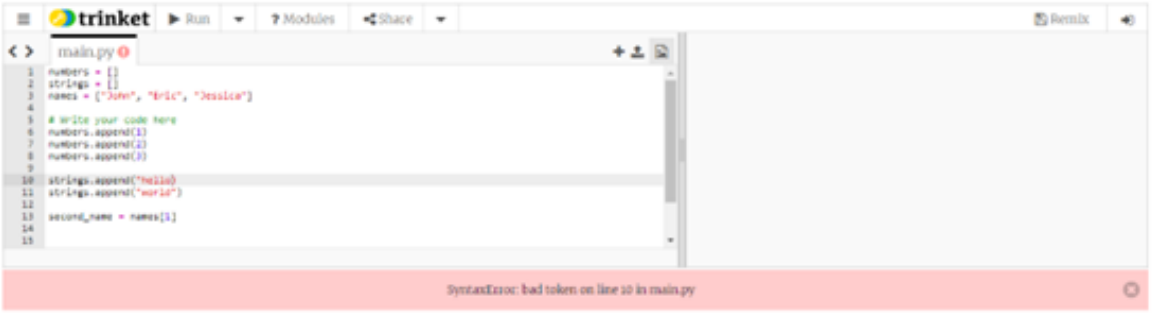

Figure 4.18: Syntax errors messages

In addition, another feature the embedded code editor provides which is ”Hints”. It displays when some segment of code is missing as shown in Figure 4.19.

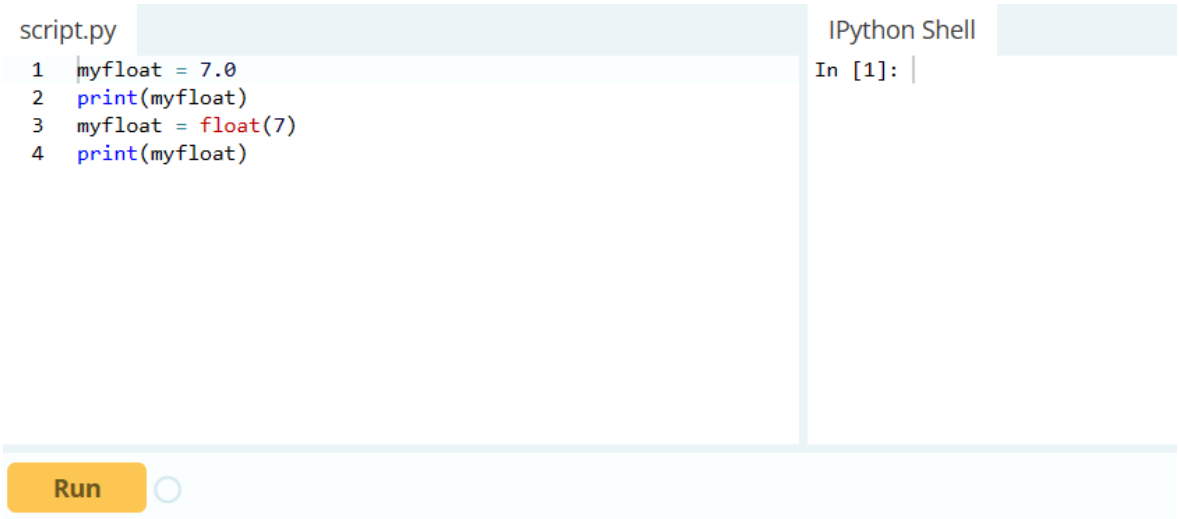

Figure 4.19: Hints

4.5.2 System features checklist

After developing Python OCTS, a comparative study (see checklist in Table 4.9 has been done in order to check if Python OCTS has been containing at least most of the identified features in the third version of the evaluation instrument that have been presented in Section 4.4. As presented in Table 4.9), three features from the third version of the evaluation instrument were not provided in Python OCTS; several programming languages, underlining syntax error and auto completion. The reasons of the absence of these supportive features are; underlining syntax error and auto-completion were not possible to be provided by the code editor that was embedded in the system prototype. In terms of providing other programming languages, it was considered to just develop a system prototype with few pages that can real users try.

Identified features in Section 4.4.6	Does the system prototype contains the feature?
Lesson content	✓
Extra list of lessons	✓
Reference materials	✓
Worked solutions	✓
Quizzes	✓
Reflection notes	✓
Several programming languages	
Content organization	✓
Syntax error messages	✓
Underlining syntax errors	
Syntax highlighting	✓
Visual map	✓
Customized hints	✓
Auto-completion	
Detailed error messages	✓
Identifying error locations	✓
Syntax-detailed editor	✓

Table 4.9: Comparative analysis of identified features in the third version of the instrument in Section 4.4 across the system prototype features (grey row indicates absence of feature in the system prototype)

4.5.3 Python OCTS and existing online coding tutorial systems

To summarise, the online coding tutorial system prototype developed in this section contains most of the features identified in this work while the current online coding tutorial systems such as TryRuby [94] and LearnPython [175] are missing some of the features in the third version of the evaluation instrument. For instance, Python OCTS [165] provides several supportive features that are found helpful such as reflection notes and visual map. These two features are not provided by LearnPython [175], TryRuby [94] and all the systems selected and analysed in Section 4.4. In addition, other features provided by Python OCTS are missing in most of the selected current systems such as customized hints, quiz and solutions.

The import of this system prototyping development is that a realistic environment for users to test most of the features identified in the evaluation instrument for online coding tutorial systems can be provided. This contributes the main aim of this research by improving the instrument based on real feedback on most of the features that are missing in current online coding tutorial systems.

This section presents the evaluation phase in design-based research approach that is followed in this work [1] [34]. In addition, this phase covers the fourth design cycle for developing the evaluation instrument for OCTSs which is the main aim of this research work. The main purpose of this section is to improve the evaluation instrument of online coding tutorial systems based on a real users' feedback from evaluation system prototype that provides most of the features in the instrument. In this section, the third version of the evaluation instrument of online coding tutorial systems proposed in the previous Section 4.4 will be updated in Section 4.5.10 which is version four of the instrument.

4.5.4 Research question 4

This evaluation study was conducted in order to improve the evaluation instrument for OCTSs based on evaluating Python OCTS [165]. In addition, this study addresses RQ4: **Building on our research findings, what would an online coding tutorial system look like? Based on a prototype implementation, to what extent are typical learners satisfied with the features of such an online coding tutorial system?**

This study contributes to the knowledge area by improving the development of the evaluation instrument based on the evaluation of real users that use an online coding tutorial system that provides most of the features in the instrument proposed in (Section 4.4.6). In addition, this study contributes by proposing the last version of the evaluation instrument.

4.5.5 Study method

In Design-based research, the evaluation phase capture the users satisfaction toward produced inventions [16]. In this evaluation study, firstly, an analysis study has been conducted in order to select the appropriate methods and data analysis techniques to evaluate the system prototype in programming learning and teaching domain. As presented in Appendix B, a study was reproduced exactly as the analysis study produced by Sheard et al. [187]. The aim of this analysis study is to investigate what is the most effective data collection method and data analysis techniques used in the programming education literature.

As results of the analysis study presented in Appendix B, in this evaluation study, two different data collection methods were used. As shown in Figure 4.20, mixed methods have been used to evaluate Python OCTS. Firstly, using an online survey data and secondly using log file data. The following sections describe in detail the methods used for this evaluation study. This study provides an opportunity to test the elements of the third version of the evaluation instrument for the online coding tutorial systems that has been offered in Section 4.4.

The methods used in this evaluation study are illustrated in Figure 4.20. Firstly, an online questionnaire was used as the primary tool for quantitative and qualitative data collection. The online questionnaire contains of several sections as presented in Appendix B. For instance, consent form, pre-testing questions, and testing questions. Before distributing the online survey, a pilot study was conducted in order to test the clarity of this online survey evaluation questions. Additionally, after the participants went through Python OCTS [165] and completed the online survey questions. The users interactions with Python OCTS were recorded in the database provided by Herkuo [93]. This log file data collection aims to give some insights how the users interacted with the system prototype' features.

4.5.5.1 Participants

The target participants in this evaluation study were users with different programming backgrounds. By distributing the online survey through various platforms such as WhatsApp, email and Teams to attract participants with diverse programming backgrounds. Initially, the target sample size was set at around 200 participants to gather a wide range of feedback and suggestions from real users on the Python OCTS [165]. Nevertheless, the actual sample size for this evaluation study turned out to be only 103 participants.

The sample size was smaller than expected number of participants and that might because of several factors. Firstly, it is possible that some of the individuals who received the online survey were not interested in trying Python OCTS out and to fill the sections of the online survey. Moreover, the timing of distributing the online survey could have influenced the participation rate because this evaluation study was conducted at the end of 2022. If the survey was sent during a period when potential participants were preoccupied with other academic commitments, it might have impacted their ability to complete the questionnaire.

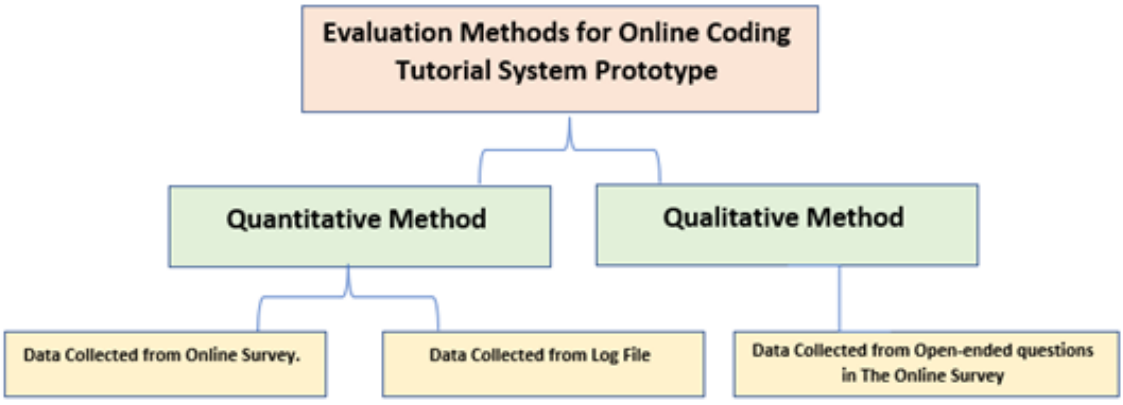

Figure 4.20: The evaluation methods that have been used for evaluating Python OCTS

4.5.5.2 Online survey

During the evaluation study, the researchers conducted an online survey to gather feedback from programming novice learners and educators. This survey aimed to collect valuable insights and feedback regarding Python OCTS [165]. The study was conducted over a duration of three months, specifically from November 2022 to January 2023 after receiving an Ethical approval from the College of Science and Engineering and the number of the application is (300210254). This online questionnaire was distributed randomly by using different platforms such as Teams and What's up. Participants were requested to provide their feedback based on their experiences and interactions with the system prototype (Python OCTS). The online questionnaire was carefully designed and structured based on the features provided in the third version of the evaluation instrument presented in Section 4.4. The online questionnaire contains four sections as presented in Appendix B:

- The first section is a consent form.
- The second section contains pre-testing questions (demographics questions).
- The third and the fourth sections contain a set of testing instructions and pro-testing questions to test the system' features.

4.5.5.3 Pilot study

Before distributing the online survey, a pilot study was conducted in order to test the clarity of the survey evaluation questions. The pilot study was performed between September and October 2022. Five PhD students from the School of Computing Science participated in this study, and the information gathered was enough to enhance the questions of the online survey.

4.5.6 Data analysis techniques

The current evaluation study used two methods to investigate user interaction with the system and measure user satisfaction. The first method, users' behaviors were observed and recorded in various aspects, including frequency distribution for users, duration of visits, pages visited, code editor usage, quiz visits, solution views, and other materials accessed. This phase utilized log data to capture user actions and interactions.

The second method, users were testing the system and providing their feedback through a questionnaire. The responses from the questionnaire were then analyzed and compared with the results obtained from the log data in the first phase. The researcher aimed to identify any similarities or differences between the two sets of results.

By combining both quantitative and qualitative data from the observation of user interactions and the analysis of questionnaire responses, a comprehensive understanding of user behavior and satisfaction with the system can be gained. This approach allows for a deeper evaluation of the system's effectiveness and provides insights for potential improvements or modifications to enhance the user experience.

In analysing qualitative data, **a thematic analysis** was used on participants' responses to the two open-ended survey questions [205]. In this study, several phases had been followed as described by [36]; gaining familiarity with data, generating initial codes or labels, searching for themes or main ideas, reviewing themes or main ideas, defining and naming themes or main ideas, and producing the report. For this study, the coding was implemented by hand. The codes chosen aimed to identify the elements that participants noted as important to them in their responses. Out of 103 participants, only 35 answered the first open-ended question, 'What did you like best about the experience?' and 11 participants responded to the second open-ended question 'What did you dislike about the experience?'.

4.5.6.1 Demographic

In the second section of the online survey as shown in Appendix B, several questions on demographic data were given. The information gathered from this section indicate the programming novice learners and educators were included in the data collection process and the sample comprises a total of 102 participants.

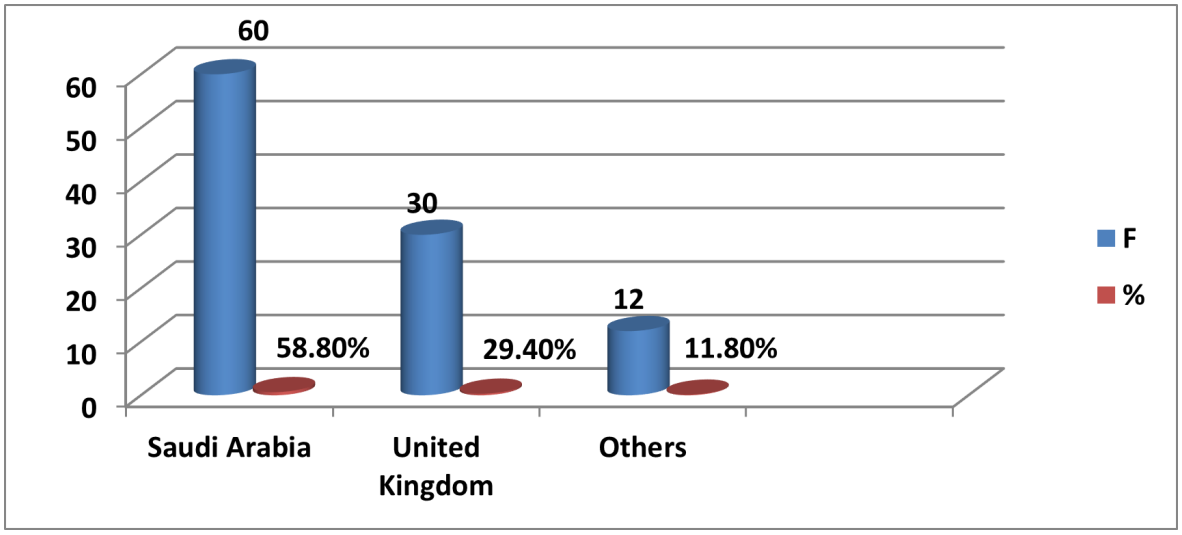

Figure 4.21: The percentage of location of the participants

In addition, the demographic data revealed that participants were from Saudi Arabia and the UK, As shown in Figure 4.21, the most popular country was Saudi Arabia, then the UK, with 12 participants from other countries. Moreover, participants ranged in age, the ages of the

study sample ranged from 19-50 years, the age group from 19 - 25 years was 33%, the age group from 26 - 35 years was 49%, the age group from 36 - 50 years was 18.6%.

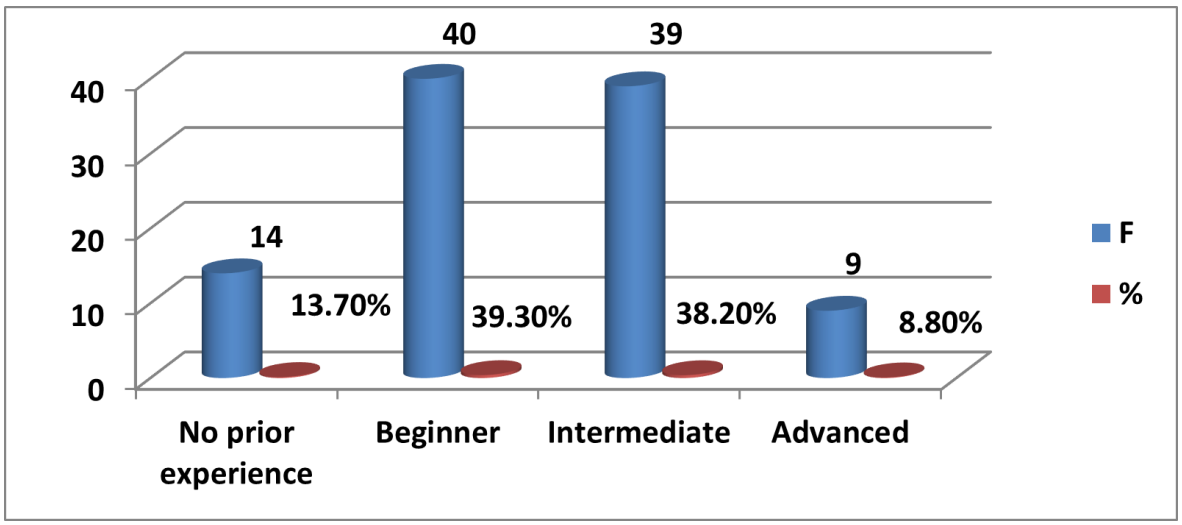

Figure 4.22: The percentage of coding experience

Additionally, as shown in Figure 4.22, the participant experience forms a neat bell-shaped curve, with representation of participants from a range of different coding experience levels.

4.5.7 Study findings

4.5.7.1 Quantitative findings

In this section, the quantitative findings from the participants' responses will be presented, as absolute values and percentages. In the third evaluation instrument presented in Section 4.4, there is a list of the system features. In this evaluation study, the online survey contains five Likert items about five supportive features provided in Python OCTS [165]. Below are discussed the responses to these items.

- **List of coding lessons** The number of respondents on (Strongly Disagree) is equal to 0 by 0 %, (Disagree) is equal to 1 by 1 %, (Neither Disagree nor Agree) is equal to 1 by 1 %, (Agree) is equal to 73 by 71.6 %, (Strongly Agree) is equal to 27 by 26.5 %.
- **Quiz** The number of respondents on (Strongly Disagree) is equal to 1 by 1 %, (Disagree) is equal to 1 by 1 %, (Neither Disagree nor Agree) is equal to 6 by 5.9 %, (Agree) is equal to 53 by 52 %, (Strongly Agree) is equal to 41 by 40.2 %.
- **Complete example** The number of respondents on (Strongly Disagree) is equal to 0 by 0 %, (Disagree) is equal to 1 by 1 %, (Neither Disagree nor Agree) is equal to 2 by 2 %, (Agree) is equal to 63 by 61.8 %, (Strongly Agree) is equal to 36 by 35.3 %.

- **Other materials** The number of respondents on (Strongly Disagree) is equal to 0 by 0 %, (Disagree) is equal to 1 by 1 %, (Neither Disagree nor Agree) is equal to 19 by 18.6 %, (Agree) is equal to 53 by 52 %, (Strongly Agree) is equal to 29 by 28.4 %.

- **Reflection notes** The number of respondents on (Strongly Disagree) is equal to (0) by (0 %), (Disagree) is equal to 1 by 1 %, (Neither Disagree nor Agree) is equal to 5 by 4.9 %, (Agree) is equal to 78 by 76.5 %, (Strongly Agree) is equal to 18 by 17.6 %.

In addition, in the third evaluation instrument presented in previous Section 4.4, there is a list of the system features. In this evaluation study, the online survey contains 11 Likert items about 11 technical supportive features provided in Python OCTS [165] or not provided. Below are discussed the responses to these items.

- **Syntax error messages** The number of respondents on (Strongly Disagree) is equal to 1 by 1 %, (Disagree) is equal to 0 by 0 %, (Neither Disagree nor Agree) is equal to 1 by 1 %, (Agree) is equal to 76 by 74.5 %, (Strongly Agree) is equal to 24 by 23.5 %.

- **Underlining syntax errors** The number of respondents on (Strongly Disagree) is equal to 1 by 1 %, (Disagree) is equal to 0 by 0 %, (Neither Disagree nor Agree) is equal to 3 by 2.9 %, (Agree) is equal to 68 by 66.7 %, (Strongly Agree) is equal to 30 by 29.4 %.

- **Syntax highlighting** The number of respondents on (Strongly Disagree) is equal to 1 by 1 %, (Disagree) is equal to 0 by 0 %, (Neither Disagree nor Agree) is equal to 2 by 2 %, (Agree) is equal to 47 by 46.1 %, (Strongly Agree) is equal to 52 by 51 %.

- **Visual map** The number of respondents on (Strongly Disagree) is equal to 0 by 0 %, (Disagree) is equal to 1 by 1 %, (Neither Disagree nor Agree) is equal to 35 by 34.3 %, (Agree) is equal to 46 by 45.1 %, (Strongly Agree) is equal to 20 by 19.6 %.

- **Code templates** The number of respondents on (Strongly Disagree) is equal to 0 by 0 %, (Disagree) is equal to 0 by 0 %, (Neither Disagree nor Agree) is equal to 29 by 28.4 %, (Agree) is equal to 51 by 50 %, (Strongly Agree) is equal to 22 by 21.6 %.

- **Identifying errors locations** The number of respondents on (Strongly Disagree) is equal to 1 by 1 %, (Disagree) is equal to 0 by 0 %, (Neither Disagree nor Agree) is equal to 2 by 2 %, (Agree) is equal to 79 by 77.5 %, (Strongly Agree) is equal to 20 by 19.5 %.

- **Customized hints** The number of respondents on (Strongly Disagree) is equal to 0 by 0 %, (Disagree) is equal to 0 by 0 %, (Neither Disagree nor Agree) is equal to 6 by 5.9 %, (Agree) is equal to 83 by 81.4 %, (Strongly Agree) is equal to 13 by 12.7 %.

- **Detailed error message** The number of respondents on (Strongly Disagree) is equal to 0 by 0 %, (Disagree) is equal to 0 by 0 %, (Neither Disagree nor Agree) is equal to 6 by 5.9 %, (Agree) is equal to 81 by 79.4 %, (Strongly Agree) is equal to 15 by 14.7 %.

- **Auto-completion** The number of respondents on (Strongly Disagree) is equal to (0) by 0 %, (Disagree) is equal to 0 by 0 %, (Neither Disagree nor Agree) is equal to 14 by 13.7 %, (Agree) is equal to 81 by 94 %, (Strongly Agree) is equal to 7 by 6.9 %.

- **Several programming languages** The number of respondents on (Strongly Disagree) is equal to 0 by 0 %, (Disagree) is equal to 1 by 1 %, (Neither Disagree nor Agree) is equal to 19 by 18.6 %, (Agree) is equal to 53 by 52 %, (Strongly Agree) is equal to 29 by 28.4 %.

- **Content organization** The number of respondents on (Strongly Disagree) is equal to 1 by 1 %, (Disagree) is equal to 1 by 1 %, (Neither Disagree nor Agree) is equal to 6 by 5.9 %, (Agree) is equal to 53 by 52 %, (Strongly Agree) is equal to 41 by 40.2 %.

4.5.7.2 Qualitative findings

After testing Python OCTS [165], two follow-up open-ended questions were asked as presented in Appendix B to capture participants' thoughts about other appropriate features to support novice learners that not provided in the third version of the evaluation instrument. The aim of these two questions is to improve the development of the evaluation instrument based on real users evaluations.

This section presents a qualitative data gathered from the two open-ended questions which involved some interesting features from participants. The participants were asked two open-ended questions to collect qualitative data. In this section the results from the participants view will be presented.

4.5.7.3 Identified themes from Q1

The first open-ended question was: **What did you like best about the experience?**

Figure 4.23: Responses to Question one

As shown in Figure 4.23, 35 responses had been received from the first open-ended question in the online survey. However , as presented in Table 4.10, only 25 responses were suggesting changes. In Table 4.10, 25 responses had been analysed based on themes and codes.

Theme 1: Personalization Personalizing the coding lessons or the coding tutorials was the main theme found from the participants' responses. For instance, one participant mentioned *"personalizing the exercises based on my performance"*. This response indicated the need personalizing the learning material in programming context. Another participant also suggested the same feature, he/she said *"personalization"*

Theme 2: Other languages Offering the coding lessons or the content of the online coding tutorial systems with different spoken languages was the main theme found from the participants' responses. For instance, one participant mentioned *"It would be nice tool if it offers different languages such as Arabic and Spanish"*. This response indicated the need to provide the learning materials with different languages the learning material in programming context.

Participants responses	Codes highlighting participant ideas: Significant ideas	Themes: Main idea
”Using the platform was so easy ”	Using the tools	Tool
”It was very interesting tool to learn programming”	All of the Features	All of the Features
” Personalization”	Personalizing the learning	Personalization
”It would be nice tool if it offers different languages such as Arabic and Spanish”	Offering other languages	Providing other languages
”Nice system with useful features”	All of the Features	All of the Features
” I think it would be nice if the system provides different languages such as Arabic”	Offering other languages	Providing other languages
”This online system helps a lot in understanding basics in Python”	List of coding lessons	Content organization
”This online tool looks helpful for beginners.”	All of the Features	All of the Features
” Personalizing the exercises based on my performance”	Personalizing the learning	Personalization
”It was interesting experience. ”	All of the Features	All of the Features
”Detailed explanations and offering Arabic language”	Offering other languages	Providing other languages
”Good platform”	All of the Features	All of the Features
”Helpful platform”	All of the Features	All of the Features
”Good experience”	All of the Features	All of the Features
”Great coding tool”	All of the Features	All of the Features
”Several useful features ”	All of the Features	All of the Features
”useful platform”	All of the Features	All of the Features
”Several useful features ”	All of the Features	All of the Features
”useful platform”	All of the Features	All of the Features
”I like the interactive shell”	Code editor	Code editor
”The organization of the content”	List of coding lessons	Content organization
”useful platform”	All of the Features	All of the Features
”Interesting code editor”	Code editor	Code editor
”Interesting tool”	All of the Features	All of the Features
”I like the organization of the coding lessons”	List of coding lessons	Content organization

Table 4.10: Participants responses to the first open-ended question in the evaluation study

4.5.7.4 Identified themes from Q2

The first open-ended question was: **What did you dislike about the experience?** 11 responses had been received from the first open-ended question in the online survey. However, As presented in Table 4.10, only 6 responses were suggesting suggestions. In Table 4.11, 6 responses had been analysed based on themes and codes.

Theme 1: Learning videos Offering learning videos that teach programming in the online coding tutorial systems was the main theme found from the participants' responses to the second open-ended question. For instance, one participant mentioned *"It was interesting experience, I think having more learning materials for example videos"*.

Participants responses	Codes highlighting participant ideas: Significant ideas	Themes: Main idea
"Few coding tutorials "	List of coding lessons	Extra content lessons
"Personalizing the coding tutorials based on users performance"	Personalizing the learning	Personalization
"I would like to see more Python topics"	List of coding lessons	Extra content lessons
"Needs for more advanced Python topics"	List of coding lessons	Extra content lessons
"the problem with the tool is has limited lessons"	List of coding lessons	Extra content lessons
"It was interesting experience, I think having more learning materials for example videos "	Adding videos	Learning videos

Table 4.11: Participants responses to the second open-ended question in the evaluation study

4.5.8 Log files

4.5.8.1 Number of visits

From the results in Figure 4.24 it is clear that the total number of visits of users is 456 times, noting that there are 3 users (ID:2, ID:5,ID:10) did not interact with the site. It is also noted that the number of times and the rate of using the site differed from one user to another. For instance, the number of visits to the site ranged from only one 0.2% for user (ID:6) and reached 64 times for user (ID:4) with a rate of 14%.

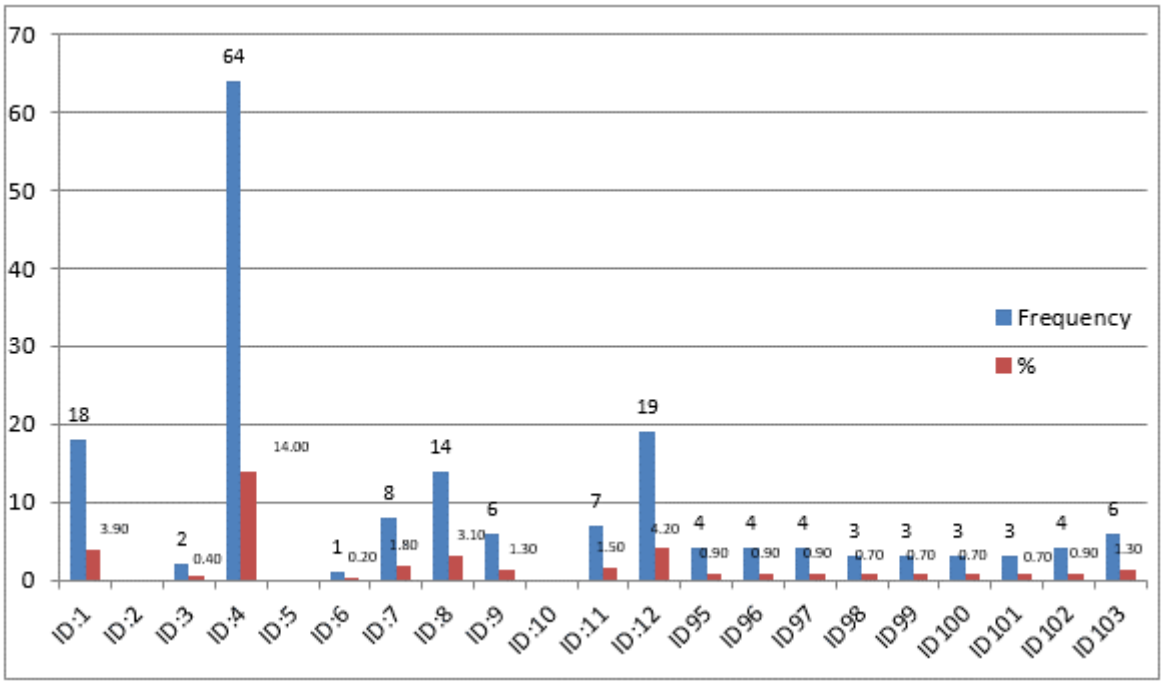

Figure 4.24: Frequency distribution for users

4.5.8.2 Time spent

The results shown in Figure 4.25 indicate that the time taken to visit the site differed from one user to another, and the least time spent (less than one minute) was for 219 visits of different users with a rate of 48.1%, and the largest time spent (more than 30 minutes) for a number of 4 visits of different users with a rate of 0.9%.

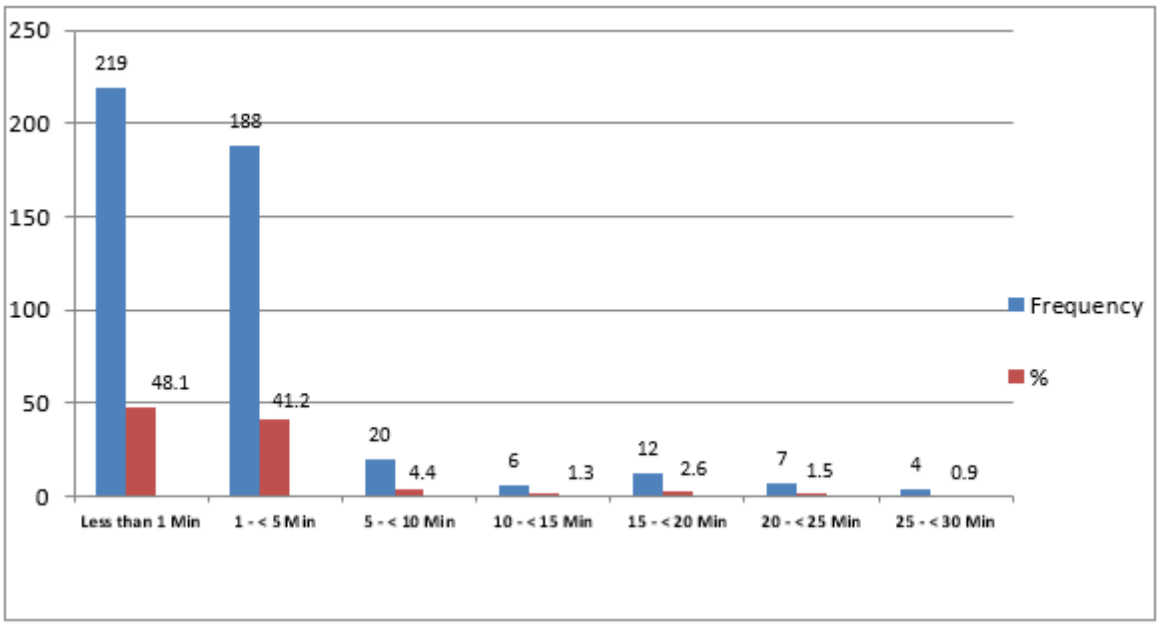

Figure 4.25: Frequency distribution for duration/min

4.5.8.3 Visited pages

The results shown in Figure 4.26 indicate that the visiting pages differed from one user to another, and the least visited page was (other matter) 3 timesat a rate of 0.7% and the largest visit was for (Lesson 1) 184 times, At a rate of 40.4%, followed by (Lesson 2) 110 times at a rate of 24.16%, then (Lesson 3) 52 times at a rate of 11.4%, then (Lesson 4) 28 times at a rate of 6.1%. The number of visits to (Quiz1) was 34 times, at a rate of 7.5%, then (Quiz2) 24 times, at a rate of 5.2%, then (Quiz3) 15 times, at a rate of 3.3%, and finally (Quiz4) 6 times by 1.4%.

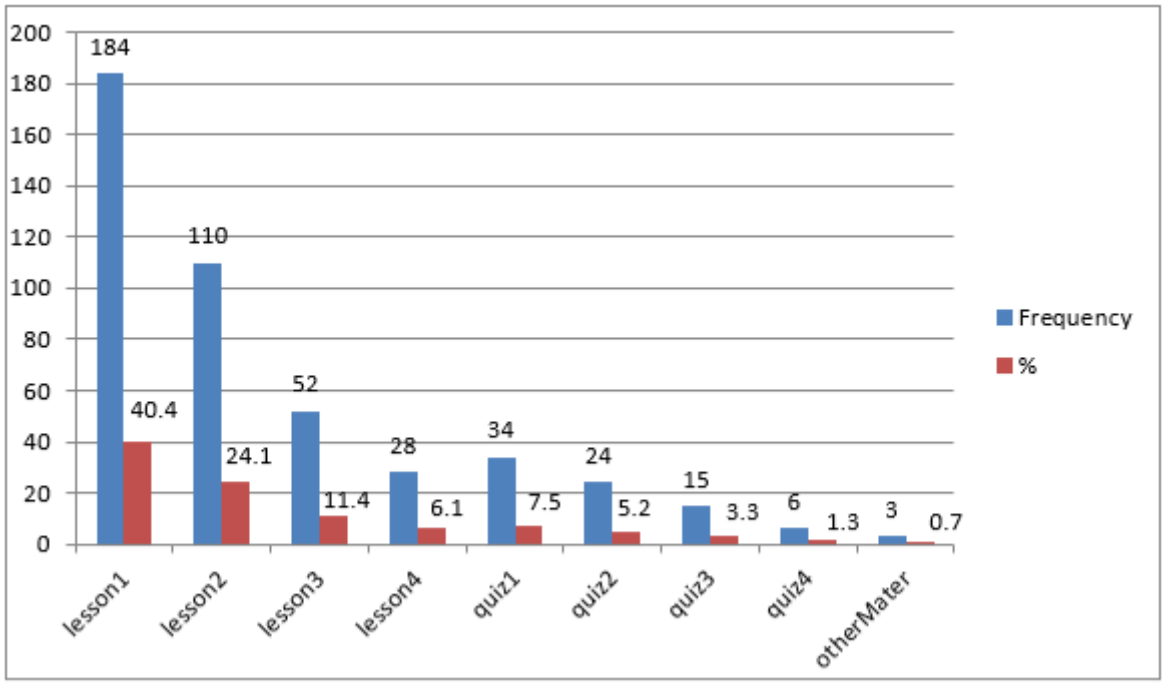

Figure 4.26: Frequency distribution for visit pages

4.5.8.4 Visiting quiz

It is clear from the results shown in Figure 4.27 that the total number of visits of the users that was not to view quiz was 430 times, at a rate of 68.4%, However the total number of visits of the users that was to view quiz was only 26 times, at a rate of 31.6%.

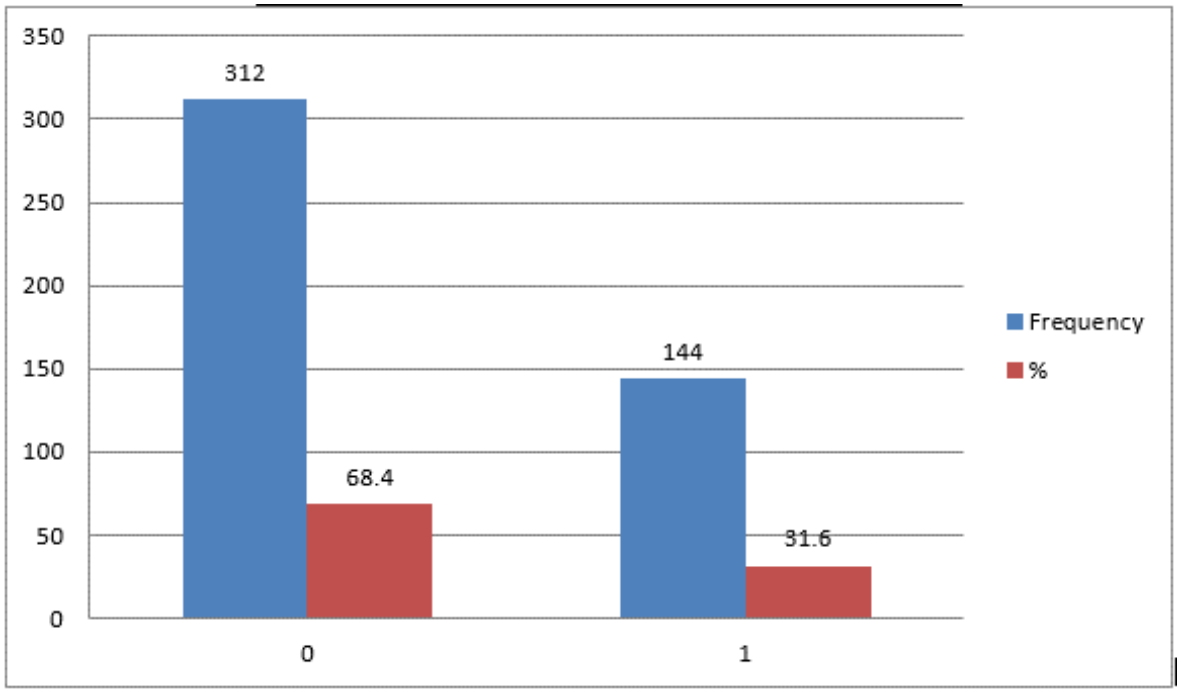

Figure 4.27: Frequency distribution for visit quiz

4.5.8.5 Viewing solution

It is clear from the results shown in Figure 4.28 that the total number of visits of the users that was not to solutions was 460 times of visits, at a rate of 94.3%. However, the total number of visits of the users that was to 26 times, at a rate of 5.7%.

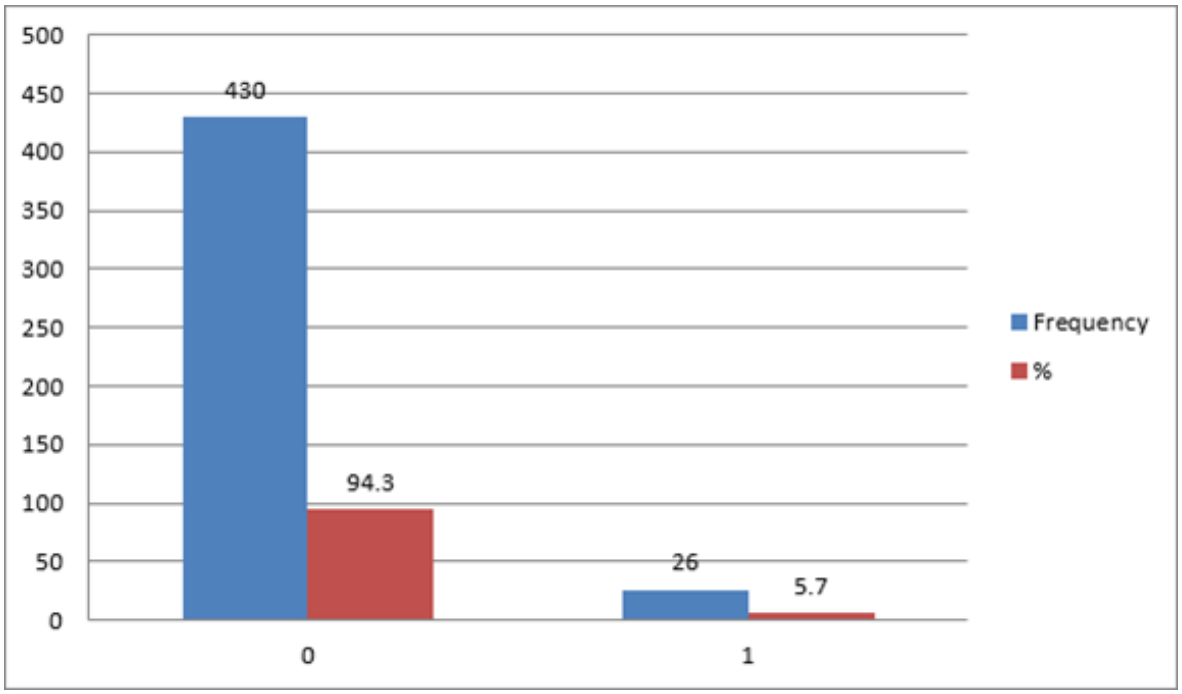

Figure 4.28: Frequency distribution for view solution

4.5.8.6 Viewing other materials

The results indicate that from the total visits of the users that was not to view other materials was 447 times, at a rate of 98%, However the total number of visits of the users to view other materials was only 9 times, at a rate of 2%.

This section fits in the design and the implementation phase of the design-based research approach [125]. It contributed to the main aim of this research work by developing an online coding tutorial system prototype that contains most of the features in the third version of the evaluation instrument in order to give users the ability to evaluate the evaluation instrument and refine it. After the responses from the online survey that were analysed. Findings showed that the vast majority of users were satisfied with the features presented in the system prototype. In this section, we discuss all the identified correlations between analysed data collected from log file that show interactions of users with our online coding tutorial systems discussed in this Section 4.5.8 and between analysed data collected from online instrument that distributed among users to measure their satisfaction toward our system prototype that are discussed in Section 4.5.5.2 . In the first phase of the current study, users and their interaction with the system were observed in terms of (Frequency distribution for users, Frequency distribution for duration/min, Frequency distribution for visit pages, Frequency distribution for use code editor, Frequency distribution for visit quiz, Frequency distribution for view solution, Frequency distribution for view other materials). In the second phase of the current study, the users tested the system and then measured their satisfaction with the system through a questionnaire. In this section, the researcher analyzes the users' responses and searches for the extent of similarity or difference between the results of the first stage (Log data) and the results of the second stage (Questionnaire). The following was noted:

- 60 % of users took less than 1 minute, 20 % 1-2 minutes, 80 % less than 2 minutes, 90 % less than 5 minutes. The researcher believes that this period is short to deal with the program and see all its content.
- **Providing quiz** 47 % of users, visit quiz, and we note that 92 % of their responses were agree to strongly agree about (Providing a quiz after each lesson allows me to test my understanding of basic concepts of Python programming language). This means that there is an actual benefit, but there may be an exaggeration (overstatement).
- **Providing solution** 4 % of users show solution, However, 97 % stated that Offering complete example programs and worked solutions helps me to understand basic concepts of Python programming language. This means that there is an actual benefit, but there may be an exaggeration (overstatement).
- **View other materials** 4 % of users view other materials, However, 80 % stated that Offering extra learning materials as other learning sources helps me to understand

basic Python concepts. This means that there is an actual benefit, but there may be an exaggeration (overstatement).

4.5.9 The changes in version three of the instrument

In this section, the updates on the version three of the instrument presented in Section 4.4.6 will be discussed. Below the updates in both categories discussed: Findings from the third fact finding study enabled the identification of new features in the instrument. These new features are as follows:

- Learning videos
- Offering several languages

Findings from the third fact finding study enabled the identification of new feature in the instrument. The new feature is as follows:

- Personalization

4.5.10 Fourth version of the instrument

As shown in Figure 4.29, the fourth version of the evaluation instrument for online coding tutorial systems is proposed. This evaluation study helped to develop and improve the evaluation instrument by adding new items.

Components	Items
Syntax of programming languages	Syntax error messages
	Underlining syntax errors
	Syntax highlighting
Structure of code	Visual map
Understanding basic concepts	Lesson content
	Worked solutions
	Quizzes
	Extra content lessons
	Content organization
Debugging	Detailed error messages
	Identifying error locations
	Customized hints
Dividing functionality into procedures	Auto-completion
Transferring algorithm to concrete implementation	Syntax-directed editor
Improving learning experiences	Reflection notes
	Personalisation
Offering several programming languages	Several programming languages
Supporting programming learning resources	Learning Videos
	Reference materials
Understanding lesson content	Offering several languages

Figure 4.29: Th evaluation instrument for online coding tutorial systems version four based on systems prototype evaluation (design cycle four, changes in red)

4.6 Chapter Summary

This chapter presents the process of developing the evaluation instrument for online coding tutorial systems in this thesis by using design-based methodology. The four design cycles of designing the instrument have been presented and discussed in detail in this chapter.

Chapter 5

Instrument Validation

5.1 Chapter Overview

This chapter presents the study conducted to validate the proposed instrument for evaluating online coding tutorial systems that was developed and presented in Chapter 4. This chapter is laid out as follows: Section 5.2 presents the research question that will be answered in this validation study, followed by Section 5.3 discusses the method used to validate the instrument, Section 5.4 discusses the data analysis techniques used in this study, Section 5.5 presents and discusses the findings of this validation study, Section 5.6 presents and discusses the proposed evaluation instrument in this thesis that could be used by programming educators, and finally Section 5.7 concludes this validation study.

5.2 Research Question 5

This study addresses the fifth research question in this thesis: RQ5: **To what extent is it applicable to use the proposed instrument for OCTSs as a tool to evaluate any online coding tutorial systems? This study contributes to** the knowledge area by checking the validity of each component and each item presented in the evaluation instrument of online coding tutorial systems in Chapter 4. The validation is based on a consultation with a group of selected experts in the programming education field. The validation process, which uses the fuzzy Delphi method [99] is reported in detail in this chapter.

5.3 Study Method

In this study, the fuzzy Delphi method (FDM) was used to validate the online coding tutorial systems assessment instrument, which was developed in Chapter 4. This method is a more

sophisticated form of the Delphi approach [122]. It uses fuzzy logic to handle uncertainties and subjective judgements inherent in the consensus of the experts [99] [181]. This method is most useful in research design that seeks to get experts opinions on some issues or on specific products, such as the evaluation of educational tools, where the process of reaching consensus must be systematic, progressive process [100]. This is in accordance with several other studies that have used the fuzzy Delphi method in various analyses. For instance, in Morales et al.[141]'s study, a technological tool called "MUETBot" has been used to enhance the reading skills of the Malaysian University English Test. Its procedures involved the construction of a checklist of questionnaires, invitations to leaders of professional panels, data collection and analysis, and cycles of feedback and improvement till important consensus was achieved among the experts concerning certain functions. The analysis of this study also demonstrates the fuzzy Delphi method's ability to map extensive consensus of knowledge by the FDM with the core attributes of educational tools, strengthening the appropriateness of applying the tool to this research [141].

Further, the fuzzy Delphi method has been successfully applied; for example, Mostafa et al. [142] conducted a study to assess the concession period of BOT projects. With regards to uncertainties, their research summarised the opinions of experts on different uncertain inputs to provide a better assessment of the concession period. The case study performed proved the FDM's potential to operate under conditions of high uncertainty and complexity, which proved the suitability of the online coding tutorial systems assessment instrument. In this thesis, the fuzzy Delphi method was used to ensure that the evaluation instrument developed is not only academically valid and reliable but can also be used in various contexts of education. The incorporation of fuzzy logic to the Delphi process of the FDM allowed the tool to identify and validate components and items as results of the process portrayed the total body of knowledge of the panel. Since the method was repetitive, it was possible to receive constant feedback, which was crucial in creating a reliable instrument for evaluating online coding tutorial systems.

5.3.1 Procedure

The procedure that has been used in this study has been discussed earlier in Chapter 3, cf. Figure 3.2. The process entails selecting and consulting with experts in the programming education domain, getting their input on the instrument's components and items, translating their views into numerical values, and presenting the instrument after identifying the main problems, components/items that require evaluation, or components/items that have been removed. Lastly, using verbal and linguistic variables, such as "extremely agree," "strongly agree," "agree," "moderately agree," "disagree," "strongly disagree," and "extremely disagree," that are translated into imprecise numerical scales is typically required when the

same experts evaluate the instrument's final version [227] [99] [141].

5.3.1.1 Expert selection

Finding the experts in the programming education field is the first step in the instrument's validation process. Several factors can be considered while estimating the number of experts. Ten to fifty experts are recommended by the Delphi approach, according to [32]. Moreover, according to [2], due to their uniformity and sufficiency, only ten experts in the field of teaching programming were selected for this investigation. According to [32], after five to ten years of employment, instructors can be considered experts. Therefore, in this study, each of the chosen experts has taught computer science and programming courses for ten years and more than ten years.

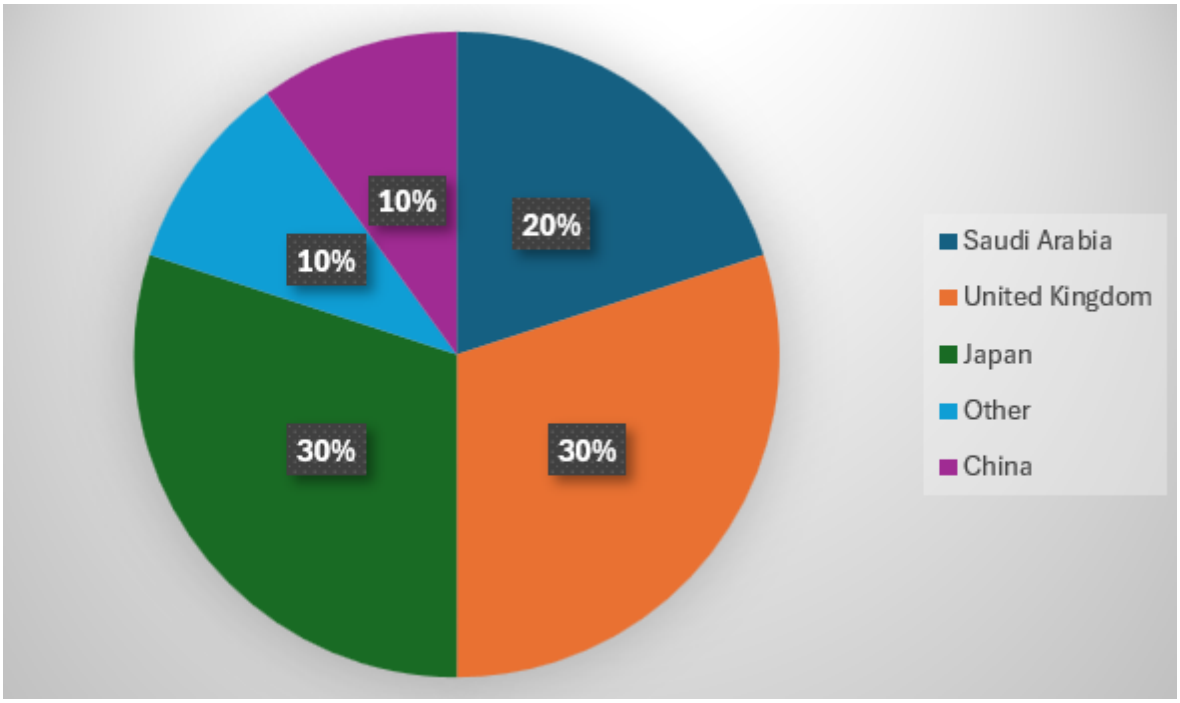

Figure 5.1: The nationalities of the experts involved in the validation study

The 10 experts were university professors from various countries, including Saudi Arabia, the UK, Japan, China, and others, as shown in Figure 5.1. The 48th IEEE International Conference on Computers, Software, and Applications (COMPSAC 2024) (`https://ieeecompsac.computer.org/2024/`), which took place in Osaka, Japan, was where the participants were chosen. The author of this thesis was present and took part in the conference. During the conference break, these programming educators were invited to contribute individually by providing the resources and noting their information. Two days following the conference, emails were sent to fourteen educators; only ten of them replied,

and four did not reply at all. As presented in Table 5.1, two Saudi educators, three educators from Britain, three educators from Japan, one educator from China, and one educator of another nationality—an ”Indian”—all responded.

One senior lecturer from a Chinese university, three senior lecturers from Japanese universities, two lecturers from two Saudi universities specialising in computing education research, three senior lecturers from British universities specialising in programming learning systems research, and one senior lecturer from an Indian university comprised this expert panel. They were able to commit themselves to this study and were appointed freely.

Code	Gender	Country	Teaching experience	Programming languages thought
Expert-1	Male	Saudi Arabia	20 years	Scratch, HTML, Python, PHP, SQL, JavaScript, C++, C
Expert-2	Male	India	18 years	Scratch, HTML, Python, PHP, SQL
Expert-3	Female	Japan	18 years	Scratch, HTML/CSS, SQL, VB, JavaScript, PHP
Expert-4	Male	Saudi Arabia	16 years	HTML, Scratch, JavaScript, SQL, VB, JavaScript, PHP
Expert-5	Male	United Kingdom	15 years	Java, Arduino, Web (PHP), Python
Expert-6	Male	United Kingdom	14 years	HTML, JavaScript, PHP, Python
Expert-7	Male	Japan	17 years	Blocky, Scratch, Python, HTML/CSS, SQL, VB, JavaScript, PHP
Expert-8	Male	Japan	14 years	Scratch, HTML/CSS, SQL, VB, JavaScript, PHP
Expert-9	Female	China	12 years	Scratch, Python, C++, Java,
Expert-10	Male	United Kingdom	10 years	HTML, JavaScript, PHP, Python

Table 5.1: Details over the participating experts. Expert’s code reflects the order of their programming teaching experiences, where Expert-1 has more number of years of experience, and Expert-10 has the less number of years

5.3.1.2 Development of the instrument' statements

Based on the findings of studies reported in Chapter 4, the mentor panel, which consists of this thesis author and the two academic supervisors, created an initial list of twenty clear statements from the twenty items as indicated in Figure 5.2 in order to give the ten experts that were selected an appropriate form of the instrument to be validated.

Components	Items No	Items
Syntax of programming languages	1	Providing a code editor that prints the result as a message that indicates errors that the code might have.
	2	Providing a code editor that underlines syntax errors to indicate errors that the code might have.
	3	Providing a code editor that provides syntax highlighting.
Structure of code	4	Providing a visual map feature might help learners understand the execution process of the code
Understanding basic concepts	5	Providing a list of coding exercises that cover most basic and complex coding concepts.
	6	Offering practical examples that demonstrate how programming concepts and techniques can be applied in real world scenarios.
	7	Offering quizzes to test novice learners' comprehension and solidify their understanding of programming concepts.
	8	Providing enough coding exercises that cover most of the basic and complex coding concepts.
	9	Providing an organised list of coding exercises that cover most of the basic and complex coding concepts.
Debugging	10	Providing a code editor that prints the result as a detailed message that indicates errors that the code might have.
	11	Identifying errors' locations in the output shell to debug the code.
	12	Providing an interactive shell that supports the "Hints feature" to avoid careless mistakes.
Dividing functionality into procedures	13	Providing a code editor that predictively completes whatever I want to type.
Transferring algorithm to concrete implementation	14	Providing code templates in the script shell.
Improving learning experiences	15	Providing a reflection note box serves as a personal space where learners can document their thoughts and reflections as they progress through their learning journey.
	16	Offering tailored learning materials, recommendations, and feedback based on individual needs and preferences.
Offering several programming languages	17	Offering several options for programming languages.
Supporting programming learning resources	18	Offering programming learning videos.
	19	Offering extra material other than the course tutorials, such as textbooks.
Understanding lesson content	20	Offering several languages.

Figure 5.2: The preliminary evaluation instrument that was created by the mentor panel

5.3.1.3 The validation process of components and items using the fuzzy Delphi method

The experts were invited to participate in a Zoom meeting for an interview, following emails that were sent to the expert panel attaching the instrument shown in Figure 5.2. In order to give the ten experts reasonable time to comprehend the study's context and a chance to come up with ideas for improving the instrument, the developed instrument's components and items were sent to them via email one week prior to the discussion via a Zoom meeting. The purpose of this semi-structured interview was to gain a deeper comprehension of the objectives and real goals of local context-based programming education [132]. In addition, to conduct the focus discussion group with the experts to get in the validation process.

Furthermore, as demonstrated previously in Chapter 3 (Figure 3.2), the validation process comprises several stages: first, agreement on the primary instrument components; second, item arrangement based on expert opinions and consensus; third, experts assess the components and items independently; and, last, data collection and analysis utilising the fuzzy Delphi technique. An explanation of each step in the process is provided below:

1. **First step: expert consensus regarding the main components**

   In the Zoom meeting, the focus discussion group aims to keep the characteristics of the fuzzy Delphi method, such as the research time frame compared to the traditional Delphi method, while also addressing the shortcoming of the iterative process seen while utilising the Delphi method [181]. Throughout the focus discussion group in the Zoom meeting, each expert gave their views and claims.

   In the first step, each expert was given a refresher on the details of the programming learning challenges (the instrument's components). In the meeting, two worksheets in a Microsoft Excel document were utilised. For the debate, the experts were divided into two groups of five people each and given worksheets with numbers one and two. Moreover, there were two phases involved in the instrument's component verification procedure; in **the first phase**, the components were assessed and verified by the experts in each group using the definitions that were provided before the meeting.

   All the expert opinions were suitably updated in the Microsoft Excel document, and the facilitator then moved the consensus from each group to the second worksheet, adhering to the group column. A consensus was reached to assess and validate the components in accordance with the recommendations made by each group throughout **the second stage** of the instrument's component verification process. The second column was completed with the final consensus. As shown in Figure 5.3, the experts agreed to take out five of the main components that present common programming learning challenges in the proposed instrument.

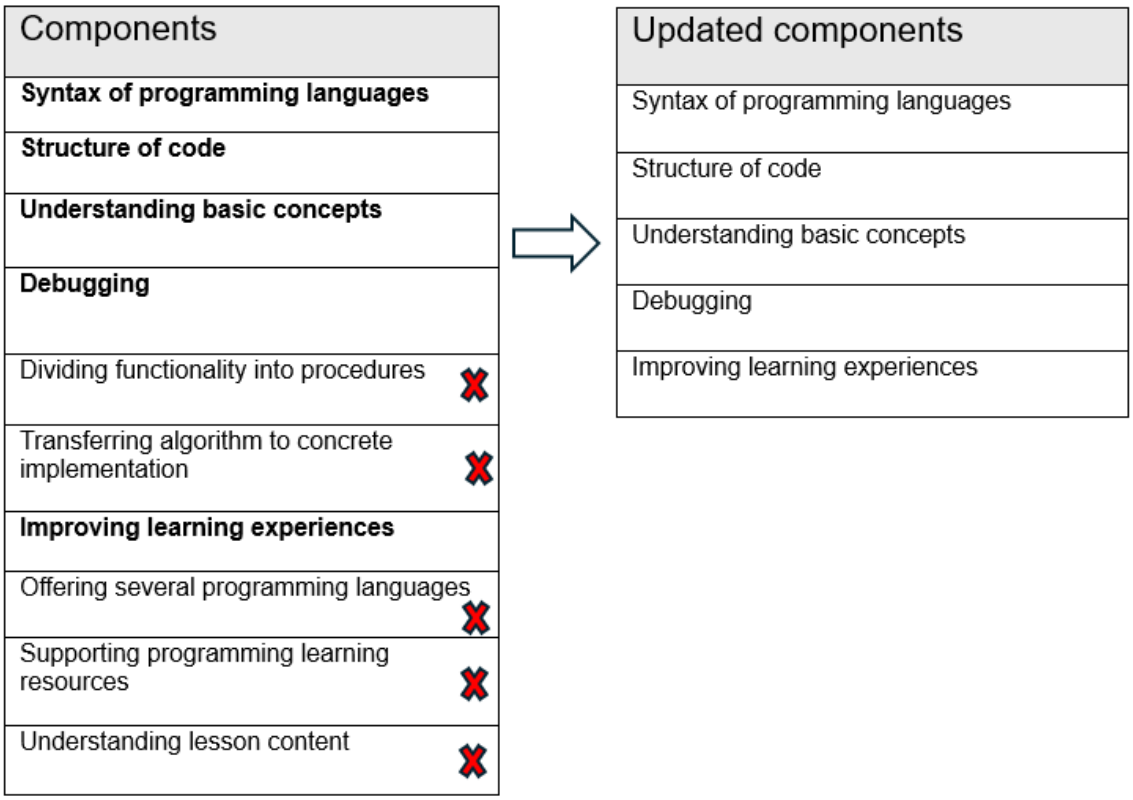

Figure 5.3: First results of the focus discussion group process (the red X indicates the components agreed to be deleted by the experts)

2. **Second step: expert consensus on the arrangement of items**

   After the expert panel validated the instrument's components, the experts assessed the list of items. As shown in Figure 5.4, four items were agreed to be deleted because they shared the same meaning as the other items in the instrument. Then, as shown in Figure 5.5, the experts confirmed to put the items next to the appropriate components. The item that the experts had confirmed was shown next to the component list. As part of the validation process, experts discussed how to improve the suggested items. The language structure was refined to be clear and consistent with the abilities' definition and the study's setting; it also avoided redundant, incorrect, or unnecessary items and proposed new ones where necessary to solve the problem's definition.

Components	Items
Syntax of programming languages	Providing a code editor that prints the result as a message that indicates errors that the code might have.
	Providing a code editor that underlines syntax errors to indicate errors that the code might have. ✘
	Providing a code editor that provides syntax highlighting.
Structure of code	Providing a visual map feature might help learners understand the execution process of the code
Understanding basic concepts	Providing a list of coding exercises that cover most basic and complex coding concepts.
	Offering practical examples that demonstrate how programming concepts and techniques can be applied in real world scenarios.
	Offering quizzes to test novice learners' comprehension and solidify their understanding of programming concepts.
	Providing enough coding exercises that cover most of the basic and complex coding concepts. ✘
	Providing an organised list of coding exercises that cover most of the basic and complex coding concepts. ✘
Debugging	Providing a code editor that prints the result as a detailed message that indicates errors that the code might have. ✘
	Identifying errors' locations in the output shell to debug the code.
	Providing an interactive shell that supports the "Hints feature" to avoid careless mistakes.
Dividing functionality into procedures ✘	Providing a code editor that predictively completes whatever I want to type.
Transferring algorithm to concrete implementation ✘	Providing code templates in the script shell.
Improving learning experiences	Providing a reflection note box serves as a personal space where learners can document their thoughts and reflections as they progress through their learning journey.
	Offering tailored learning materials, recommendations, and feedback based on individual needs and preferences.
Offering several programming languages ✘	Offering several options for programming languages.
Supporting programming learning resources	Offering programming learning videos.
✘	Offering extra material other than the course tutorials, such as textbooks.
Understanding lesson content ✘	Offering several languages.

Figure 5.4: Second results of the focus discussion group process (the red X indicates the components and items were deleted by the experts)

Components	Items
Syntax of programming languages	The code editor that the system offers prints the outcome as a message indicating any potential mistakes in the code.
	The system provides a code editor that provides syntax highlighting.
Structure of code	The system has a visual map function that could aid students in comprehending how the code is executed.
Understanding basic concepts	The system provides a list of code exercises covering the majority of fundamental and advanced coding principles.
	The system provides practical examples that demonstrate how programming concepts and techniques can be applied in real-world scenarios.
	The system provides quizzes to test novice learners' comprehension and solidify their understanding of programming concepts.
Debugging	The system provides an output shell that shows errors' locations to debug the code.
	The system provides a code editor that predictively completes whatever I want to type.
Improving learning experiences	The system provides a reflection note box that serves as a personal space where learners can document their thoughts and reflections as they progress through their learning journey.
	The system provides tailored learning materials, recommendations, and feedback based on individual needs and preferences.
	The system offers several options for programming languages.
	The system offers programming learning videos.
	The system offers extra material other than the course tutorials, such as textbooks.
	The system offers several languages.
	The system provides an interactive shell that supports the "Hints feature" to avoid careless mistakes.

Figure 5.5: The updated instrument after the experts advised to put the items next to the appropriate components

3. **Third step: expert consensus on the arrangement of items according to priority** After verifying and updating the components and the items of the instrument, the experts evaluated and validated the suggested items for every component as shown in Figure 5.6. the items for each component had been arranged according to the priority based on the experts experiences. For instance, the items come under the understanding basic concepts are more important than the items come under syntax of programming languages.

Components	Items no	Items
Understanding basic concepts	1	The system provides a list of code exercises covering the majority of fundamental and advanced coding principles.
	2	The system provides practical examples that demonstrate how programming concepts and techniques can be applied in real-world scenarios.
	3	The system provides quizzes to test novice learners' comprehension and solidify their understanding of programming concepts.
Structure of code	4	The system has a visual map function that could aid students in comprehending how the code is executed.
Syntax of programming languages	5	The code editor that the system offers prints the outcome as a message indicating any potential mistakes in the code.
	6	The system provides a code editor that provides syntax highlighting.
Debugging	7	The system provides an output shell that shows errors' locations to debug the code.
	8	The system provides a code editor that predictively completes whatever I want to type.
Improving learning experiences	9	The system provides a reflection note box that serves as a personal space where learners can document their thoughts and reflections as they progress through their learning journey.
	10	The system provides tailored learning materials, recommendations, and feedback based on individual needs and preferences.
	11	The system offers several options for programming languages.
	12	The system offers programming learning videos.
	13	The system offers extra material other than the course tutorials, such as textbooks.
	14	The system offers several languages.
	15	The system provides an interactive shell that supports the "Hints feature" to avoid careless mistakes.

Figure 5.6: Third results of the focus discussion group process (the updated arrangement of the instrument's items according to the priority based on the experts opinions)

4. **Fourth step: expert evaluation individually**

   In the last step, the components and items of the instrument were moved into Microsoft Forms for the purpose of obtaining expert evaluations from each of them individually, as presented in Appendix C. Data analysis was made easier, and data transmission to Microsoft Excel was made possible by the use of Microsoft Forms. The Microsoft Form URL was then used to send the questionnaire to experts via email after the meeting, as presented in Appendix C. The experts then completed each questionnaire separately, selecting an option on a 7-point Likert scale that ranges from strongly disagree to strongly agree. This allowed them to assess the instrument items. All of the experts completed the questionnaire using Microsoft Forms, and their responses were immediately saved in a Microsoft Excel document to be analysed easily.

5.4 Data Analysis Techniques

The data collected from the expert evaluation step, as discussed in the previous section, was analysed using specific data analysis techniques that were selected to interrupt the experts responses to the Likert scale survey.

5.4.1 Converting Likert scale to fuzzy scale

There are two mains in FDM which is Triangular Fuzzy Number and Defuzzification Process [229]. Triangular Fuzzy Number is m is made up of the value of the m1, m2, and m3 where m1 represents the value of the minimum (smallest value), representing the most reasonable value m2 (most plausible value) and m3 is referring to the maximum value (but there is value) [168].

All of these three values is in the range of 0 to 1 and it coincided with fuzzy numbers [168]. As shown in Table 5.2, the corresponding fuzzy scale for each Likert scale are presented. In addition, by using Microsoft Excel's VLOOKUP tool, the Likert scale data collected from the experts were transformed into fuzzy numbers so that the fuzzy Delphi method could be used to study them [121].

Linguistic variables	Likert scale	Fuzzy scale (m1)	(m2)	(m3)
Extremely agree	7	0.9	1	1
Strongly agree	6	0.7	0.9	1
Agree	5	0.5	0.7	0.9
Moderately agree	4	0.3	0.5	0.7
Disagree	3	0.1	0.3	0.5
Strongly disagree	2	0	0.1	0.3
Extremely disagree	1	0	0	0.1

Table 5.2: Linguistic Variables for 7 Point Scale

5.4.1.1 Data analysis using the fuzzy Delphi method

As mentioned in the previous section, the triangular fuzzy number and the defuzzification procedure are the primary factors taken into account when employing the fuzzy Delphi technique [229] [168]. These procedures were used to check if each item was accepted or rejected according to the opinion of experts. Firstly, the triangular fuzzy number's lowest (m1), reasonable (m2), and maximum (m3) for each expert's responses. Secondary, the defuzzification process was used, and the instrument's item acceptance was calculated by using the threshold (d) and the percentage of consensus, and creating a fuzzy score (A) [229].

1. **Triangular fuzzy number: average of fuzzy number (m1, m2, m3)**

   A triangle graph is displayed against triangle values in Formula1: $m = \frac{\Sigma_{i=1}^{n} m_i}{n}$. Every value (m1, m2, m3) falls between 0 and 1, which is known as the fuzzy number (0,1). A fuzzy number's average value was calculated by using the formula 1 where the number of experts is denoted by (n).

2. **Triangular fuzzy number: threshold (d) value** To determine the degree of expert consensus for each questionnaire item, the threshold value (d) was computed [206]. The threshold value (d) for the fuzzy numbers m (m1, m2, m3) and n = (n1, n2, n3) can be found using the following Formula 2, as seen in Figure 5.7, based on the fuzzy numbering (0,1).

   According to Cheng et al. [43], all experts are deemed to have reached an agreement if the difference between the mean value and the expert evaluation data is less than or equal to the threshold value (d) $\leqslant$ 0.2. The analysis of the data according to the threshold value (d) is presented in Table 5.3.

$$d(\tilde{m},\tilde{n}) = \sqrt{\frac{1}{3}[(m_1 - n_1)^2 + (m_2 - n_2)^2 + (m_3 - n_3)^2]}$$

Figure 5.7: Formula 2

The expert consensus's percentage value must equal or exceed 75% in order for each item to satisfy the acceptance criteria set forth by the experts. An additional round is held against the non-consenting expert, or the item must be deleted if the expert's consensus percentage is less than 75%.

Threshold (d) value	Descriptions	Interpretation
d $\leqslant$ 0.2	The threshold (d) value is less than or equal to 0.2	Accepted
d $\geqslant$ 0.2	The threshold (d) value is greater than 0.2	Rejected OR conduct the second cycle, which involved only experts who disagreed.

Table 5.3: Interpretation of the data based on the threshold value (D)

5.4.1.2 Defuzzification: the fuzzy score

The fuzzy score (A) that is obtained through the defuzzification technique indicates whether or not an item is acceptable depending on the consensus of experts. The element is deemed acceptable when the fuzzy score (A) is at least as high as the median (α - cut) value of 0.5 [44].

Furthermore, the fuzzy score value (A) can be used to determine the ranking and order of the instrument items. Since the study focuses on the problem-solving technique in programming, the experts' debate defined the arrangement of each component's contents. Results may deviate from the programming approach to problem solving if the defuzzification process establishes the element's priority. The expert will reevaluate the importance and order of the items if any analysis results in the rejection of any part of the instrument.

5.5 Study Findings

The focus group discussion with experts via the Zoom meeting was held as part of this study to validate the components and items of the instrument proposed in this thesis to evaluate

online coding tutorial systems. Then, as presented in Appendix C, every item in the instrument that the experts verified was approved. Components and items revealed average scores between 6 and 7 on a 7-point Likert scale for every item, indicating strong and extreme agreement. Then, the Likert scale scores were converted to fuzzy scales for analysis. Furthermore, as shown in Table 5.4 the validation study's findings demonstrated that, according to the consensus of ten experts, each instrument's item has been verified and accepted by the experts, which was threshold (d) ≤ 0.2. Regarding the second prerequisite, for the validated 15 items, a 91.7% to 100% percent consensus was reached. To assess the acceptance of the evaluation instrument items for online coding tutorial systems, the third need was to receive a fuzzy score (A). The instrument item is allowed if the fuzzy score (A) is more than 0.5. For each item that was assessed, a fuzzy score (A) between 0.772 and 0.900 was created as shown in Table 5.4. Programming educators can use the instrument to assess online coding tutorial systems because they validated that each item of the evaluation instrument was approved.

No. Item	The threshold value, d	Percentage of Consensus Expert Group,%	m1	m2	m3	Score Fuzzy (A)	The Consensus of Experts
1	0.174	100.0%	0.633	0.800	0.933	0.789	ACCEPT
2	0.220	91.67%	0.767	0.892	0.942	0.867	ACCEPT
3	0.174	100.0%	0.767	0.900	0.967	0.878	ACCEPT
4	0.147	100.0%	0.800	0.925	0.975	0.900	ACCEPT
5	0.191	91.67%	0.617	0.783	0.917	0.772	ACCEPT
6	0.191	91.67%	0.617	0.783	0.917	0.772	ACCEPT
7	0.196	100.0%	0.700	0.850	0.950	0.833	ACCEPT
8	0.206	91.67%	0.750	0.883	0.950	0.861	ACCEPT
9	0.211	91.67%	0.650	0.808	0.925	0.794	ACCEPT
10	0.220	91.67%	0.683	0.833	0.933	0.817	ACCEPT
11	0.219	91.67%	0.717	0.858	0.942	0.839	ACCEPT
12	0.234	91.7%	0.633	0.792	0.908	0.778	ACCEPT
13	0.206	91.67%	0.750	0.883	0.950	0.861	ACCEPT
14	0.220	91.67%	0.683	0.833	0.933	0.817	ACCEPT
15	0.211	91.67%	0.650	0.808	0.925	0.794	ACCEPT

Table 5.4: Result of a Consensus of the Experts

5.6 The Guidelines to Use the Instrument

Firstly, the targeted audience of the proposed evaluation instrument created within this thesis is professional programming educators. These individuals maintain miniature oversight of the several processes that novices go through while learning how to code [45]. Their experience in the ability to analyse different programming ideas and algorithms into bite-sized portions that are easy to comprehend is relevant since novices face the abstract kind of learning [45]. Professional programming educators can easily understand the difficulties of the programmer novices and can adjust their teaching methodology to fit these difficulties [140]. Through the proposed evaluation instrument in this thesis, educators can measure the extent to which the online coding tutorial systems are helpful to novices.

As shown in Figure 5.8, the evaluation instrument presented in this thesis uses a checklist format to be used by programming educators, and the reason for using this format is because the checklist format has many advantages [202], for instance, the minimal likelihood level of leaving out any aspect of evaluating such systems while using such a format [204] [202]. Moreover, checklists are also flexible in nature, it means that the educators implementing the checklists can apply them to any educational setting and any learning situation [12]. Therefore, it is convenient for professional programming educators to use the instrument as a checklist, as shown in Figure 5.8.

5.7 Chapter Summary

This chapter presents a study that aims to validate the components and items of the proposed instrument in Chapter 4 as an evaluation instrument for online coding tutorial systems by obtaining expert consensus using the fuzzy Delphi method. The analysis's findings demonstrated that, according to expert consensus, all parts and components were approved. Therefore, these aspects and components may be employed in programming instruction and learning as well as serving as an assessment tool for use by professional programming educators.

Evaluation Instrument to Programming Educators			
The aim of this instrument is to assess online coding tutorial systems to gain a better understanding of whether they support novices to overcome programming' difficulties currently faced by them.			
Educator' name:			
School:			
Province:			
Programming language teach:			
Number of years teaching IT	0-5 years	6-10 years	More than 10 years

Components	Items no	Items	Yes	No
Understanding basic concepts	1	**Does the system provide a list of code exercises covering most fundamental and advanced coding principles?**		
	2	**Does the system provide practical examples that demonstrate how programming concepts and techniques can be applied in real-world scenarios?**		
	3	**Does the system provide quizzes to test novice learners' comprehension and solidify their understanding of programming concepts?**		
Structure of code	4	**Does the system have a visual map function that could aid learners in comprehending how the code is executed?**		
Syntax of programming languages	5	**Does the system offer a code editor that prints the outcome as a message indicating any potential mistakes in the code?**		
	6	**Does the system provide a code editor that provides syntax highlighting?**		
Debugging	7	**Does the system provide an output shell that shows the errors' locations to debug the code?**		
	8	**Does the system provide a code editor that predictively completes whatever I want to type?**		
Improving learning experiences	9	**Does the system provide a reflection note box that serves as a personal space where learners can document their thoughts and reflections as they progress through their learning journey?**		
	10	**Does the system provide tailored learning materials, recommendations, and feedback based on individual needs and preferences?**		
	11	**Does the system offer several options for programming languages?**		
	12	**Does the system offer programming learning videos?**		
	13	**Does the system offer extra material other than the course tutorials, such as textbooks?**		
	14	**Does the system offer several languages?**		
	15	**Does the system offer an interactive shell that supports the "Hints feature" to avoid careless mistakes?**		

Figure 5.8: The deployable version of the evaluation instrument form that can be used by programming educators.

Chapter 6

Programming Educators' Experiences with the Instrument

6.1 Chapter Overview

This chapter presents a case study conducted to investigate the programming educators experiences with the use of the proposed evaluation instrument presented in Chapter 5. This chapter is laid out as follows: Section 6.2 presents the research question; Section 6.3 presents the method used for this case study: Section 6.4 discusses the data analysis techniques used; Section 6.5 presents the results of this study; and Section 6.6 concludes this chapter.

6.2 Research question 6

In this study to explore the programming educators' experience on using the instrument, an online survey instrument was designed, cf. Appendix D, and distributed to address RQ6: **What are the attitudes of programming educators toward using the instrument to evaluate online coding tutorial systems?**

This study contributes to the knowledge area by exploring the experience of the programming educators on the use of the instrument to evaluate and select the effective online coding tutorial systems for novices, which is the main purpose of this thesis.

6.3 Study Method

To investigate how the targeted audience of the instrument developed and validated in this thesis finds the use of the instrument to evaluate online coding tutorial systems, an online sur-

vey as presented in Appendix D was distributed via two Whatsapp academic groups that have multi-national academic computer science educators. The WhatsApp groups were selected as a channel to have more participants for this case study because of their quick approach to finding the targeted participants. In recent research studies, it has been noted that there is significant usage of WhatsApp for academic reasons because of factors such as the real-time nature of the application and the convenience that comes along with it, and also the general use of the application in almost all regions of the world where the study is to be conducted with the aim of carrying out surveys and other related communications in study processes [207]. The two WhatsApp groups are: one is the official group for female lecturers in the computer science school at the Saudi Electronic University. In addition, the other group was the information technology research group. As presented in Appendix D, in the online survey, participants have been given a controlled scenario in order to use and evaluate the selected online coding tutorial systems by using the evaluation instrument developed and validated in this thesis. Moreover, to capture their feedback on their experience using the instrument, one open-ended question was asked.

6.3.1 Participants

The 24 invited participants came from a range of backgrounds; they were all computer science educators from both higher education and the classroom, and they had all previously used websites with coding tutorials. Educators with different experiences in programming education may work together to build a more widely recognised and usable tool that better fits the needs of all novice programmers [146]. The main reason for using WhatsApp to distribute the online poll was to attract teachers with varying levels of programming experience. About 35 people were initially thought to be the ideal sample size in order to gather a range of comments and viewpoints. However, it turned out that there were actually just 24 participants in the sample size for this study. There could be a number of reasons for the lower-than-expected participation rate. One of the reasons is that the case study was conducted in the summer semester of higher education in Saudi Arabia and the United Kingdom.

6.3.2 Procedure

This case study was conducted over a period of two weeks, from the end of July 2024 to the middle of August 2024. The procedure was followed in conducting this case study; a specific scenario was demonstrated to the participants in order to prepare them to evaluate the selected online coding tutorial systems after accepting the consent form in the first place in the online survey. Then, they were asked to complete the online coding tutorial systems' evaluation instrument form that was proposed earlier in Chapter 5. In addition, the main

contribution of this case study is to capture the programming educators' feedback about their experiences after evaluating such systems by using the instrument developed and validated in this thesis. Below, the online survey sections are described briefly, as shown in Appendix D, which contains four sections:

1. **First section:** In this section, the consent form has been introduced to the participants.
2. **Second section:** In this section, list pre-testing questions (demographics questions) are presented.
3. **Third section:** In this section, a list of instructions (the scenario) has been presented. **Firstly,** the online coding tutorial systems have been defined and introduced to the participants. **Secondary,** the participants were given some examples of online coding tutorial systems in order to give them a clear picture of what these systems look like. **Thirdly,** they were asked to select one online coding tutorial system. **Fourthly,** they were requested to write the selected systems' names.
4. **Fourth section:** In this section, the evaluation instrument's items that have been presented in the previous Chapter 5 in Figure 5.8 were provided in order to be used by the participants. The suggested assessment tool was shown to the participants, who were then instructed to go through each item one by one. This technique was employed to evaluate the instrument's usage progress. In computer science research, it is a useful approach [77].
5. Fifth section: This section is the core of this study because the programming educators have been asked one open-ended question, which is **"How did you find the instrument in evaluating online coding tutorial systems?"**. The aim of asking this question is to investigate the feedback of the programming educators on their experiences of using the instrument to evaluate online coding tutorial systems.

6.3.3 Demographics

Following the online survey's distribution, participant demographic data from the second section was gathered. The programming educators that participated in the data collection process fit this description. The participants' ages ranged from 25 to 60 years old, and the demographic data revealed, as shown in Figure 6.1, that they were from Saudi Arabia, the United Kingdom, and others. In addition, in terms of programming teaching experience, the data showed that 8% of the sample (n = 2) had less than five years of experience teaching, 38% (n = 9) had five to ten years of experience, and 54% of the sample (n = 13) had more than ten years of experience.

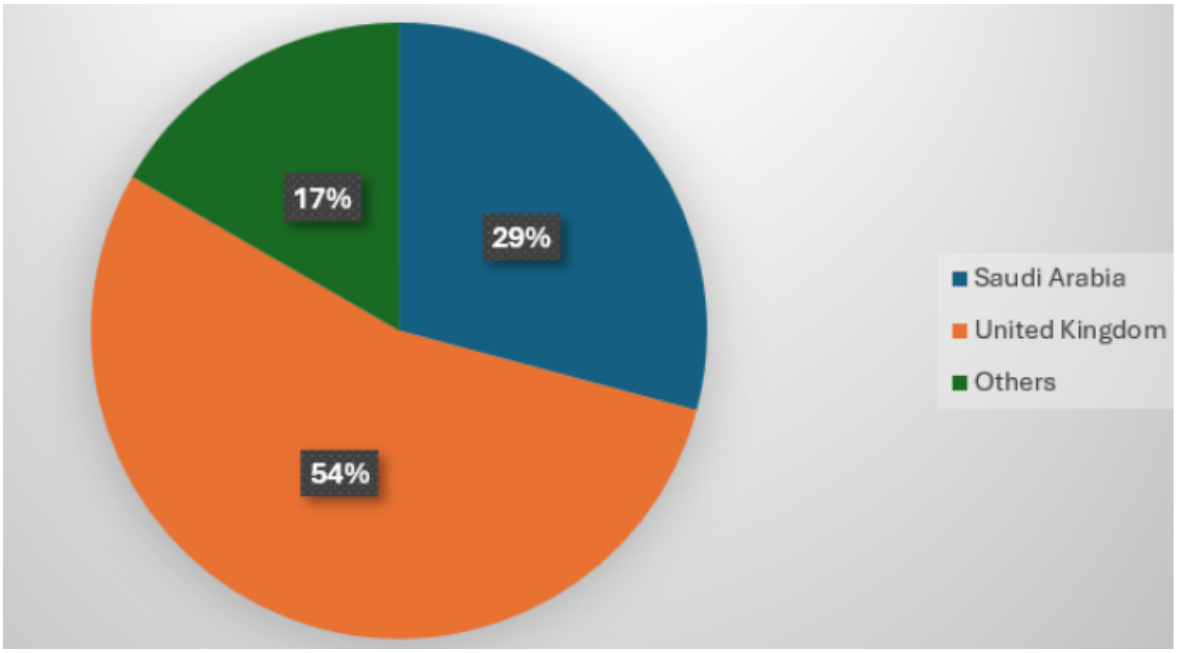

Figure 6.1: The nationalities of the programming educators involved in the case study

6.4 Data Analysis Technique

As discussed earlier in this chapter, the main purpose of this study is to investigate how programming educators (the targeted audience of the proposed instrument in this thesis) find the use of the instrument in a practical setting. Therefore, only the qualitative data that was collected from the open-ended question was analysed in order to present and analyse the programming educators experiences. The data analysis technique used to analyse the programming educators' answers to the open-ended question is **a thematic analysis** [205]. According to [36], to perform the thematic analysis, a number of phases had been completed: becoming acquainted with the data, creating preliminary codes or labels, looking for themes or primary ideas, evaluating themes or primary ideas, defining and labelling themes or primary ideas, and creating the report. The coding for this study was done by hand. The selection of codes was done with the intention of highlighting the aspects that respondents indicated were significant to them. From the twenty-four programming educators who had used the evaluation instrument, only 18 educators had responded to the open-ended question.

6.5 Study Findings

After the demographic questions, the participants were given a list of instructions that had been discussed in the previous section. Then, each programming educator had selected one

online coding system. For instance, as shown in Figure 6.2 most of the participants evaluated Replit. Moreover, some of the educators evaluated W3school, Learnpython, Learnjava, Tryruby, and Tryjavascript. In the following section, each educator evaluated the selected online coding tutorial system by using the instrument presented in the previous chapter (Figure 5.8).

Figure 6.2: Online coding tutorial systems were evaluated by the programming educators.

After evaluating the selected online coding tutorial systems by educators by using the instrument, the programming educators shared their thoughts about their experiences in using the proposed evaluation instrument to evaluate online coding tutorial systems, as discussed earlier in this chapter. The qualitative data gives an in-depth understanding of programming educators feedback about the instrument developed in this thesis. The responses to the open-ended question were short; however, these responses show very positive sights; most of the respondents had positive experiences using the provided assessment instrument to evaluate any online coding tutorial systems. For instance, one participant said, ”The instrument is a comprehensive tool to evaluate such a system.” Moreover, another participant said, ”Important aspects have been covered in this assessment tool.” All the responses are coded into main themes as shown in Table 6.1.

Code	Participant responses	Codes highlighting participant ideas: Significant ideas	Themes: Main idea
E1	”This instrument is helpful to evaluate online coding systems”	The use of the instrument	Easy to use
E2	”The instrument is a comprehensive tool to evaluate such a system.”	The content	Comprehensiveness
E3	”Important aspects have been covered in this assessment tool.”	The aspect covered	Comprehensiveness
E4	”Using this tool can support us to select good system for novices”	Supporting the learners	Effectiveness of the instrument
E5	”It is an instrument that can help us to evaluate systems”	The quality of the instrument	Effectiveness of the instrument
E6	”Good instrument”	The quality of the instrument	Effectiveness of the instrument
E7	”By using this assessment tool, the systems can be evaluated easily”	The use of the instrument	Easy to use
E8	”This instrument can be used for evaluating any online coding systems”	The flexibility of the instrument	System-specific evaluation
E9	”This instrument can be used easily”	The use of the instrument	Easy to use
E10	”This instrument is applicable for online coding systems”	The applicability of the instrument	System-specific evaluation
E11	”This instrument items are interesting and it covers all important aspects”	The content of the instrument	Comprehensiveness
E12	”This items are enough for such systems”	The content of the instrument	Comprehensiveness
E13	”This instrument contains specific features of online coding tutorial systems”	The applicability of the instrument	System-specific evaluation
E14	”My experience of evaluating the system was perfect”	The use of the instrument	Easy to use
E15	”This tool is efficient to evaluate the selected system”	The applicability of the instrument	Effectiveness of the instrument
E16	”This instrument can be used easily to test the systems”	The use of the instrument	Easy to use
E17	”The form has all important points to be considered”	The content of the instrument	Comprehensiveness
E18	”This instrument can be used for evaluating and selecting the best systems for novices”	The effectiveness of the instrument	Effectiveness of the instrument

Findings from the open-ended question, are data relating to the 18 programming educators' views and experiences about the use of the evaluation instrument in this thesis are presented in this section. As shown in Table 6.1, the identified themes will be presented in the following points; these points are shown in the following order; effectiveness of the instrument, instrument comprehensiveness, ease of use and applicability of the instrument, and finally, system-specific evaluation instrument. In addition, as shown in Table 6.1, the following themes were identified:

1. **Theme 1: Effectiveness of the instrument:** Terms such as "effectiveness" and "quality" dwelt on the usefulness of the instrument in evaluating the tutorials on coding offered online. The primary benefit of using the tool was distinguished by educators who noted that the instrument gives definite and practical judgements on the quality of educational platforms. Five educators expressed that the instrument supports them in evaluating systems. For example, E18 said that "This instrument can be used for evaluating and selecting the best systems for novices" and E4 states that ""Using this tool can support us to select good system for novices"

2. **Theme 2: Instrument comprehensiveness:** Terms like 'comprehensive' 'other important aspects' were used to capture the overall picture of the important criteria the instrument explored to assess coding tutorials adequately. Five programming educators revealed that the instrument covers the most important aspect. For instance, E3 said "Important aspects have been covered in this assessment tool." and E12 states "This items are enough for such systems".

3. **Theme 3: Easy to use:** Educators found the instrument easy to use and valuable across different learning environments as said by one user, ''It was quite easy to fill and very useful in different education settings." This makes the tool friendly to the educators so that anyone no matter his or her level of experience using the tool can easily use it. For instance, E9 states "This instrument can be used easily" and E16 says "This instrument can be used easily to test the systems".

4. **Theme 4: System-specific evaluation instrument:** Some of the terms include specific system and interactive system showed that educators appreciated the flexibility of the instrument that allowed for different online coding software to be used in education settings. For instance, E13 states "This instrument contains specific features of online coding tutorial system" and E10 "This instrument is applicable for online coding systems"

The findings of this study show the stasfaction of the programming educators toward the use of the proposed instrument to evaluate the selected online coding tutorial systems. The

programming educators's feedback and responses to the open-ended question show that the instrument meets their satisfaction. They emphasised the applicability, simplicity, and usefulness of the instrument, especially to evaluate online coding tutorial systems. This tool can be used across different such systems, as highlighted by the educators, which means that it can be adapted to fit the needs of programming educators internationally in different education contexts to support novices by selecting the effective system.

6.6 Chapter Summary

This chapter presents a detailed discussion of the application and the use of the proposed instrument to evaluate online coding tutorial systems by the programming educators. The results of the qualitative data analysis show that the programming educators found the instrument helpful and efficient for them to assess such systems. The programming educators agreed that the instrument provides a set of criteria for the analysis of online coding tutorial systems that can support them in selecting the effective system for novices. This study proves that the instrument stays valid and could positively contribute to improving novices' programming learning.

Chapter 7

Discussion

7.1 Chapter Overview

This thesis aims to propose an evaluation instrument for online coding tutorial systems to be used by programming educators to support novices in their programming learning journey. Therefore, this chapter presents a summary of the findings of the four design cycles for developing the proposed instrument, and it discusses how this thesis is evidence of the claimed primary contribution of this thesis. Moreover, the key findings, together with relevant literature, will be used to discuss these findings. This chapter discusses the research findings from the investigations done in the previous chapters (Chapter 4, Chapter 5, and Chapter 6). This chapter is structured as follows: Section 7.2 presents the findings of this thesis; Section 7.3 discusses this thesis findings; and Section 7.4 concludes this chapter.

7.2 Summary of the Findings

On the basis of the data gathered and presented in previous chapters (Chapter 4, Chapter 5 and Chapter 6, the following are the significant findings:

- Identifying a list of common programming learning challenges that novice learners need to overcome as the evaluation instrument's main components.
- Identifying an initial list of items of the evaluation instrument for online coding tutorial systems based on exploring a list of possible solutions for the programming learning problems that have been identified from the literature.
- Learners and educators found that most of the features in the current online coding tutorial systems and those proposed in the initial evaluation instrument are helpful.

- Learners and educators suggested some other interesting items to be added to the instrument, such as reflection notes, and based on their suggestions, the initial instrument had been updated.
- Current online coding tutorial systems have been found to miss some of the features in the proposed evaluation instrument, such as auto completion.
- It was noticed that the proposed instrument is missing one feature that was found in two current systems, and based on these findings, the evaluation instrument has been updated.
- Users are satisfied with the proposed features in the third version of the evaluation instrument based on the evaluation of the system prototype.
- Users suggested some other interesting features after they tested the system prototype, and based on that, the evaluation instrument had been updated.
- The final version of the evaluation instrument for online coding tutorial systems has been developed based on the four design cycles in this research.
- The proposed evaluation instrument for online coding tutorial systems has been validated by ten experts in the programming education field.
- It was noticed that the educators agreed that the proposed instrument is suitable to evaluate any online coding tutorial systems.

To develop the instrument, a DBR model was followed (like [1], [34]) which was based on McKenney's original generic model for design research (GMDR) [132]. Design-based research follows a cyclic process containing cycles of analysis, design, evaluation, and revision [132]. In the instrument development studies, the design process is divided into four design cycles and produces five versions of the evaluation instrument for online coding tutorial systems. The first version of the instrument emerges from the literature review. The second version (revision of versions 2 and 3) emerges from the initial (explorative) fact-finding studies. The fourth version of the instrument comes from the development of the system prototype, which corresponds to the design and construction of the GMDR [132]. Also, version four of the instrument comes from design cycle four, which corresponds to the evaluation and reflection phases of the GMDR. In design cycle four, the instrument goes through iterative evaluation, reflection, and revision cycles of the individual case study.

The main aim of this thesis was to develop and validate an evaluation instrument for online coding tutorial systems to be used by specialised programming educators. This research is framed by five questions. The first question is "RQ1: What are common programming learning difficulties for novices? Which supportive features are potential solutions for these

identified difficulties?" To address this, a systematic literature review has been conducted, which is discussed in Chapter 4 Section 4.2. Section 4.2 presents the first design cycle in this research that aims to identify common programming learning challenges by conducting a systematic literature review and a set of features as solutions to develop the first version (draft one) of the evaluation instrument for online coding tutorial systems.

In the systematic literature review of research conducted between 1980 and 2023, relevant articles were found. Then, we performed an initial inclusion screening based on title and abstract to get a subset of candidate studies that only focus on programming learning challenges, and we filtered the article set to 52 after removing duplicates and out-of-focus papers. From these 52 remaining articles, further screening was performed by considering full-text content, excluding articles that did not discuss programming learning difficulties, and removing duplicate articles and non-English articles. The final number of the selected articles was 7, which were [138],[35],[167],[155],[92],[203],[78]. From these articles, six major challenges for a novice programmer were filtered out: syntax of programming languages, structure of code, debugging, dividing functionality into procedures, and transferring algorithms to concrete implementation.

A semi-systematic review has been done to discover possible solutions for the set of common programming learning difficulties identified in the previous section. These identified challenges and solutions presented components and items of the instrument. In the analysis phase, in the second design cycle that is presented in Chapter 4 Section 4.3, an online survey was distributed among novice learners and educators to investigate their needs and their feedback on the tested online coding tutorial system (LearnPython) [175]. In this study, an online survey instrument was designed to address RQ2: What are the appropriate supportive features in online coding tutorial systems from learner and educator perspectives? This study was conducted for 2 months, from March 2021 to May 2021, after receiving ethical approval from the College of Science and Engineering. The questionnaire was distributed randomly to fellow educators and learners, and participants were asked to fill in a questionnaire on their opinions with the list of system features in the selected online coding tutorial system called LearnPython [175]. The target participants in this study encompassed novice, intermediate, and expert programmers. By distributing the online survey through various platforms, such as WhatsApp and email, the researchers aimed to attract participants with diverse programming backgrounds. Initially, the target sample size was set at around 200 participants to gather a wide range of feedback and suggestions from various individuals. However, the actual sample size for this study turned out to be only 37 participants.

The quantitative findings from the research shed light on the participants' perspectives regarding various features of online coding tutorial systems. The results revealed that a significant majority of participants recognised the value of incorporating coding lessons and tutorials into these systems, as it was believed to enhance novice learners' understanding of

coding concepts. This finding was particularly relevant for visual learners, who found it easier to grasp the basics of programming languages through this approach. Furthermore, the participants expressed mixed opinions regarding the inclusion of additional coding learning materials. While the majority agreed that offering supplementary resources such as books and extra learning materials would enhance their learning experience, a notable portion disagreed, considering this feature unhelpful in their educational journey. This discrepancy suggests the importance of considering individual preferences and needs when designing online coding tutorial systems.

In the qualitative findings, the participants were asked two open-ended questions to collect qualitative data. The first open-ended question was: ”What other helpful and usable features or characteristics of online coding tutorial systems would you suggest?” The learner and educator perspectives in the qualitative research study identified several suggestions and recommendations for improving online coding tutorial systems. From these suggestions, four features were filtered out: reflection note, content organisation, more coding lessons, and changes and updates to the instrument.

Moving forward, the third phase of the research presents the third design cycle in Section 4.4. Additionally, a comparative study is conducted to assess current online coding tutorial systems for the presence or absence of the identified supportive features. This study addresses RQ3: What are the supportive features that exist in current deployed online coding tutorial systems but are absent from the instrument? It involves looking at the top ten languages by popularity on GitHub, and seven systems were selected for analysis: LearnPython [175], TryJavaScript [159], LearnJava [173], Codecademy [49], LearnTypeScript [48], Tour of Go [75], RubyMonk [160], and LearnPHP [174]. From the analysis, 16 supportive features have been identified, and it is observed that most of the identified features exist in some systems. However, four supportive features are not provided by any of the studied systems, such as underlining syntax errors, reflection notes, visual maps, and auto-completion. Additionally, two features, quizzes and customised hints, are only provided by the single system under study. These findings emphasise the urgent need for enhanced support for novice programmers through a more considerate design process for online coding tutorial systems. Thus, the research aims to develop an evaluation instrument based on novice learners' perspectives.

Then, in the fourth design cycle, after developing a system prototype as presented in Chapter 4 Section 4.5, this prototype was evaluated to answer RQ4: ”Building on our research findings, what would an online coding tutorial system look like? Based on a prototype implementation, to what extent are typical learners satisfied with the features of such an online coding tutorial system?”, An online survey was distributed, and responses from the survey were analysed. Findings showed that the vast majority of users were satisfied with the features in Section 4.5.

After the instrument development process, the final version of the instrument has been validated by experts in Chapter 5 to answer RQ5. Then, a case study has been conducted to answer RQ6 in Chapter 6 to investigate the attitude of the target audience toward the instrument. The results revealed that the proposed instrument can be used by programming educators as an evaluation instrument to evaluate or select any online coding tutorial systems. The programming educators expressed the benefits and challenges they experienced when using the proposed evaluation instrument for online coding tutorial systems in Chapter 6.

7.3 Discussion of the Findings

The previous section presents the findings that have been found in this thesis. This section presents the discussions of these findings as reflections of the research questions answered in Chapter 4, Chapter 5 and Chapter 6.

7.3.1 Instrument Development

7.3.1.1 RQ1: What are common programming learning difficulties for novices? And which supportive features are potential solutions for these identified difficulties? (Design cycle one)

This question has been answered in Section 4.2. The findings show that novice learners are struggling with learning programming. Six programming learning problems were found, and a list of supportive features was identified from the literature. In addition, these features were grouped based on the six identified difficulties. Similar to other studies that discussed programming learning difficulties [203], this study finds that novices are struggling to understand the syntax and semantics of individual statements in the programming languages [177] [177]. In addition, this study identifies understanding functions and procedures as a real problem faced by novices [155].

The work of Kader et al. [103] lists difficulties in teaching and learning programming to improve the educators teaching approaches for basic programming courses, enhance students' interest, and increase students' performance in programming subjects. In this study, the programming learning problems were identified to propose and discuss possible solutions in online systems. Therefore, in this section, 13 supportive features were identified. For instance, in order to understand syntax, reporting syntax errors in a programming environment might help novice learners reduce mistakes in spelling, punctuation, and the order of keywords in their programs[59, 96]. Moreover, to assist in understanding the code structure,

visual maps are included to help beginners understand fundamental programming concepts, structure, and execution [201].

7.3.1.2 RQ2: What are appropriate supportive features in online coding tutorial systems from learner and educator perspectives? (Design cycle two)

Results from the first fact-finding study (Section 4.3) highlighted what educators and learners thought and measured satisfaction regarding the online coding tutorial system' features. Findings were compared and contrasted with the literature. The fact-finding study (open-ended questions in the online survey) offered a more detailed overview of learners' and educators' thoughts and any new supportive features suggested. This section, summarises the findings from the case study. First of all, a summary of the learners' and educators' feedback, then a summary of the new supportive features added to the initial instrument.

- (Educators and learners) satisfied with all the features.

Summary of new features from Chapter 4 and Section 4.3:

- **Content organisation**

  The open-ended questions' answers showed that some participants found organising the coding lessons in terms of basic coding concepts to advanced coding concepts might help them learn programming languages more easily.

- **Extra content lessons**

  In addition, the answers to the open-ended questions showed that some participants found that providing more coding lessons with more coding topics might help them in their learning journey.

- **Reflection note**

  Some participants think that reflecting on their programming learning might help them in their learning.

7.3.1.3 RQ3: What are the supportive features that exist in current deployed online coding tutorial systems but are absent from the instrument? Do the identified supportive features in the evaluation instrument exist in these systems? (Design cycle three)

Despite the limited number of systems analysed in this case study (Section 4.4), the result of the analysis showed that the current online coding tutorial systems have missed some

supportive features such as auto completion, underlining errors, reflection notes, and visual maps. However, this result is considered a motivation to propose an evaluation instrument for online coding tutorial systems in this research. Another main result from this study is identifying one new feature added to the instrument that was found in current online coding systems but is missing in the instrument. This feature has multi-language support.

- **Several programming languages**

  The results of this comparative analysis study showed that some current online coding tutorial systems are offering content with several options for different programming languages. For instance, the LearnPython platform [175] provides several programming languages, such as C++ and Java.

7.3.1.4 RQ4: Building on our research findings, what would a usable online coding tutorial system look like? Based on a prototype implementation, to what extent are typical novice learners satisfied with the features of such an online coding tutorial system? (Design cycle four)

The results of this evaluation study in Section 4.5 showed that most of the online coding tutorial system prototype users are satisfied with the features provided in the system prototype. In addition, one of the main suggestions highlighted by users in this evaluation study is the responses to the two open-ended questions. The users suggested some interesting features that might be helpful in improving programming learning.

- **Several languages**

  The open-ended questions showed that some users found that offering the content of the lessons in different languages, such as ”Arabic,” might help them learn programming languages more easily.

- **Learning videos**

  Moreover, the open-ended questions' answers showed that some users found that offering educational videos that teach coding concepts might help them learn programming languages more easily.

7.3.2 Instrument validation

7.3.2.1 RQ5: To what extent is it applicable to use the proposed evaluation instrument for OCTSs to evaluate any online coding tutorial systems?

As presented in Chapter 5, the validation study conducted on the final version of the design instrument for online coding tutorial systems revealed several key insights into its efficacy

and applicability for programming educators. The participants in the study provided feedback on various aspects of the instrument, highlighting its comprehensibility, specialisation, and the need for clarification. Firstly, educators found the evaluation instrument to be comprehensible, indicating that it was easy to understand and apply in the evaluation process. This aspect is crucial, as a complex or convoluted instrument could hinder its practical utility in assessing OCTSs effectively. The ability for educators to grasp the instrument easily enhances its usability and encourages its adoption within educational settings. Secondly, the specialization of the evaluation instrument for OCTSs was recognised by participants as a valuable attribute. This specificity ensures that the instrument addresses the unique features and requirements of online coding tutorial systems, rather than offering a generic evaluation approach. By tailoring the instrument to the specific needs of OCTSs, educators can conduct more accurate and insightful assessments, leading to better-informed decisions regarding the selection and implementation of such systems in programming education. However, despite its strengths, the validation study identified areas where the evaluation instrument could be improved. One notable finding was the absence of performance-related metrics within the instrument. Participants highlighted the importance of evaluating the performance of OCTSs based on technical and psychological factors, suggesting that the inclusion of performance items would enhance the instrument's comprehensiveness and utility.

In conclusion, the findings of the validation study support the claim that the proposed evaluation instrument for OCTSs is suitable for evaluating online coding tutorial systems. Educators described it as a comprehensive tool with specific features tailored to the assessment of OCTSs. Nevertheless, the study also underscored the need for ongoing refinement and enhancement of the instrument, particularly in areas such as performance evaluation. By addressing these areas of improvement, the evaluation instrument can continue to serve as a valuable resource for programming educators seeking to evaluate and improve online coding tutorial systems.

7.3.3 Programming educators' experiences on the instrument

7.3.3.1 RQ6: What are the attitudes of programming educators toward using the instrument to evaluate online coding tutorial systems?

As presented in Chapter 6, the use of the proposed instrument by programming educators to evaluate online coding tutorial systems and to investigate their experiences of using the instrument. The findings of the qualitative data analysis demonstrate that the tool was useful and effective for programming educators in assessing such systems. The educators concurred that the tool offers a set of standards for evaluating online coding tutorial programs, which can support them choose the effective online coding tutorial systems for novices.

7.4 Chapter Summary

The claim of this thesis is twofold, as reflected in the thesis statement. The first claim is that *Online coding tutorial systems should be designed and deployed in such a way that they satisfy novice learners' needs to ensure effective interactive platform use, leading to the successful delivery of programming teaching by educators and consequent positive impacts for novice learners*. The second claim of the thesis statement proposes that *Such effectiveness can be achieved by the adoption of a systematically constructed instrument of features to support programming educators to evaluate and select the most appropriate online coding tutorial system for their learning context.* This thesis explores this claim by developing an evaluation instrument that can be used by programming educators to evaluate and select effective online coding tutorial systems. This work is the first that proposes an instrument for evaluating online coding tutorial systems. Through validating the proposed instrument by experienced programming educators, the thesis concludes that the proposed instrument is applicable to be used as an evaluation instrument.

Chapter 8

Conclusion

8.1 Chapter Overview

This chapter demonstrates that the objectives of the research highlighted in Chapter 1 have been fully or partially met, indicating the strengths and limitations of the study. This chapter is structured as follows: Section 8.2 presents the summary of this research work; Section 8.3 discusses the contributions to computer science education; Section 8.4 outlines the research achievements; and finally, Section 8.5 presents the research limitations.

8.2 Summary

In order to achieve the aim of this thesis, an evaluation instrument for online coding tutorial systems was developed and validated in order to support programming educators in evaluating online coding tutorial systems. The empirical evidence in this research suggests that the proposed evaluation instrument for OCTSs can enhance a programming learning journey. Furthermore, based on the data analysis carried out from the study in Chapter 6, it can be stated that this research achieved that objective. Using design cycles as required in design-based research, an evaluation instrument for online coding tutorial systems was developed. Moreover, the fuzzy Delphi method has been used to validate the instrument.

8.2.1 Theoretical development of the evaluation instrument

The main contribution of this thesis was the evaluation instrument based on the design-based research process. This approach ensured that the refinement of the instrument was done systematically over several cycles thus ensuring that the instrument measures various param-

eters of the online coding tutorial system including usability, pedagogy, learner engagement level, adaptability and feedback mechanisms.

1. **Iterative Design Process:**

   The process of design following the developmental-cycle approach, where the design was developed in cycles, was instrumental in making the instrument both substantive and feasible. Thus, the approach used was the design-based research, which was based on the review of the previous literature and established that there is a significant lack of the evaluation tool that would respond to the needs of programming education [137]. Thus, every cycle had input from experts and educators, which enabled the development of the instrument and its adaptation to meet the needs of novice programmers, and thus can be applied across different educational settings.

2. **Integration of multiple perspectives:**

   The fact that both academic and front-line educators being involved in the development of the instrument was a strength in the process. This diversity of the authors guaranteed that the instrument did not only reflect the state-of-practice in educational technology, but also the realities of educators involved in programming education. It also enabled progressive development, since feedback was received from each cycle to improve and optimise the aspect of the instrument [137].

3. **Prevalence of online coding tutorial systems:**

   The development of this instrument is particularly important, given the codes for online coding tutorial systems. That is, as these systems emerge, educators face numerous possibilities, which can significantly vary in quality and efficacy. The instrument offers a way to assess these systems systematically and grounded on available research, so educators can arrive at the best course of action that will support their instructional philosophy and their learners. As such, the instrument aims at the criteria of usability and pedagogical benefits to ensure that only efficient and effective systems that are also educationally beneficial are selected [137] [134].

8.2.2 Assumption of validity of the instrument

The validation of the instrument was done using the Fuzzy Delphi Method (FDM), a systematic approach that uses the Delphi technique of consensus among experts, involving the fuzzy logic technique that deals with the imprecision and uncertainty inherent in human judgement [226]. This was done out of the belief that the FDM is more effective in integrating expert opinions, making the validation more accurate.

1. **Expert consensus and fuzzy logic:**

   The FDM enabled researchers to gather the opinion of experts on the essential sections of the instrument and make a consensus. This is especially useful in educational research, as it involves expert judgement in the validation of tools used in sophisticated, context-specific use [226]. To this end, the FDM made sure the instrument used was valid, reliable, and transportable across different contexts in education, thereby adding to the value of using the instrument in programming educators.

2. **Ensuring applicability across contexts:**

   The validation process revealed that educators can well employ the instrument to assess and choose the most suitable online coding tutorial systems that would fit their teaching-learning context. This is especially important in programming education, where the effectiveness of a learning system can greatly affect the student learning and retention of the content, as noted by [191]. During the FDM process, several experts participated in the development of the instrument, and thus the developed instrument is more likely to be valid in diverse educational settings [226].

3. **Reflecting emerging trends:** The application of the FDM also helped identify other related trends and best practices in the application of educational technology. The process of programming education is constantly developing, and it is important to incorporate new ideas and approaches to assessment into the instrument [226] [184]. This approach helps in ensuring that the instrument is most current, in as much as improvements in educational technology and instructional methods.

8.3 Emerging Findings and Contributions to the Larger Field of Computer Science Education Research

This section discusses the implications of these investigations for the field of computer science education research. It is concerned with its educational, theoretical and methodological significance. This section builds on the prior contributions by locating the findings within the existing literature in computer science education. It focuses on how they respond to some of the issues that are relevant in the field. This section provides a detailed analysis of the findings across all chapters with detailed exploration. It highlights how the research enhances understanding of the evaluation of online coding tutorial systems, the experience of novice programmers, and the practical application of instructional design principles.

1. **Integration of pedagogical principles:**

This study also offers the following main research contributions; the assessment of technological resources has incorporated instructional design principles as the focal point. Many of the current available assessment tools focus primarily on the technical aspects of the goods and services, such as the usability and performance. However, this research points to the fact that the systems enhance learning, and this is a critical feature of programming education. Such components as the pedagogical effectiveness and the learners' engagement when incorporated in the instrument provide a better evaluation of programming education as intended by the objectives of the programming education [137] [198].

2. **Support for online and hybrid learning:**

   Especially with the shift of the world system towards online learning, there is a need for tools that help educators identify available technologies for their use. This instrument developed in the current study will be useful, where educators can select systems that are technically efficient, but also educationally efficient [137] [134]. This contribution is especially relevant now, as the tendencies towards using the online and blended learning environments remain high and rising in many educational contexts [134].

3. **Theoretical and methodological contributions:**

   Apart from the practical impact, this research provides important theoretical findings into the assessment of educational technologies. Using the theories of usability, instructional design effectiveness, learner interaction, flexibility, and feedback for the development of this instrument, this research offers a framework that can be used in future research in the evaluation of online learning systems [191] [134]. The instrument's validation process also has a methodological contribution, showing how the Fuzzy Delphi Method can be applied in the development and validation of educational instruments [226] [149] [139].

4. **Implications for future research:**

   This dissertation offers the practical solution for educators, but also opens further development in the context of educational technology research. Reflections on the iterative design and validation can help create new evaluation instruments and approaches, especially in the fields that combine technology and education [191] [184]. Further studies could extend this research by assessing the effect of the instrument on education results in the long term, or by applying the instrument in another educational context, rather than programming education.

8.4 Research Achievements

In this thesis, a list of objectives has been achieved, as presented in Table 8.1, that have led to the development and validation of an instrument for evaluating online coding tutorial systems.

Research Objectives	Achieved?
Identifying the main components of the evaluation instrument by identifying the programming learning challenges.	✓
Identifying the initial evaluation instrument' items by identifying the possible solutions to the identified challenges (the main components)	✓
Updating the initial evaluation instrument' items by identifying new features of online coding tutorial systems from system stakeholders' perspectives, i.e. novice learners and educators to be evaluated.	✓
Examine a set of current online coding tutorial systems to determine whether they provide the identified features presented in the instrument and whether any features that exist in the selected current systems are missing in the proposed instrument	✓
Developing an interactive online coding tutorial system prototype that provides most of the identified features that exist in the evaluation instrument as items.	✓
Evaluating the system prototype with real users, capturing their interaction with the system prototype, and collecting their feedback and suggestions on the system to improve date the instrument' items.	✓
Validating the proposed evaluation instrument with experts in the field of computing education.	✓
Conducting a study to investigate how the target audience (programming educators) find the evaluation instrument to evaluate online coding tutorial systems.	✓

Table 8.1: List of research objectives that have been achieved

8.5 Research Limitations and Future Work

- **Instrument development**

  Literature review study (Chapter 4 -Section 4.2)

While considerable effort was devoted to conducting a systematic literature review, it is essential to acknowledge the possibility of bias influencing the selection of articles and supportive features. Despite employing rigorous methods, the inclusion or exclusion of certain papers may have been inadvertently influenced by subjective judgement, personal preferences, or unconscious biases. Such biases can introduce limitations in the representativeness and comprehensiveness of the selected research papers [58]. To mitigate this limitation in future research, it is recommended that clear and objective criteria be established for the article selection process. These criteria should be predefined and based on the research objectives and questions. By defining and adhering to transparent and objective criteria, the potential for bias can be minimised, ensuring a more impartial selection of research papers.

The generalisability of the study's results may be constrained due to the specific databases searched and the exclusion of other potential sources. In this study, the search was primarily focused on three databases: ACM, IEEE, and Google Scholar. While these databases are widely recognised and frequently used in academic research, they may not provide a comprehensive representation of the entire body of knowledge in the field of programming education [76]. To enhance the generalisability of future research, it is recommended to broaden the scope of the search by including a wider range of databases. In addition to the aforementioned databases, relevant journals, conferences, and books in the field of programming education should also be considered. Expanding the search to include diverse sources can help capture a more extensive and diverse range of studies, thereby enhancing the generalisability of the findings [76].

Another potential limitation of this study relates to the time constraints imposed during the literature review process. The search and analysis were conducted within a specific timeframe, which may have inadvertently excluded relevant papers published after the search cutoff date. This time restriction could result in a potential lack of inclusion of recent research and emerging trends in the field. To address this limitation, future research should consider extending the literature search period to encompass a more current and up-to-date range of publications. By expanding the timeframe, researchers can ensure the inclusion of the most recent studies, thus enhancing the relevance and currency of the findings. The language bias is an inherent limitation in this study, as the search was restricted to articles published in English. By excluding papers published in other languages, valuable research contributions from non-English-speaking regions may have been overlooked. This limitation may introduce a potential bias in the findings and limit the cross-cultural generalisability of the study.

To overcome this limitation, future research should strive to include relevant literature published in languages other than English. Employing language translation services or collaborating with researchers fluent in different languages can help overcome lan-

guage barriers and ensure a more comprehensive and inclusive analysis [84].

Current systems evaluation study (Chapter 4 -Section 4.3)

Initially, a sample size of approximately 200 participants was targeted to gather feedback and suggestions from a wide range of individuals. However, the actual sample size for this study was only 37 participants, which is significantly smaller than expected. The smaller number of participants could be attributed to various factors, indicating potential biases in the sample. Firstly, it is possible that individuals who received the survey were not interested in the subject matter or did not perceive it as relevant to their current activities or research focus. Consequently, they may have chosen not to participate, resulting in a lower response rate.

The timing of the survey distribution could also have influenced participation. If the survey was conducted during a period when potential participants were occupied with other academic or personal commitments, their willingness or ability to complete the questionnaire might have been affected [188]. While the study managed to collect valuable insights and feedback from the 37 participants, it is important to recognise the limitations imposed by the small sample size. Generalising the findings should be done with caution due to the reduced representativeness of the sample. The extent to which the results can be applied to a broader population of programming educators and learners may be limited [188]. Future research endeavours should aim to increase the sample size by implementing additional recruitment strategies or targeting specific programming communities or platforms known for their engagement in educational discussions. Expanding the participant pool would provide a more comprehensive and representative set of feedback, enhancing the validity and generalisability of the study's findings. It would help mitigate the biases associated with the small sample size and improve the overall quality of the research.

Comparative study (Chapter 4 -Section 4.4)

Some limitations should be taken into account when interpreting the results of this current systems analysis study. Firstly, the case study was conducted on only seven online coding tutorial systems, and therefore the results may not be generalised to all current online coding tutorial systems. Comparative studies comparing the programming learning platform with other similar platforms or educational methods would help identify its unique strengths and weaknesses. Incorporating qualitative research methods, such as interviews or focus groups, would provide deeper insights into participants' experiences, perceptions, and suggestions. Additionally, ongoing user feedback and iterative design processes can lead to continuous improvement of the programming learning platform, ensuring that it meets the needs and preferences of its users.

Evaluation study (Chapter 4 -Section 4.5)

The study has several potential biases and limitations that should be considered when interpreting the findings. Firstly, the sample used in the study primarily consisted of participants from Saudi Arabia and the UK, which may limit the generalisability of the results to a broader population. It would be beneficial to include participants from a more diverse range of countries and cultural backgrounds in future studies to obtain a more comprehensive understanding. Another potential bias is self-selection bias, as participants voluntarily chose to participate in the study. This could introduce bias if those who were more motivated or had a particular interest in programming were more likely to participate. Consequently, the sample may not fully represent the wider population of programming novice learners.

Social desirability bias is another consideration, as participants' responses in the survey may be influenced by their desire to provide socially acceptable answers or appear favorably. This bias could affect the accuracy and reliability of the reported satisfaction levels and other self-reported data. Additionally, recall bias could impact the accuracy of the data collected, as participants may struggle to accurately recall their interactions with the programming learning platform or their satisfaction levels. Memory limitations or selective recall could introduce inaccuracies in the reported data. Despite these limitations, there are several future opportunities for research in this area. Firstly, diversifying the sample to include participants from various regions, cultures, and educational backgrounds would provide a more comprehensive understanding of the effectiveness and usability of the programming learning platform. Longitudinal studies tracking participants' progress and performance over an extended period would offer insights into the long-term impact of the platform on their skill development and career outcomes.

- **Instrument validation**

  Instrument validation study (Chapter 5)

  The study has some limitations that should be considered when interpreting the findings. Firstly, the sample used in the study, which primarily consisted of participants, was a small sample of experts. It would be beneficial to include more experts in future studies to obtain a more comprehensive understanding.

- **Using the instrument in practice**

  Programming educators' experiences with the instrument (Chapter 6)

  The study has some limitations that should be considered when interpreting the findings. Firstly, the sample used in the study primarily consisted of participants from

Saudi Arabia and the United Kingdom, which may limit the generalisability of the results to a broader population. It would be beneficial to include participants from a more diverse range of cultural backgrounds in future studies to obtain a more comprehensive understanding.

Appendix A

Online Survey: The Learners and Educators Perspectives Study

A.1 The consent form:

You are invited to participate in this online survey on evaluating the features and characteristics of online coding tutorials systems from learners and educators' perspectives. This is a research study being conducted by a PhD student at the University of Glasgow. This study should take (45-60) minutes of your time approximately to complete.

- **ABOUT THE STUDY**

  This survey aims to evaluate the features and characteristics of Online Coding Tutorials Systems (OCTSs) from learners and educators' perspectives, the data that will be collected from this study will indicates if learners and educators find the features and characteristics of OCTSs helpful.

- **WHY ME?**

  You are being asked to participate in this study because you are a learner or an educator.

- **PARTICIPATION**

  Please understand that your participation in this survey is voluntary and you may, anytime, refuse to take part in this research. In case you do wish to withdraw, you can freely decline in answering questions that may feel uncomfortable to you.

- **PRIVACY**

  Anonymity and confidentiality being preserved. All data will be dealt with in accordance with the University's GDPR guidelines.

- **CONTACT**

  Thank you for taking the time to complete this survey" For further inquiries about this re-search study please feel free to contact me via email:

  o.alasmari.1@research.gla.ac.uk

- **CONSENT**

  By accepting this form, I agree and affirm that:

  1. I have read and understood the purpose of the information stated above.
  2. I am over 18 years of age or older.
  3. I am submitting this form and participating in this research study voluntarily.

A.2 Demographic Questions

As shown in Table A.1, four demographic questions were asked.

Demographic Questions	Answers
1- What is your age?	————-
2- How long have you been coding?	0 – One year , Two years- Three years, More than three years
3- How do you identify your primary role?	An educator, A learner
4- Where are you located?	England, Scotland, Saudi Arabia, Others

Table A.1: Demographic questions

A.3 User tasks

- **SCENARIOS 1:**
  - Click here to open the online coding tutorials systems. https://www.learnpython.org/
  - Go through the introduction page.
  - Read the content topics list (Learn the Basics-Data Science Tutorials-Advanced Tutorials)
- **SCENARIOS 2:**

  - Click on any section in the content topics list.
  - Go through the exercise instructions.
  - Look at the code segment in the left shell (script shell).
  - Press the “Run” button.
  - Look at the output that will be shown to you in the other shell.

- **SCENARIOS 3:**
  - Repeat the same steps in Scenarios 2 but write a wrong code segment.
  - Press the “Run” button.
  - Look at the output that will be shown to you in the other shell and be aware of if any syntax error is highlighted.

A.4 Post-Testing Questions (Part 1)

This part contains a set of assessment statements as shown in Table A.2; the following statements are about the tested online coding tutorials system’ features and characteristics. We are asking whether you found these features and characteristics helpful. Please rate the extent to which you agree/disagree with the following statements.

Questions	Answers
1- Providing a list of coding exercises that cover most of basic and complex coding concepts.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
2- Providing a REPL (interactive shell) that allows learners to actively practice coding, and type code snippets that executed incrementally.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
3- Providing a REPL (Interactive shell that supports "Hints feature" that help learners to be guided when to use a particular function and help to avoid careless mistakes.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
4- Providing a REPL interpreters that print the result as message that indicates errors that might code has.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
5- Offering extra material other than the course tutorials such as video.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
6- Providing a REPL interpreters that provide syntax highlighting.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
7- Providing a REPL interpreters that underline syntax error that indicates errors that might code has.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
8- Providing a REPL interpreters that print the result as detailed message that indicates errors that might code has.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
9- Providing code templates in the script shell helps me to use a programming language to implement an algorithm for solving a specific problem.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
10- Identifying errors locations in the output shell to debug the code.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
11- Offering complete example programs and worked solutions helps to understand basic concepts the programming language.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree

Table A.2: A set of assessment statements (Part 1)

A.5 Post-Testing Questions (Part 2)

Please rate the extent to which you agree/disagree with the following statements (As shown in TableA.3. In terms of features and characteristics, the tested online coding tutorials system SHOULD also be .

Questions	Answers
1- Providing an assessment activity that allows learners to evaluate his/her skills at the end of each section.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
2- Providing a visual map feature to understand the execution process of the code.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
3- Providing a code editor that predictively completes whatever I want to type might helps to produce good code.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree

Table A.3: A set of assessment statements (Part2)

A.6 Open-ended questions

As presented in TableA.4, two open-ended questions were asked.

Questions	Answers
What other helpful features or characteristics of Online Coding Tutorials Systems would you suggest?	————-
Any other comments?	————-

Table A.4: Open-ended questions

Appendix B

Online Survey: Python OCTS Evaluation Study

B.1 The consent form:

You are invited to participate in this online survey on evaluating the features of Python OCTS prototype from learners' perspectives. This is a research study being conducted by a PhD student at the University of Glasgow. This study should take (20-30) minutes of your time approximately to complete. be

- **ABOUT THE STUDY**

  This survey aims to measure the participants' satisfaction levels for the features provided in our system prototype. The data that will be collected from this study will indicates if learners find the features of our system prototype usable and helpful.

- **WHY ME?**

  You are being asked to participate in this study because you are a programming learner.

- **PARTICIPATION**

  Please understand that your participation in this survey is voluntary and you may, anytime, refuse to take part in this research. In case you do wish to withdraw, you can freely decline in answering questions that may feel uncomfortable to you.

- **PRIVACY**

  Anonymity and confidentiality being preserved. All data will be dealt with in accordance with the University's GDPR guidelines.

- **CONTACT**

  Thank you for taking the time to complete this survey" For further inquiries about this re-search study please feel free to contact me via email: o.alasmari.1@research.gla.ac.uk

- **CONSENT**

  By accepting this form, I agree and affirm that: 1. I have read and understood the purpose of the information stated above. 2. I am over 18 years of age or older. 3. I am submitting this form and participating in this research study voluntarily.

B.2 Demographic Questions

Demographic Questions	Answers
1- What is your age?	————-
2- How would you describe your coding expe-rience?	No prior experience , Beginner, Interme-diate, Advanced
3- Where are you located?	Saudi Arabia, United Kingdom, Other

B.3 Testing the content of the system porotype

This section is a set of instructions that you should go through to test our the content-based features of the system porotype. In addition, this section contains a set of assessment state-ments to measure whether you found these content-based features helpful.

B.3.1 First scenarios to test content-based features

- Click here to open our Python OCTS prototype. python-octs.herokuapp.com
- Click on Try Coding button.
- Enter your personal details to register. Read the content topics list (lessons list; Lesson 1"Hello world"...etc).
- Select any lesson and click on it.
- Try the code editor.
- Reflect at the end of the lesson in the Reflection Note box.
- Click on Quiz button.

- Click on Solution button (available after one minute).
- Click on “Other materials “button on the top of the menu bar.

B.3.2 Post-testing questions

Please rate the extent to which you agree/disagree with the following statements.

Questions	Answers
1- Providing a list of coding lessons and exercises helps me to understand basic Python concepts.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
2- Providing a quiz after each lesson allows me to test my understanding of basic concepts of Python programming language.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
3- Offering complete example programs and worked solutions helps me to understand basic concepts of Python programming language.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
4- Offering extra learning materials as other learning sources helps me to understand basic Python concepts.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
5- Providing a list of coding lessons that organized from basic to advance helps me to understand basic Python concepts.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
6- Providing a reflection note in each coding lesson helps me to reflect on each coding lesson.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree

B.4 Testing the features in the system porotype

This section is a set of instructions that you should go through to test the technical-based features of the system porotype. In addition, this section contains a set of assessment statements to measure whether you found these features helpful.

B.4.1 Second scenarios to test technical-based features

- Click here to open our Python OCTS prototype. python-octs.herokuapp.com
- Click on Try Coding button.
- Enter your personal details to register.

- Read the content lessons list (lessons list; Lesson 1”Hello world”...etc).
- Select one lesson and click on it.
- Look at the code segment in the left shell (script shell) and be aware of syntax highlighting.
- Type a wrong code statement such as ”print[word] ”in the script shell and press the “Run” button. Be aware of syntax error message and error locations that will be shown to you in the other shell.
- Repeat the previous step and be aware of the yellow highlight shown on the problematic code segment and the hints that will pop up.
- Go to Lesson 4, then click on Next button on the left side of the page and see how the code is visualized.
- Go to Lesson 2, and look at the attached image below the code editor that shows how the code editor could predictively complete whatever you want to type.

B.4.2 Post-testing questions

Please rate the extent to which you agree/disagree with the following statements.

Notes:

- Syntax is the set of rules that define what the various combinations of symbols mean.
- Visual map is a way to take a programming concept and transform it into a visual aid for better understanding.
- Scripts shell is the left part in the code editor where we can type code segments.

B.5 Testing the technical features in the system porotype

B.5.1 Post-testing question

Please rate the extent to which you agree/disagree with the following statement. In terms of features, the tested online coding tutorials system prototype SHOULD also has the feature shown in this image.

Questions	Answers
1- Reporting syntax error messages in the output shell helps me to reduce mistakes in spelling, punctuation and order of keywords in my code and to understand syntax of Python programming language.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
2- Providing syntax highlighting helps me to identify keywords and become familiar with Python programming language syntax.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
3- The visual map given by Python OCTS helps me understand the execution process of the code.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
4- Providing code templates in the script shell helps me to use a programming language to implement an algorithm for solving a specific problem.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
5- Identifying errors locations in the output shell helps me to debug my code.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
6- Providing some guidance through the hints helps me to find and fix the bug.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
7- Providing detailed error messages helps me to solve the error effectively.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree

Questions	Answers
1- Providing a code editor that predictively completes whatever I want to type might helps me to produce good code. (As shown in the image)	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
2- Providing several programming languages might helps me to produce good code.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
3- Providing a REPL interpreters that underline syntax error that indicates errors that might code has might helps me to produce good code.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree

B.6 Open-ended questions

Questions	Answers
What did you like best about the experience?	————
What did you dislike about the experience?	—————

Appendix C

Online Survey: The Experts Evaluation Study

C.1 The consent form:

You are invited to participate in this online survey on evaluating the evaluation instrument' items. This is a research study being conducted by a PhD student at the University of Glasgow. This study should take (20-30) minutes of your time approximately to complete.

- **ABOUT THE STUDY**

  This survey aims to measure the participants' satisfaction levels for the evaluation instrument' items. The data that will be collected from this study will indicates if experts find the items of the instrument are valid.

- **WHY ME?**

  You are being asked to participate in this study because you are an expert in computing education field.

- **PARTICIPATION**

  Please understand that your participation in this survey is voluntary and you may, anytime, refuse to take part in this research. In case you do wish to withdraw, you can freely decline in answering questions that may feel uncomfortable to you.

- **PRIVACY**

  Anonymity and confidentiality being preserved. All data will be dealt with in accordance with the University's GDPR guidelines.

- **CONTACT**

  Thank you for taking the time to complete this survey" For further inquiries about this re-search study please feel free to contact me via email: o.alasmari.1@research.gla.ac.uk

- **CONSENT**

  By accepting this form, I agree and affirm that: 1. I have read and understood the purpose of the information stated above. 3. I am submitting this form and participating in this research study voluntarily.

C.2 Demographic Questions

Demographic Questions	Answers
1- What is your age?	———-
2- How long have you been instructing in programming?	0 – Five years , Five years- Ten years, More than ten years
3- How do you identify your primary role?	An university professor, A school programming instructor
4- Where are you located?	Saudi Arabia, United Kingdom, Japan, China, Others

C.2.1 Post-testing questions

Please rate the extent to which you agree/disagree with the following statements.

Questions	Answers
Understanding basic concepts: 1- The system provides a list of code exercises covering the majority of fundamental and advanced coding principles.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
Understanding basic concepts: 2- The system provides practical examples that demonstrate how programming concepts and techniques can be applied in real-world scenarios.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
Understanding basic concepts: 3- The system provides quizzes to test novice learners' comprehension and solidify their understanding of programming concepts.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
Structure of code: 4- The system has a visual map function that could aid students in comprehending how the code is executed.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
Syntax of programming languages: 5- The code editor that the system offers prints the outcome as a message indicating any potential mistakes in the code.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
Syntax of programming languages: 6- The system provides a code editor that provides syntax highlighting.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
Debugging: 7- The system provides an output shell that shows errors' locations to debug the code.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
Debugging: 8- The system provides a code editor that predictively completes whatever I want to type.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
Improving learning experiences: 9- The system provides a reflection note box that serves as a personal space where learners can document their thoughts and reflections as they progress through their learning journey.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
Improving learning experiences: 10- The system provides tailored learning materials, recommendations, and feedback based on individual needs and preferences.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
Improving learning experiences: 11- The system offers several options for programming languages.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
Improving learning experiences: 12- The system offers programming learning videos.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
Improving learning experiences: 13- The system offers extra material other than the course tutorials, such as textbooks.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
Improving learning experiences: 14- The system offers several languages.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree
Improving learning experiences: 15- The system provides an interactive shell that supports the "Hints feature" to avoid careless mistakes.	Strongly Disagree, Disagree, Neither, Agree, Strongly Agree

Appendix D

Online Survey: The Educators User Case Study

D.1 The consent form:

You are invited to participate in this online survey on using the provided instrument to evaluate any online coding tutorial systems. This is a research study being conducted by a PhD student at the University of Glasgow. This study should take (20-30) minutes of your time approximately to complete.

- **ABOUT THE STUDY**

  This survey aims to capture the programming educators' attitude toward using the provided instrument to evaluate any online coding tutorial systems. The data that will be collected from this study will indicates if the targeted audience find the evaluation instrument applicable to be used for such systems.

- **WHY ME?**

  You are being asked to participate in this study because you are a programming educator.

- **PARTICIPATION**

  Please understand that your participation in this survey is voluntary and you may, anytime, refuse to take part in this research. In case you do wish to withdraw, you can freely decline in answering questions that may feel uncomfortable to you.

- **PRIVACY**

  Anonymity and confidentiality being preserved. All data will be dealt with in accordance with the University's GDPR guidelines.

- **CONTACT**

  Thank you for taking the time to complete this survey" For further inquiries about this re-search study please feel free to contact me via email: o.alasmari.1@research.gla.ac.uk

- **CONSENT**

  By accepting this form, I agree and affirm that: 1. I have read and understood the purpose of the information stated above. 2. I am over 18 years of age or older. 3. I am submitting this form and participating in this research study voluntarily.

D.2 Demographic Questions

Demographic Questions	Answers
1- What is your age?	————-
2- How long have you been instructing in programming?	0 – Five years , Five years- Ten years, More than ten years
3- How do you identify your primary role?	An university professor, A school programming instructor
4- Where are you located?	Saudi Arabia, United Kingdom, Others

D.3 Instructions

- Select one online coding tutorial system
- evaluate it by using the below list of items, and leave below the name of the system

D.4 The evaluation Instrument

D.5 Open-ended question

One open-ended question was asked.

Question	Answer
How did you find the instrument in evaluating online coding tutorial systems?	————-

Table D.1: Open-ended question

Questions	Yes ✓	No ✓
Understanding basic concepts: 1- Does the system provide a list of code exercises covering most fundamental and advanced coding principles?	Yes	No
Understanding basic concepts: 2- Does the system provide practical examples that demonstrate how programming concepts and techniques can be applied in real-world scenarios?	Yes	No
Understanding basic concepts: 3- Does the system provide quizzes to test novice learners' comprehension and solidify their understanding of programming concepts?	Yes	No
Structure of code: 4- Does the system have a visual map function that could aid learners in comprehending how the code is executed?	Yes	No
Syntax of programming languages: 5- Does the system offer a code editor that prints the outcome as a message indicating any potential mistakes in the code?	Yes	No
Syntax of programming languages: 6- Does the system provide a code editor that provides syntax highlighting?	Yes	No
Debugging: 7- Does the system provide an output shell that shows the errors' locations to debug the code?	Yes	No
Debugging: 8- Does the system provide a code editor that predictively completes whatever I want to type?	Yes	No
Improving learning experiences: 9- Does the system provide a reflection note box that serves as a personal space where learners can document their thoughts and reflections as they progress through their learning journey?	Yes	No
Improving learning experiences: 10- Does the system provide tailored learning materials, recommendations, and feedback based on individual needs and preferences?	Yes	No
Improving learning experiences: 11- Does the system offer several options for programming languages?	Yes	No
Improving learning experiences: 12- Does the system offer programming learning videos?	Yes	No
Improving learning experiences: 13- Does the system offer extra material other than the course tutorials, such as textbooks?	Yes	No
Improving learning experiences: 14- Does the system offer several languages?	Yes	No
Improving learning experiences: 15- Does the system offer an interactive shell that supports the "Hints feature" to avoid careless mistakes?	Yes	No

Bibliography

[1] Mireilla Bikanga Ada. Using a mobile web application for assessment feedback to enhance student motivation, engagement and communication in tertiary education. PhD thesis, University of the West of Scotland, 2017.

[2] Michael Adler and Erio Ziglio. Gazing into the oracle: The Delphi method and its application to social policy and public health. Jessica Kingsley Publishers, 1996.

[3] S Ahmad and Juzlinda Ghazali. Programming teaching and learning: Issues and challenges. Fstm. Kuis. Edu. My, 16(1):724–398, 2020.

[4] Marzieh Ahmadzadeh, Dave Elliman, and Colin Higgins. An analysis of patterns of debugging among novice computer science students. In Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education, pages 84–88, 2005.

[5] Ayat Al Ahmad and Randa Obeidallah. Studying the effectiveness of a proposed methodology for teaching programming labs online and students" perspectives toward it during covid-19: A case study of hashemite university. iJIM, 16(05):53, 2022.

[6] Mostafa Al-Emran and Khaled Shaalan. Learners and educators attitudes towards mobile learning in higher education: State of the art. In 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pages 907–913. IEEE, 2015.

[7] Satu Alaoutinen and Kari Smolander. Student self-assessment in a programming course using bloom's revised taxonomy. In Proceedings of the fifteenth annual conference on Innovation and technology in computer science education, pages 155–159, 2010.

[8] Ohud Alasmari, Jeremy Singer, and Mireilla Bikanga Ada. Analysis of research into the teaching and learning of programming: An updated review. In 2024 9th International STEM Education Conference (iSTEM-Ed), pages 1–6. IEEE, 2024.

[9] Ohud Alasmari, Jeremy Singer, and Mireilla Bikanga Ada. Online coding tutorial systems: A new category of programming learning platforms. In 2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC), pages 2222–2227. IEEE, 2024.

[10] Ohud Alasmari, Jeremy Singer, and Mireilla Bikanga Ada. Python octs: Design, implementation, and evaluation of an online coding tutorial system prototype. In 2024 IEEE World Engineering Education Conference (EDUNINE), pages 1–6. IEEE, 2024.

[11] Ohud Abdullah Alasmari, Jeremy Singer, and Mireilla Bikanga Ada. Do current online coding tutorial systems address novice programmer difficulties? In Proceedings of the 15th International Conference on Education Technology and Computers, pages 242–248, 2023.

[12] Myrtede Alfred, Laura H Barg-Walkow, Joseph R Keebler, and Alex Chaparro. Checking all the boxes: a checklist for when and how to use checklists effectively. BMJ Quality & Safety, 2024.

[13] B. Alizadeh. A formal approach to debug polynomial datapath designs. 17th Asia and South Pacific Design Automation Conference, 2012.

[14] I Elaine Allen and Jeff Seaman. Digital compass learning: Distance education enrollment report 2017. Babson survey research group, 2017.

[15] John R Anderson, C Franklin Boyle, and Brian J Reiser. Intelligent tutoring systems. Science, 228(4698):456–462, 1985.

[16] Terry Anderson and Julie Shattuck. Design-based research: A decade of progress in education research? Educational researcher, 41(1):16–25, 2012.

[17] Leonard A Annetta, James Minogue, Shawn Y Holmes, and Meng-Tzu Cheng. Investigating the impact of video games on high school students' engagement and learning about genetics. Computers & Education, 53(1):74–85, 2009.

[18] Paolo Antonucci, Christian Estler, Durica Nikolić, Marco Piccioni, and Bertrand Meyer. An incremental hint system for automated programming assignments. In Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education, pages 320–325, 2015.

[19] Michal Armoni, Orni Meerbaum-Salant, and Mordechai Ben-Ari. From scratch to "real" programming. ACM Transactions on Computing Education, 14(4):1–15, 2015.

[20] Anthony R Artino Jr. Online learning: Are subjective perceptions of instructional context related to academic success? The Internet and Higher Education, 12(3-4):117–125, 2009.

[21] William G Axinn and Lisa D Pearce. Mixed method data collection strategies. Cambridge University Press, 2006.

[22] Lynne P Baldwin and Jasna Kuljis. Learning programming using program visualization techniques. In Proceedings of the 34th Annual Hawaii International Conference on System Sciences, pages 8–pp. IEEE, 2001.

[23] Sasha Barab and Kurt Squire. Design-based research: Putting a stake in the ground. The journal of the learning sciences, 13(1):1–14, 2004.

[24] Elizabeth F Barkley, K Patricia Cross, and Claire H Major. Collaborative learning techniques: A handbook for college faculty. John Wiley & Sons, 2014.

[25] Matthew Barr. Student attitudes to games-based skills development: Learning from video games in higher education. Computers in human behavior, 80:283–294, 2018.

[26] Brigid Barron and Linda Darling-Hammond. Teaching for meaningful learning: A review of research on inquiry-based and cooperative learning. book excerpt. George Lucas Educational Foundation, 2008.

[27] Theresa Beaubouef and John Mason. Why the high attrition rate for computer science students: some thoughts and observations. ACM SIGCSE Bulletin, 37(2):103–106, 2005.

[28] Brett A Becker. An effective approach to enhancing compiler error messages. In Proceedings of the 47th ACM Technical Symposium on Computing Science Education, pages 126–131, 2016.

[29] Brett A Becker, Paul Denny, Raymond Pettit, Durell Bouchard, Dennis J Bouvier, Brian Harrington, Amir Kamil, Amey Karkare, Chris McDonald, Peter-Michael Osera, et al. Compiler error messages considered unhelpful: The landscape of text-based programming error message research. In Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education, pages 177–210. 2019.

[30] Randy L Bell, Lara Smetana, and Ian Binns. Simplifying inquiry instruction. The science teacher, 72(7):30–33, 2005.

[31] Jens Bennedsen and Michael E Caspersen. Failure rates in introductory programming. AcM SIGcSE Bulletin, 39(2):32–36, 2007.

[32] David C Berliner. The near impossibility of testing for teacher quality. Journal of teacher education, 56(3):205–213, 2005.

[33] H Russell Bernard and Harvey Russell Bernard. Social research methods: Qualitative and quantitative approaches. Sage, 2013.

[34] Mireilla Bikanga Ada. Using design-based research to develop a mobile learning framework for assessment feedback. Research and Practice in Technology Enhanced Learning, 13(1):3, 2018.

[35] Yorah Bosse and Marco Aurélio Gerosa. Why is programming so difficult to learn? patterns of difficulties related to programming learning mid-stage. ACM SIGSOFT Software Engineering Notes, 41(6):1–6, 2017.

[36] Virginia Braun and Victoria Clarke. Using thematic analysis in psychology. Qualitative Research in Psychology, 3(2):77–101, 2006.

[37] Virginia Braun, Victoria Clarke, Elicia Boulton, Louise Davey, and Charlotte McEvoy. The online survey as a qualitative research tool. International journal of social research methodology, 24(6):641–654, 2021.

[38] Karen Brennan and Mitchel Resnick. New frameworks for studying and assessing the development of computational thinking. In Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada, volume 1, page 25, 2012.

[39] Ann L Brown. Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. The journal of the learning sciences, 2(2):141–178, 1992.

[40] Joshua Charles Campbell, Abram Hindle, and José Nelson Amaral. Syntax errors just aren't natural: Improving error reporting with language models. In Proceedings of the 11th Working Conference on Mining Software Repositories, pages 252–261, 2014.

[41] Saul Carliner. An overview of online learning. 2004.

[42] Iain Chalmers, Douglas G Altman, et al. Systematic reviews. BMJ Publishing London, 1995.

[43] Lijia Chen, Pingping Chen, and Zhijian Lin. Artificial intelligence in education: A review. Ieee Access, 8:75264–75278, 2020.

[44] Ching-Hsue Cheng and Yin Lin. Evaluating the best main battle tank using fuzzy decision theory with linguistic criteria evaluation. European journal of operational research, 142(1):174–186, 2002.

[45] Edith Cherry. Programming for design: From theory to practice. John Wiley & Sons, 1998.

[46] A Ciptono, S Setiyono, F Nurhidayati, and R Vikaliana. Fuzzy delphi method in education: A mapping. In Journal of Physics: Conference Series, volume 1360, page 012029. IOP Publishing, 2019.

[47] Paul Cobb, Jere Confrey, Andrea DiSessa, Richard Lehrer, and Leona Schauble. Design experiments in educational research. Educational researcher, 32(1):9–13, 2003.

[48] Codecademy. LearnTypeScript, 2022. `https://www.codecademy.com/learn/learn-typescript`.

[49] Codecademy. Codecademy, 2023. `https://www.codecademy.com`.

[50] Codecombat. Codecombat, 2023. `https://codecombat.com/`.

[51] Code.org. Code.org, 2023. `https://code.org/educate/curriculum`.

[52] Design-Based Research Collective. Design-based research: An emerging paradigm for educational inquiry. Educational researcher, 32(1):5–8, 2003.

[53] Mick P Couper. Web surveys: A review of issues and approaches. The public opinion quarterly, 64(4):464–494, 2000.

[54] Coursera. Coursera, 2023. `https://www.coursera.org/`.

[55] Tyne Crow, Andrew Luxton-Reilly, and Burkhard Wuensche. Intelligent tutoring systems for programming education: a systematic review. In Proceedings of the 20th Australasian Computing Education Conference, pages 53–62, 2018.

[56] Leslie A Curry, Ingrid M Nembhard, and Elizabeth H Bradley. Qualitative and mixed methods provide unique contributions to outcomes research. Circulation, 119(10):1442–1452, 2009.

[57] Nada Dabbagh and Anastasia Kitsantas. Personal learning environments, social media, and self-regulated learning: A natural formula for connecting formal and informal learning. The Internet and higher education, 15(1):3–8, 2012.

[58] Roxana Daneshjou, Mary P Smith, Mary D Sun, Veronica Rotemberg, and James Zou. Lack of transparency and potential bias in artificial intelligence data sets and algorithms: a scoping review. JAMA dermatology, 157(11):1362–1369, 2021.

[59] Paul Denny, Andrew Luxton-Reilly, and Ewan Tempero. All syntax errors are not equal. In Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education, pages 75–80, 2012.

[60] Paul Denny, Andrew Luxton-Reilly, Ewan Tempero, and Jacob Hendrickx. Understanding the syntax barrier for novices. In Proceedings of the 16th annual joint conference on Innovation and technology in computer science education, pages 208–212, 2011.

[61] Omer Deperlioglu and Utku Kose. The effectiveness and experiences of blended learning approaches to computer programming education. Computer Applications in Engineering Education, 21(2):328–342, 2013.

[62] Shivangi Dhawan. Online learning: A panacea in the time of covid-19 crisis. Journal of educational technology systems, 49(1):5–22, 2020.

[63] Don A Dillman, Jolene D Smyth, and Leah Melani Christian. Internet, phone, mail, and mixed-mode surveys: The tailored design method. John Wiley & Sons, 2014.

[64] Chris Done. TryHaskell, 2022. `https://tryhaskell.org`.

[65] Christopher Douce, David Livingstone, and James Orwell. Automatic test-based assessment of programming: A review. Journal on Educational Resources in Computing (JERIC), 5(3):4–es, 2005.

[66] Matthew W Easterday, Daniel Rees Lewis, and Elizabeth M Gerber. Design-based research process: Problems, phases, and applications. Boulder, CO: International Society of the Learning Sciences, 2014.

[67] edx. edx, 2023. `https://www.edx.org/`.

[68] Weimiao Fan and Zheng Yan. Factors affecting response rates of the web survey: A systematic review. Computers in human behavior, 26(2):132–139, 2010.

[69] Murray J Fisher and Andrea P Marshall. Understanding descriptive statistics. Australian critical care, 22(2):93–97, 2009.

[70] Sue Fitzgerald, Renée McCauley, Brian Hanks, Laurie Murphy, Beth Simon, and Carol Zander. Debugging from the student perspective. IEEE Transactions on Education, 53(3):390–396, 2009.

[71] Bent Flyvbjerg. Case study. The Sage handbook of qualitative research, 4:301–316, 2011.

[72] freecodecamp. freecodecamp, 2023. `https://www.freecodecamp.org/`.

[73] D Randy Garrison and Heather Kanuka. Blended learning: Uncovering its transformative potential in higher education. The internet and higher education, 7(2):95–105, 2004.

[74] Kevin Gary. Project-based learning. Computer, 48(9):98–100, 2015.

[75] Andrew Gerrand et al. A Tour of Go, 2022. `https://go.dev/tour`.

[76] Adrian Gheorghe, Tracy E Roberts, Jonathan C Ives, Benjamin R Fletcher, and Melanie Calvert. Centre selection for clinical trials and the generalisability of results: a mixed methods study. PLoS One, 8(2):e56560, 2013.

[77] Paul Gill, Kate Stewart, Elizabeth Treasure, and Barbara Chadwick. Methods of data collection in qualitative research: interviews and focus groups. British dental journal, 204(6):291–295, 2008.

[78] Anabela Gomes and Antonio Mendes. A teacher's view about introductory programming teaching and learning: Difficulties, strategies and motivations. In Proceedings of the IEEE Frontiers in Education Conference, pages 1–8, 2014.

[79] Anabela Gomes and António José Mendes. An environment to improve programming education. In Proceedings of the 2007 international conference on Computer systems and technologies, pages 1–6, 2007.

[80] Samuel D Gosling, Simine Vazire, Sanjay Srivastava, and Oliver P John. Should we trust web-based studies? a comparative analysis of six preconceptions about internet questionnaires. American psychologist, 59(2):93, 2004.

[81] TR Green. G & m. ptre (1996) usability analysis of visual programming environments: A'cognitive dimensions' framework. Journal of Visual Languages and Computing, 7.

[82] Jan Gregar. Research design (qualitative, quantitative and mixed methods approaches). Book published by SAGE Publications, 228, 1994.

[83] Jiancheng Guan and Nan Ma. A comparative study of research performance in computer science. Scientometrics, 61:339–359, 2004.

[84] Louise Guillouet, Amit K Khandelwal, Rocco Macchiavello, Madhav Malhotra, and Matthieu Teachout. Language barriers in multinationals and knowledge transfers. Review of Economics and Statistics, pages 1–56, 2024.

[85] Philip J Guo. Online python tutor: embeddable web-based program visualization for cs education. In Proceeding of the 44th ACM technical symposium on Computer science education, pages 579–584, 2013.

[86] Mark Guzdial. Education teaching computing to everyone. Communications of the ACM, 52(5):31–33, 2009.

[87] Mark Guzdial. Exploring hypotheses about media computation. In Proceedings of the ninth annual international ACM conference on International computing education research, pages 19–26, 2013.

[88] Mark Guzdial. Learner-centered design of computing education: Research on computing for everyone. Morgan & Claypool Publishers, 2015.

[89] Brian Hanks, Charlie McDowell, David Draper, and Milovan Krnjajic. Program quality with pair programming in cs1. In Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education, pages 176–180, 2004.

[90] Christoph Hannebauer, Marc Hesenius, and Volker Gruhn. Does syntax highlighting help programming novices? Empirical Software Engineering, 23(5):2795–2828, 2018.

[91] I Harb. The effectiveness of a blended learning program on developing and retention of palestinian tenth graders' english writing skills. Unpublished Master Thesis, Islamic University, Gaza, Palestine, 2013.

[92] Ahmad Sobri Hashim, Rohiza Ahmad, and Muhammad Shafiq Shahrul Amar. Difficulties in learning structured programming: A case study in utp. In 7th World Engineering Education Forum, pages 210–215, 2017.

[93] heroku. heroku, 2023. `https://www.heroku.com/`.

[94] Ivo Herweijer. TryRuby, 2022. `https://try.ruby-lang.org`.

[95] Anthony JG Hey, Tony Hey, and Gyuri Pápay. The computing universe: a journey through a revolution. Cambridge University Press, 2014.

[96] Maria Hristova, Ananya Misra, Megan Rutter, and Rebecca Mercuri. Identifying and correcting java programming errors for introductory computer science students. ACM SIGCSE Bulletin, 35(1):153–156, 2003.

[97] Tsun-Yu Huang, Wen-Kuo Chen, Venkateswarlu Nalluri, and Thao-Trang Huynh-Cam. Evaluating e-teaching adoption criteria for indian educational organizations using fuzzy delphi-topsis approach. Mathematics, 10(13):2175, 2022.

[98] Gwo-Jen Hwang, Pei-Shan Tsai, Chin-Chung Tsai, and Judy CR Tseng. A novel approach for assisting teachers in analyzing student web-searching behaviors. Computers & Education, 51(2):926–938, 2008.

[99] Akira Ishikawa, Michio Amagasa, Tetsuo Shiga, Giichi Tomizawa, Rumi Tatsuta, and Hiroshi Mieno. The max-min delphi method and fuzzy delphi method via fuzzy integration. Fuzzy sets and systems, 55(3):241–253, 1993.

[100] Galina Ivanova, Vasil Kozov, and Pavel Zlatarov. Gamification in software engineering education. In 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pages 1445–1450. IEEE, 2019.

[101] David W Johnson, Roger T Johnson, and Karl A Smith. Cooperative learning: Improving university instruction by basing practice on validated theory. Journal on Excellence in University Teaching, 25(4):1–26, 2014.

[102] Insung Jung and Colin Latchem. A model for e-education: Extended teaching spaces and extended learning spaces. British Journal of Educational Technology, 42(1):6–18, 2011.

[103] Rozita Kadar, Naemah Abdul Wahab, Jamal Othman, Maisurah Shamsuddin, and Siti Balqis Mahlan. A study of difficulties in teaching and learning programming: a systematic literature review. International Journal of Academic Research in Progressive Education and Development, 10(3):591–605, 2021.

[104] Yasmin B Kafai and Quinn Burke. Constructionist gaming: Understanding the benefits of making games for learning. Educational psychologist, 50(4):313–334, 2015.

[105] Yasmin B Kafai and Mitchel Resnick. Constructionism in practice: Designing, thinking, and learning in a digital world. Routledge, 1996.

[106] Caitlin Kelleher and Randy Pausch. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. ACM computing surveys (CSUR), 37(2):83–137, 2005.

[107] Khanacademy. Khanacademy, 2023. `https://www.khanacademy.org/computing/computer-programming`.

[108] Ada S Kim and Amy J Ko. A pedagogical analysis of online coding tutorials. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, pages 321–326, 2017.

[109] Barbara Kitchenham. Procedures for performing systematic reviews. Keele, UK, Keele University, 33(2004):1–26, 2004.

[110] Barbara Kitchenham and Stuart Charters. Guidelines for performing systematic literature reviews in software engineering. 2007.

[111] Barbara Kitchenham et al. Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE-2007-01, Software Engineering Group, School of Computer Science and Mathematics, Keele University, 2007.

[112] Dimitra Kokotsaki, Victoria Menzies, and Andy Wiggins. Project-based learning: A review of the literature. Improving schools, 19(3):267–277, 2016.

[113] Michael Kölling, Bruce Quig, Andrew Patterson, and John Rosenberg. The bluej system and its pedagogy. Computer Science Education, 13(4):249–268, 2003.

[114] Utku Köse. A web based system for project-based learning activities in "web design and programming" course. Procedia-Social and Behavioral Sciences, 2(2):1174–1184, 2010.

[115] Theodora Koulouri, Stanislao Lauria, and Robert D Macredie. Teaching introductory programming: A quantitative evaluation of different approaches. ACM Transactions on Computing Education (TOCE), 14(4):1–28, 2014.

[116] Ranjit Kumar. Research methodology: A step-by-step guide for beginners. Sage, 2018.

[117] Kartikadyota Kusumaningtyas, Eko Dwi Nugroho, and Adri Priadana. Online integrated development environment (ide) in supporting computer programming learning process during covid-19 pandemic: A comparative analysis. IJID (International Journal on Informatics for Development), 9(2):66–71, 2020.

[118] Essi Lahtinen, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. A study of the difficulties of novice programmers. Acm sigcse bulletin, 37(3):14–18, 2005.

[119] Kris MY Law, Victor CS Lee, and Yuen-Tak Yu. Learning motivation in e-learning facilitated computer programming courses. Computers & Education, 55(1):218–228, 2010.

[120] Jaeho Lee, Wonsung Sohn, Kyeong Hur, Sunghun Ahn, Inhwan Yoo, Youngkwon Bae, Dukhoi Koo, and Seungki Shin. A delphi study for the direction to design the curriculum of computer education in elementary school. Journal of The Korean Association of Information Education, 25(1):1–11, 2021.

[121] Qing Li. A novel likert scale based on fuzzy sets theory. Expert Systems with Applications, 40(5):1609–1618, 2013.

[122] Harold A Linstone, Murray Turoff, et al. The delphi method. Addison-Wesley Reading, MA, 1975.

[123] Raymond Lister, Elizabeth S Adams, Sue Fitzgerald, William Fone, John Hamer, Morten Lindholm, Robert McCartney, Jan Erik Moström, Kate Sanders, Otto Seppälä, et al. A multi-national study of reading and tracing skills in novice programmers. ACM SIGCSE Bulletin, 36(4):119–150, 2004.

[124] Andrew Luxton-Reilly, Ibrahim Albluwi, Brett A Becker, Michail Giannakos, Amruth N Kumar, Linda Ott, James Paterson, Michael James Scott, Judy Sheard, and Claudia Szabo. Introductory programming: a systematic literature review. In Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education, pages 55–106, 2018.

[125] Marjorie M MacKinnon. Core elements of student motivation in problem-based learning. New directions for teaching and learning, 1999(78):49–58, 1999.

[126] Lauri Malmi, Ian Utting, and Amy J. Ko. Tools and Environments, pages 639–662. Cambridge University Press, 2019.

[127] John H Maloney, Kylie Peppler, Yasmin Kafai, Mitchel Resnick, and Natalie Rusk. Programming by choice: urban youth learning programming with scratch. In Proceedings of the 39th SIGCSE technical symposium on Computer science education, pages 367–371, 2008.

[128] Lauren E Margulieux, Mark Guzdial, and Richard Catrambone. Subgoal-labeled instructional material improves performance and transfer in learning to develop mobile applications. In Proceedings of the ninth annual international conference on International computing education research, pages 71–78, 2012.

[129] Samiha Marwan, Joseph Jay Williams, and Thomas Price. An evaluation of the impact of automated programming hints on performance and learning. In Proceedings of the 2019 ACM Conference on International Computing Education Research, pages 61–70, 2019.

[130] Renee McCauley, Sue Fitzgerald, Gary Lewandowski, Laurie Murphy, Beth Simon, Lynda Thomas, and Carol Zander. Debugging: a review of the literature from an educational perspective. Computer Science Education, 18(2):67–92, 2008.

[131] Sean McDirmid. Usable live programming. In Proceedings of the 2013 ACM international symposium on New ideas, new paradigms, and reflections on programming & software, pages 53–62, 2013.

[132] Susan McKenney and Thomas C Reeves. Conducting educational design research. Routledge, 2018.

[133] Ruth McQuirter. Lessons on change: Shifting to online learning during covid-19. Brock Education: A Journal of Educational Research and Practice, 29(2):47–51, 2020.

[134] Giansalvatore Mecca, Donatello Santoro, Nazzareno Sileno, and Enzo Veltri. Diogene-ct: tools and methodologies for teaching and learning coding. International Journal of Educational Technology in Higher Education, 18(1):12, 2021.

[135] Rodrigo Pessoa Medeiros, Geber Lisboa Ramalho, and Taciana Pontual Falcão. A systematic literature review on teaching and learning introductory programming in higher education. IEEE Transactions on Education, 62(2):77–90, 2018.

[136] Microsoft. Microsoft, 2023. `https://visualstudio.microsoft.com/downloads/`.

[137] Kathy A Mills, Jen Cope, Laura Scholes, and Luke Rowe. Coding and computational thinking across the curriculum: A review of educational outcomes. Review of Educational Research, page 00346543241241327, 2024.

[138] Iain Milne and Glenn Rowe. Difficulties in learning and teaching programming—views of students and tutors. Education and Information technologies, 7(1):55–66, 2002.

[139] Vincent W Mitchell. The delphi technique: An exposition and application. Technology Analysis & Strategic Management, 3(4):333–358, 1991.

[140] Shuhaida Mohamed Shuhidan, Margaret Hamilton, and Daryl D'Souza. Understanding novice programmer difficulties via guided learning. In Proceedings of the 16th annual joint conference on Innovation and technology in computer science education, pages 213–217, 2011.

[141] Jeovani Morales, Rosana Montes, Noe Zermeno, Jeronimo Duran, and Francisco Herrera. The use of fuzzy linguistic information and fuzzy delphi method to validate by consensus a questionnaire in a blended-learning environment. In International conference on information processing and Management of Uncertainty in knowledge-based systems, pages 137–149. Springer, 2018.

[142] Khanzadi Mostafa, Nasirzadeh Farnad, and Alipour Majid. Using fuzzy-delphi technique to determine the concession period in bot projects. In 2010 2nd IEEE International Conference on Information and Financial Engineering, pages 442–446. IEEE, 2010.

[143] Andreas Leon Aagaard Moth, Joergen Villadsen, and Mordechai Ben-Ari. Syntaxtrain: relieving the pain of learning syntax. In Proceedings of the 16th Annual Joint

Conference on Innovation and Technology in Computer Science Education, pages 387–387, 2011.

[144] Peter Mozelius and Marie Olsson. Putting the programming hut online; self learning for the net-generation. In ECEL 2015, European Conference on e-Learning, page 417, 2015.

[145] Peter M Nardi. Doing survey research: A guide to quantitative methods. Routledge, 2018.

[146] Joshua Noble. Programming interactivity. " O'Reilly Media, Inc.", 2012.

[147] Zhanat Nurbekova, Talant Tolganbaiuly, Bahyt Nurbekov, Ainur Sagimbayeva, and Zhadira Kazhiakparova. Project-based learning technology: An example in programming microcontrollers. International Journal of Emerging Technologies in Learning (iJET), 15(11):218–227, 2020.

[148] A. Okhotin. Describing the syntax of programming languages using conjunctive and boolean grammars. ArXiv, 2020.

[149] Chitu Okoli and Suzanne D Pawlowski. The delphi method as a research tool: an example, design considerations and applications. Information & management, 42(1):15–29, 2004.

[150] David B Palumbo. Programming language/problem-solving research: A review of relevant issues. Review of educational research, 60(1):65–89, 1990.

[151] John F Pane and Brad A Myers. Usability issues in the design of novice programming systems. 1996.

[152] Marina Papastergiou. Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation. Computers & education, 52(1):1–12, 2009.

[153] Arnold Pears, Stephen Seidman, Lauri Malmi, Linda Mannila, Elizabeth Adams, Jens Bennedsen, Marie Devlin, and James Paterson. A survey of literature on the teaching of introductory programming. Working group reports on ITiCSE on Innovation and technology in computer science education, pages 204–223, 2007.

[154] Raymond Scott Pettit, John D Homer, Kayla Michelle McMurry, Nevan Simone, and Susan A Mengel. Are automated assessment tools helpful in programming courses? In 2015 ASEE Annual Conference & Exposition, pages 26–230, 2015.

[155] Martinha Piteira and Carlos Costa. Learning computer programming: study of difficulties in learning programming. In Proceedings of the 2013 International Conference on Information Systems and Design of Communication, pages 75–80, 2013.

[156] Paul Piwek and Simon Savage. Challenges with learning to program and problem solve: an analysis of student online discussions. In Proceedings of the 51st ACM Technical Symposium on Computer Science Education, pages 494–499, 2020.

[157] Jan L Plass, Bruce D Homer, and Charles K Kinzer. Foundations of game-based learning. Educational psychologist, 50(4):258–283, 2015.

[158] Tjeerd Plomp and Nienke Martien Nieveen. An introduction to educational design research: Proceedings of the seminar conducted at the East China Normal University, Shanghai (PR China), November 23-26, 2007. Stichting Leerplan Ontwikkeling (SLO), 2010.

[159] PluralSight. TryJavaScript, 2022. `https://www.javascript.com/try`.

[160] Sidu Ponnappa and Jasim A Basheer. Ruby Monk, 2022. `http://rubymonk.com`.

[161] Leo Porter, Cynthia Bailey Lee, Beth Simon, and Daniel Zingaro. Peer instruction: Do students really learn from peer discussion in computing? In Proceedings of the seventh international workshop on Computing education research, pages 45–52, 2011.

[162] processing. processing, 2023. `https://processing.org/`.

[163] Mikhail Semenovich Prokopiev, Elena Zotikovna Vlasova, Tatiana N Tretiakova, Maxim Anatolyevich Sorochinsky, and Rimma Alekseevna Soloveva. Development of a programming course for students of a teacher training higher education institution using the programming language python. Propositos y representaciones, 8(3):33, 2020.

[164] Keith F Punch. Introduction to social research: Quantitative and qualitative approaches. sage, 2013.

[165] Python-OCTS. Python-OCTS, 2023. `https://python-octs.herokuapp.com/`.

[166] Yizhou Qian and James Lehman. Students' misconceptions and other difficulties in introductory programming: A literature review. ACM Transactions on Computing Education, 18(1):1–24, 2017.

[167] Yizhou Qian, Peilin Yan, and Mingke Zhou. Using data to understand difficulties of learning to program: A study with chinese middle school students. In Proceedings of the ACM Conference on Global Computing Education, pages 185–191, 2019.

[168] Charles C Ragin. Qualitative comparative analysis using fuzzy sets (fsqca). Configurational comparative methods: Qualitative comparative analysis (QCA) and related techniques, pages 87–122, 2009.

[169] MOGANADASS RAMALINGAM, SITI HAJAR HALILI, and SAEDAH SIRAJ. Experts' agreement of the personalized m-learning curriculum model based on fuzzy delphi method. Indonesian Research Journal in Education— IRJE—, pages 407–420, 2019.

[170] Tathagata Ray, Aruna Malapati, and NL Bhanu Murthy. Teaching computer programming using moocs in multiple campuses: Challenges and solutions. In 2016 IEEE Eighth International Conference on Technology for Education (T4E), pages 160–163. IEEE, 2016.

[171] Thomas Reeves. Design research from a technology perspective. In Educational design research, pages 64–78. Routledge, 2006.

[172] Peter Reimann. Design-based research. In Methodological choice and design: Scholarship, policy and practice in social and educational research, pages 37–50. Springer, 2010.

[173] Ron Reiter. LearnJava, 2022. `https://www.learnjavaonline.org/`.

[174] Ron Reiter. LearnPHP, 2022. `https://www.learn-php.org/`.

[175] Ron Reiter. LearnPython, 2022. `https://www.learnpython.org`.

[176] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, et al. Scratch: programming for all. Communications of the ACM, 52(11):60–67, 2009.

[177] Robert S Rist. Teaching eiffel as a first language. Journal of object-oriented programming, 9(1):30–41, 1996.

[178] Eric Roberts. An interactive tutorial system for java. ACM SIGCSE Bulletin, 38(1):334–338, 2006.

[179] Anthony Robins, Janet Rountree, and Nathan Rountree. Learning and teaching programming: A review and discussion. Computer science education, 13(2):137–172, 2003.

[180] Shanna Russ and Foad Hamidi. Online learning accessibility during the covid-19 pandemic. In Proceedings of the 18th International Web for All Conference, pages 1–7, 2021.

[181] N Amira M Saffie, Khairul A Rasmani, et al. Fuzzy delphi method: Issues and challenges. In 2016 International Conference on Logistics, Informatics and Service Sciences (LISS), pages 1–7. IEEE, 2016.

[182] Eddie Antonio Santos, Joshua Charles Campbell, Dhvani Patel, Abram Hindle, and José Nelson Amaral. Syntax and sensibility: Using language models to detect and correct syntax errors. In Proceedings of the IEEE 25th International Conference on Software Analysis, Evolution and Reengineering, pages 311–322, 2018.

[183] Stephen R Schach. Software engineering. Aksen associates, 1990.

[184] Carsten Schulte, Johannes Magenheim, Kathrin Müller, and Lea Budde. The design and exploration cycle as research and development framework in computing education. In 2017 IEEE Global Engineering Education Conference (EDUCON), pages 867–876. IEEE, 2017.

[185] scratch. scratch, 2023. `https://scratch.mit.edu/`.

[186] Pratim Sengupta, John S Kinnebrew, Satabdi Basu, Gautam Biswas, and Douglas Clark. Integrating computational thinking with k-12 science education using agent-based computation: A theoretical framework. Education and Information Technologies, 18:351–380, 2013.

[187] Judy Sheard, S Simon, Margaret Hamilton, and Jan Lönnberg. Analysis of research into the teaching and learning of programming. In Proceedings of the fifth international workshop on Computing education research workshop, pages 93–104, 2009.

[188] Paul B Sheatsley. Questionnaire construction and item writing. Handbook of survey research, 4(1):195–230, 1983.

[189] Kim Bartel Sheehan. E-mail survey response rates: A review. Journal of computer-mediated communication, 6(2):JCMC621, 2001.

[190] Shuhaida Shuhidan, Margaret Hamilton, and Daryl D'souza. A taxonomic study of novice programming summative assessment. In Proceedings of the Eleventh Australasian Conference on Computing Education-Volume 95, pages 147–156. Citeseer, 2009.

[191] Valerie J Shute, Chen Sun, and Jodi Asbell-Clarke. Demystifying computational thinking. Educational research review, 22:142–158, 2017.

[192] Kassu Jilcha Sileyew. Research design and methodology. IntechOpen Rijeka, 2019.

[193] Tze Ying Sim and Sian Lun Lau. Online tools to support novice programming: A systematic review. In 2018 IEEE Conference on e-Learning, e-Management and e-Services (IC3e), pages 91–96. IEEE, 2018.

[194] Beth Simon, Michael Kohanfars, Jeff Lee, Karen Tamayo, and Quintin Cutts. Experience report: peer instruction in introductory computing. In Proceedings of the 41st ACM technical symposium on Computer science education, pages 341–345, 2010.

[195] Traci Sitzmann, Kurt Kraiger, David Stewart, and Robert Wisher. The comparative effectiveness of web-based and classroom instruction: A meta-analysis. Personnel psychology, 59(3):623–664, 2006.

[196] Kenneth Slonneger and Barry L Kurtz. Formal syntax and semantics of programming languages, volume 340. Addison-Wesley Reading, 1995.

[197] Bryan J Smith. Conceptual graphs as a visual programming language for teaching programming. In Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing, pages 258–259, 2009.

[198] C Estelle Smith, Kylee Shiekh, Hayden Cooreman, Sharfi Rahman, Yifei Zhu, Md Kamrul Siam, Michael Ivanitskiy, Ahmed M Ahmed, Michael Hallinan, Alexander Grisak, et al. Early adoption of generative artificial intelligence in computing education: Emergent student use cases and perspectives in 2023. In Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1, pages 3–9. 2024.

[199] Hannah Snyder. Literature review as a research methodology: An overview and guidelines. Journal of Business Research, 104:333–339, 2019.

[200] Elliot Soloway and James C Spohrer. Studying the novice programmer. Psychology Press, 2013.

[201] Juha Sorva, Ville Karavirta, and Lauri Malmi. A review of generic program visualization systems for introductory programming education. ACM Transactions on Computing Education, 13(4):1–64, 2013.

[202] Daniel L Stufflebeam. Evaluation checklists: Practical tools for guiding and judging evaluations. American Journal of Evaluation, 22(1):71–79, 2001.

[203] Phit-Huan Tan, Choo-Yee Ting, and Siew-Woei Ling. Learning difficulties in programming courses: undergraduates' perspective and perception. In Proceedings of the International Conference on Computer Technology and Development, pages 42–46, 2009.

[204] Sigmar-Olaf Tergan. Checklists for the evaluation of educational software: Critical review and prospects. Innovations in education and training international, 35(1):9–20, 1998.

[205] Gareth Terry, Nikki Hayfield, Victoria Clarke, Virginia Braun, et al. Thematic analysis. The SAGE handbook of qualitative research in psychology, 2(17-37):25, 2017.

[206] Nikolaos S Thomaidis, Nikitas Nikitakos, and Georgios D Dounias. The evaluation of information technology projects: A fuzzy multicriteria decision-making approach. International Journal of Information Technology & Decision Making, 5(01):89–122, 2006.

[207] Martin Mabeifam Ujakpa, Delene Heukelman, Victoria Kaleinasho Lazarus, Petsy Neiss, and GD Rukanda. Using whatsapp to support communication in teaching and learning. In 2018 IST-Africa Week Conference (IST-Africa), pages Page–1. IEEE, 2018.

[208] Martine Van Selm and Nicholas W Jankowski. Conducting online surveys. Quality and quantity, 40:435–456, 2006.

[209] Arto Vihavainen, Jonne Airaksinen, and Christopher Watson. A systematic review of approaches for teaching introductory programming and their influence on success. In Proceedings of the tenth annual conference on International computing education research, pages 19–26, 2014.

[210] VS Code. VS Code, 2023. `https://code.visualstudio.com/`.

[211] w3schools. w3schools, 2023. `https://www.w3schools.com/`.

[212] Feng Wang and Michael J Hannafin. Design-based research and technology-enhanced learning environments. Educational technology research and development, 53(4):5–23, 2005.

[213] Xuefeng Wei, Lin Lin, Nanxi Meng, Wei Tan, Siu-Cheung Kong, et al. The effectiveness of partial pair programming on elementary school students' computational thinking skills and self-efficacy. Computers & education, 160:104023, 2021.

[214] David Weintrop. Block-based programming in computer science education. Communications of the ACM, 62(8):22–25, 2019.

[215] David Weintrop and Uri Wilensky. To block or not to block, that is the question: students' perceptions of blocks-based programming. In Proceedings of the 14th international conference on interaction design and children, pages 199–208, 2015.

[216] Jacqueline Whalley, Raymond Lister, Errol Thompson, Tony Clear, Phil Robbins, PK Ajith Kumar, and Christine Prasad. An australasian study of reading and comprehension skills in novice programmers, using the bloom and solo taxonomies. 2006.

[217] Laurie Williams, Robert R Kessler, Ward Cunningham, and Ron Jeffries. Strengthening the case for pair programming. IEEE software, 17(4):19–25, 2000.

[218] Jeannette M Wing. Computational thinking. Communications of the ACM, 49(3):33–35, 2006.

[219] Jeannette M Wing. Computational thinking and thinking about computing. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366(1881):3717–3725, 2008.

[220] Kevin B Wright. Researching internet-based populations: Advantages and disadvantages of online survey research, online questionnaire authoring software packages, and web survey services. Journal of computer-mediated communication, 10(3):JCMC1034, 2005.

[221] Chih-Hung Wu and Wen-Chang Fang. Combining the fuzzy analytic hierarchy process and the fuzzy delphi method for developing critical competences of electronic commerce professional managers. Quality & Quantity, 45:751–768, 2011.

[222] Nick Wu. Trylinks: an interactive online platform to learn the links programming language. Technical report, University of Edinburgh, 2018. Honours project report.

[223] Stelios Xinogalos. Using flowchart-based programming environments for simplifying programming and software engineering processes. In Proceedings of the IEEE Global Engineering Education Conference, pages 1313–1322, 2013.

[224] Awad A Younis, Rajshekhar Sunderraman, Mike Metzler, and Anu G Bourgeois. Developing parallel programming and soft skills: A project based learning approach. Journal of Parallel and Distributed Computing, 158:151–163, 2021.

[225] Sen-Chi Yu. Comparison of internet-based and paper-based questionnaires in taiwan using multisample invariance approach. Cyberpsychology & behavior : the impact of the Internet, multimedia and virtual reality on behavior and society, 2007.

[226] Norhanisha Yusof, Nor Laily Hashim, and Azham Hussain. A review of fuzzy delphi method application in human-computer interaction studies. In AIP Conference Proceedings, volume 2472. AIP Publishing, 2022.

[227] Karimah Mohd Yusoff, Noraidah Sahari Ashaari, Tengku Siti Meriam Tengku Wook, and Noorazean Mohd Ali. Validation of the components and elements of computational thinking for teaching and learning programming using the fuzzy delphi method. International Journal of Advanced Computer Science and Applications, 12(1), 2021.

[228] Shuofei Zhu, Ziyi Zhang, Boqin Qin, Aiping Xiong, and Linhai Song. Learning and programming challenges of rust: A mixed-methods study. In Proceedings of the 44th International Conference on Software Engineering, pages 1269–1281, 2022.

[229] Hans-Jürgen Zimmermann. Fuzzy set theory—and its applications. Springer Science & Business Media, 2011.

[230] IS Zinovieva, VO Artemchuk, Anna V Iatsyshyn, OO Popov, VO Kovach, Andrii V Iatsyshyn, YO Romanenko, and OV Radchenko. The use of online coding platforms as additional distance tools in programming education. In Journal of physics: Conference series, volume 1840, page 012029. IOP Publishing, 2021.