

Interactive Learning on the Go: Managing Student-Centered Programming Education through Mobile Applications

Omobola Gambo
*Department of Arts & Social Science
Education, Lead City University,
Ibadan, Nigeria*
gambo.omobola@lcu.edu.ng

Odianose John Abuya
*School of Education and Sociology,
Faculty of Humanities and Social
Science, University of Portsmouth, UK.*
Odiajohn87@gmail.com

Adedapo Adelokun
*Department of Educational
Management,
Obafemi Awolowo University, Ile-Ife*
dapoadelokun@gmail.com

Feyisayo Damilola Faloye
*Department of Computer Science &
Engineering, Obafemi Awolowo
University Ile-Ife, Nigeria.*
fdfaloye@student.oauife.edu.ng

Wasiu Olanrewaju-Smart
*Harvard University,
Cambridge, MA, USA*
wasiusmart@gmail.com

Christopher Agbonkhese
*Department of Digital and
Computational Studies, Bates College,
Lewiston, ME 04240, USA.*
cagbonkhese@bates.edu

Abstract— This study addresses the limitations of traditional programming education by developing a mobile application designed for undergraduate students learning programming languages. Conventional teaching methods often fail to engage students and meet the diverse learning needs of novice programmers. The project aims to create a mobile application that integrates interactive coding exercises, real-time feedback, video tutorials, and collaborative tools to improve engagement and learning outcomes. The research focuses on designing an educational technology application that offers personalized, interactive, and accessible learning experiences. A structured methodology was employed, including an extensive literature review, requirement analysis, and the development of the mobile application using Android Studio, Flutter, and Firebase. The application was tested for usability, engagement, and effectiveness through pilot studies and user feedback. The results demonstrate a significant improvement in student engagement, reduced learning difficulties, and better comprehension of programming concepts through real-time feedback and interactive modules. The flexible design of the application allows for updates and customization, making it adaptable to different educational settings and programming languages. This study contributes to educational technology by providing a practical solution to challenges in programming education. The mobile application offers a holistic approach to learning, integrating interactive features and personalized learning paths to enhance the overall student experience. The findings emphasize the potential of mobile learning in improving educational outcomes, particularly in fostering engagement and addressing the needs of diverse learners.

Keywords— *Mobile Learning, Programming Education, Interactive Learning, Educational Technology, Student Engagement, Personalized Learning Paths.*

I. INTRODUCTION

The integration of mobile devices in education has significantly evolved over the past decade, transforming from simple applications to complex, interactive learning platforms [1]. Mobile learning has emerged as a critical component in modern educational strategies, particularly in the teaching and learning of technical subjects like

programming. The increasing demand for skilled programmers has necessitated innovative approaches to programming education, with institutions and educators actively exploring methods to enhance the accessibility and effectiveness of programming instruction [2].

Traditional programming education methods, such as lectures, textbooks, and individual assignments, often fail to fully engage students, particularly those with varying learning preferences [3]. Novice programmers, in particular, struggle with abstract concepts and practical application, highlighting the limitations of conventional teaching techniques that are not tailored to diverse learning needs [4]. Consequently, mobile devices offer a unique opportunity to create personalized and interactive learning experiences that can better support these learners [5].

The widespread adoption of smartphones and tablets has revolutionized how students access and engage with educational content. Mobile applications have proven effective in various educational contexts, such as language learning, as demonstrated by Katijah et al. [6], and are now being applied to programming education. The global shift toward online and mobile learning, especially post-COVID-19, underscores the importance of developing tailored educational tools that can address challenges such as the digital divide and unequal access to resources [7, 8].

This research examines the convergence of education and technology through the development of a mobile application specifically designed for programming education. By leveraging the benefits of mobile learning, the study seeks to address the critical gaps in traditional teaching methods and contribute to the ongoing discourse on innovative educational practices in programming.

Notably, this paper contributes to the fields of educational technology and programming education by developing an innovative mobile application that enhances the learning experience for novice programming students. By integrating real-time coding exercises, interactive tutorials, and collaborative tools, this work addresses key challenges in

traditional programming education and provides a scalable, effective solution for university settings. Thus, the specific contribution of this paper is four-fold:

i. Systematically, the research aims to create a framework for integrating educational technology into programming instruction. This framework will help us understand how mobile learning tools can support programming education and identify best practices for designing apps for programming learners.

ii. Methodologically, the study will develop and validate new methods for assessing the effectiveness of mobile applications in education. This involves creating evaluation techniques to measure the impact of the app on students' learning experiences and collecting user feedback to improve the app.

iii. Practically, the mobile app will provide interactive coding exercises, video tutorials, and real-time feedback for undergraduate students. It aims to enhance students' learning experiences and address common challenges in programming education.

iv. Analytically, the research will assess the impact of the mobile app on students' programming skills and engagement, providing insights into the effectiveness of educational tools. Additionally, the study will explore the challenges faced by students while using the app and offer recommendations for improvements.

The remainder of this paper is structured as follows: Section 2 reviews existing literature on mobile learning and application, discussing the strengths and limitations of previous approaches. Next, in Section 3, the paper describes the overall methodology used. Section 4 presents the results and discussion, and system evaluation approach. Section 5 presents the findings of the study and Section 6 presents the recommendation for future work. Finally, Section 7 presents the conclusion of the study.

II. RELATED WORKS

Mobile learning (m-learning) has emerged as a transformative response to the increasing availability and accessibility of mobile devices, representing a significant shift in educational paradigms [9]. Extensive research into this concept underscores its potential to enhance learning outcomes across diverse educational contexts [10]. The initial integration of mobile devices into existing learning management systems (LMS) facilitated a more flexible and accessible learning environment, effectively transforming traditional educational frameworks [11]. However, while many studies highlight the advantages of m-learning, they often overlook specific strategies for effective integration within established curricula, suggesting a need for a more comprehensive approach.

Research indicates that m-learning activities grounded in constructivist learning principles can lead to notable improvements in student achievement. For instance, a study in a History of Art course demonstrated enhanced learning outcomes, particularly among low achievers, through the utilization of platforms such as Pinterest and Piazza [12]. Nevertheless, despite these promising findings, there remains a lack of research exploring the long-term impacts of such platforms on student engagement and retention, indicating a gap that this study seeks to address. Additionally, investigations into social learning applications have revealed

their efficacy in fostering behavioral, social, cognitive, and emotional engagement, often surpassing traditional teaching methods [13]. A systematic review focused on user interface design emphasized the importance of cognitive load considerations and identified significant gaps in the development of effective mobile learning applications [14]. These insights suggest that thoughtful design and user-centered approaches are essential for the success of m-learning tools, an area where further investigation is warranted.

The application of m-learning extends to various educational settings, including corporate training and professional development. Research indicates that m-learning in the corporate sector enhances accessibility and flexibility, but it also underscores the necessity of considering contextual factors such as usefulness, ease of use, and individual learning styles [15]. While usability studies highlight the importance of well-designed m-learning applications for positive learning experiences, there remains limited exploration into how these applications can be tailored to diverse organizational cultures and employee needs.

In the context of refugee education, m-learning has been recognized as a valuable tool for addressing diverse educational needs. A systematic literature review identified key characteristics of mobile learning applications tailored for refugees, emphasizing the integration of psychological and educational features alongside cultural considerations [16]. However, the review lacks an in-depth analysis of the specific challenges faced by refugee learners, presenting an opportunity for further research to identify and address these barriers.

Usmonova [10] proposed a conceptual framework for m-learning that focuses on the integration of applied computing, information systems, and the theory of computation. This framework aims to enhance educational quality through the active utilization of information and communication technologies (ICT). Despite its innovative approach, the framework does not sufficiently address the practical implementation challenges faced by educators, indicating a critical area for future inquiry.

The evolution of m-learning has also been influenced by various design guidelines and principles, including Nielsen's design guidelines and signaling principles. These guidelines are vital for developing effective and engaging mobile learning applications, as highlighted by Rosmani et al. [17]. However, the existing literature often fails to link these principles to specific case studies, leaving a gap in understanding their real-world applicability. By adhering to these principles, educators and developers can create m-learning environments that significantly enhance the learning experience, which this study aims to explore further.

Numerous studies have contributed to the development of mobile application tools designed to enhance undergraduate students' learning experiences in programming languages. These studies have illustrated the positive impact of mobile learning (m-learning) on student engagement and academic outcomes. For instance, Hikmah et al. [18] demonstrated that utilizing WhatsApp as a learning tool for historical sites significantly increased student interest and engagement. However, while this study highlights the effectiveness of mobile communication tools, it lacks a focus on structured

programming education. This gap suggests that further investigation is needed to explore how similar tools can be effectively employed in teaching programming languages.

Coelho et al. [19] conducted a systematic literature review on m-learning tools specifically aimed at teaching programming logic and design. They identified twelve mobile applications utilized in university settings, noting their positive effects on student learning processes, engagement, motivation, and retention. While their findings offer valuable insights, the review does not delve into specific user experiences or the design features that contribute to these positive outcomes. Addressing this gap, the present study aims to investigate user interactions and feedback on the design and usability of programming mobile applications.

In their experience-based study, Ortiz et al. [20] developed an m-learning application focused on teaching control structures in structured programming languages. This application enabled teachers to monitor student progress and provide immediate feedback, thus enhancing the overall learning experience. Although the study underscores the importance of feedback in programming education, it does not explore the long-term retention of knowledge or the scalability of the application in broader educational contexts. The current research will expand on these findings by evaluating the application's long-term effectiveness in different educational settings.

Matthews et al. [21] explored the effectiveness of multimedia learning objects in teaching introductory programming languages. Their study revealed that multimedia tools can facilitate better knowledge transfer and help novice learners comprehend essential programming paradigms, syntax, and logic. However, the research does not address the integration of these multimedia elements into mobile applications. This study will fill this gap by examining how multimedia learning objects can be incorporated into mobile platforms to enhance the learning experience.

Falode et al. [22] developed an interactive mobile application for teaching educational technology concepts using the ADDIE instructional design model. Built with JavaScript, HTML, and CSS, the application demonstrated improved academic achievement among students and was validated through expert reviews and user testing. While this study provides evidence of the effectiveness of interactive design in educational tools, it lacks a focus on programming languages specifically. The present study will apply similar design principles to programming education, ensuring that the tools are tailored to address the unique challenges faced by programming students.

Additionally, while not exclusively focused on programming languages, the Technovation Challenge offers a model for hands-on, project-based learning, enabling high school girls to learn programming and entrepreneurship [23]. This model could be adapted for undergraduate students, emphasizing practical implementation and feedback from industry professionals. The current research aims to explore how such project-based learning approaches can be effectively integrated into mobile applications for programming education.

Finally, Patil et al. [24] highlighted the importance of incorporating active learning strategies and visualization tools to maintain student interest in programming courses. Their research points to the effectiveness of various editors

and tools for writing and executing programs, which can be effectively delivered through mobile applications. The present study will investigate how these active learning approaches can be specifically implemented in mobile applications designed for programming instruction.

Overall, the existing literature underscores the benefits of mobile learning, multimedia learning objects, and active learning strategies in education. However, gaps remain in the context of programming language instruction, particularly regarding user experiences, long-term effectiveness, and the integration of interactive design elements. This study aims to address these gaps by focusing on the design, implementation, and validation of mobile application tools tailored for enhancing the learning experiences of undergraduate programming students.

III. GAPS IN LITERATURE

The existing literature on m-learning for programming education reveals significant gaps that hinder the optimization and widespread adoption of mobile learning technologies. One prominent area of concern is the limited focus on specific programming languages. Most research has concentrated on general programming concepts rather than tailored applications for specific languages like Python, Java, or C++. This lack of attention to language-specific nuances restricts the development of effective mobile tools that can address the unique learning needs associated with different programming paradigms, such as object-oriented and functional programming.

Another critical gap lies in the methodological approaches utilized in existing studies. Many rely predominantly on questionnaires and surveys, leading to a lack of methodological diversity. This limitation prevents a comprehensive understanding of the effectiveness of mobile learning tools, as mixed-methods and experimental designs could yield richer insights. Furthermore, the scarcity of longitudinal studies that track students' learning progress over extended periods raises questions about the long-term impact of m-learning tools on programming skills. Such research is essential for understanding the sustained effects of these technologies on learners.

Additionally, the geographical and cultural contexts of mobile learning research are often narrow, predominantly focusing on Western or Asian regions. This oversight limits the applicability of findings in diverse cultural settings, particularly in regions like Africa and Latin America, where educational needs and contexts may differ significantly. Moreover, the literature lacks adequate discussion on the cultural adaptation of mobile learning tools, suggesting that tools designed for one cultural group may not resonate with others. Lastly, there is a dearth of research investigating the integration of emerging technologies, such as AI and machine learning, into m-learning for programming education. Exploring these integrations could significantly enhance personalized learning experiences and improve overall educational outcomes. Addressing these gaps is vital for fostering the effectiveness and acceptance of mobile learning technologies in programming education.

IV. METHODOLOGY

This section outlines the systematic approach employed in developing the mobile application tool designed to enhance the learning experience for undergraduate students studying programming languages. The methodology encompasses several critical phases: requirements gathering and analysis, user-centered design, system design specifications, workflow design, and interface design.

Initially, extensive research was conducted to review relevant literature on programming education, mobile applications, and the integration of educational technology. This comprehensive review served as the theoretical foundation for the study, helping to identify the challenges faced by novice programmers and the existing solutions available. The insights gained from this literature informed the requirement analysis for designing and developing a mobile application tailored to address the unique needs of beginner programming students.

Following the literature review, the design and implementation phase commenced, focusing on the architecture of the mobile application. The design outlined key components, including the user interface, interactive coding environments, real-time feedback systems, and collaborative tools. During implementation, software development tools such as Android Studio, Flutter, and Firebase were utilized to construct the application. Particular attention was given to developing the coding exercise module, tutorial system, and peer collaboration features, emphasizing user interaction and the integration of real-time feedback. The application was designed to support multiple programming languages, with a backend database hosted on Firebase to manage user data, track progress, and deliver personalized feedback.

Subsequently, the system underwent rigorous testing to ensure it met the intended objectives. Usability testing was conducted with a sample of university students who evaluated the application's features, interface, and overall user experience. This testing phase validated the application's functionality, its capacity to enhance learning through interactive exercises, and the effectiveness of real-time feedback in improving student comprehension. User feedback played a crucial role in informing further refinements to the application, ensuring it was not only efficient but also user-friendly and engaging.

A. Requirements Gathering and Analysis

The initial phase involves defining the necessary operational behaviors of the system, commonly referred to as functional requirements, as well as identifying the essential software and hardware environment requirements for optimal system performance. The functional requirements articulate the intended behavior of the application, specifying the functions the system and its components must perform. The key functional requirements include, User Authentication, Access to Learning Content, Interactive Coding Environment, Collaboration and Community Tools, Progress Tracking, Content Management, and Gamified Learning.

To ensure comprehensive requirements gathering, multiple channels were employed, including user feedback,

market analysis, and a review of existing scholarly literature. This approach guarantees a thorough understanding of user needs and expectations.

User feedback was collected through surveys and interviews with potential users, who are students and lecturers, providing valuable insights into their preferences and pain points. Market analysis involved examining existing programming language learning applications to identify trends and areas for differentiation. The literature review offered insights into user behavior and best practices in user experience (UX) and user interface (UI) design.

Subsequently, gathered requirements were prioritized based on user value, technical feasibility, and strategic importance. A feasibility study assessed the technical, operational, and economic viability of the requirements to ensure their implementation within project constraints. A use-case diagram was employed to visualize user interactions and system responses, identifying actors (e.g., students, system administrators) and their interactions with the system.

All requirements were meticulously documented to create a blueprint for developers and designers. This documentation included both functional requirements, detailing what the system should accomplish, and non-functional requirements, outlining performance and usability standards. Fig. 1 presents the snippet of data insight gathered from participants, who are students between the age of 18 and 30.

B. User-Centered Design Approach

The User-Centered Design (UCD) approach is integral to the development of the mobile application, ensuring that the solutions align with the real needs of users. This method emphasizes the importance of involving users throughout the development process, thereby enhancing user satisfaction—a crucial factor for the success of any mobile application.

The UCD process adheres to foundational principles, including early focus on users and tasks, iterative design, and empirical measurement. User interface flow diagrams are utilized to illustrate navigation paths, optimizing user interactions by identifying potential usability issues. Rigorous user testing, including usability testing and A/B testing, is employed to gather feedback on the prototypes, ensuring alignment with user expectations.

C. System Design Specifications

In the system design specifications phase, the architecture of the application is articulated, detailing how the components interact with one another and with users. Utilizing Unified Modeling Language (UML) diagrams, the system architecture is visualized, providing clarity on the application's structure and processes.

Fig. 2 shows the flow chart diagram indicating the user interactions, detailing the progression from user registration to engaging with various features such as interactive coding exercises and video tutorials. Fig. 3 shows the use-case diagram that outlines interactions from the user's perspective, ensuring all necessary services are included.

Age	Gender	Level of Study	Field of Study	How currently learn programming	Which programming languages are you learning	What are the most challenging topics in your programming	How often do you practice coding	Do you use any mobile applications for learning	What features would you like to see in a mobile application for learning	How important is it for the mobile application to have these features	Would you like to use a mobile application for learning	
18-20	Female	400 Level	Computer Science and Engineering	Mobile applications	Java	Algorithms	About once a week	Yes	Collaboration tools	Not at all important	Undecided	Somewhat likely
21-23	Male	300 Level	Computer Science and Engineering	Textbooks	Java	Object-Oriented Programming	Daily	Yes	Collaboration tools	Not at all important	Yes	Not at all likely
24-26	Male	500 Level	Electrical Engineering	Online tutorials	Python	Object-Oriented Programming	Less than once a week	No	Video tutorials	Somewhat important	Undecided	Very likely
24-26	Female	500 Level	Electrical Engineering	Textbooks	JavaScript	Web development	Several times a week	Yes	Collaboration tools	Very important	Undecided	Somewhat likely
18-20	Male	200 Level	Mechanical Engineering	Mobile applications	Python	Object-Oriented Programming	Daily	No	Real-time feedback on code	Somewhat important	Undecided	Not at all likely
21-23	Male	200 Level	Mechanical Engineering	Textbooks	Python	Data structures	Daily	Yes	Video tutorials	Not very important	No	Somewhat likely
24-26	Female	400 Level	Electrical Engineering	Classroom instruction	Python	Data structures	Rarely	No	Gamification elements	Somewhat important	No	Not at all likely
21-23	Male	200 Level	Computer Science and Engineering	Mobile applications	Python	Algorithms	Daily	Yes	Offline access	Somewhat important	Undecided	Somewhat likely
18-20	Male	500 Level	Computer Science and Engineering	Textbooks	JavaScript	Web development	Less than once a week	Yes	Interactive coding exercises	Very important	No	Not at all likely
21-23	Male	200 Level	Computer Science and Engineering	Online forums	Python	Object-Oriented Programming	Rarely	Yes	Collaboration tools	Not very important	No	Somewhat likely
24-26	Female	300 Level	Mechanical Engineering	Mobile applications	Python	Object-Oriented Programming	About once a week	Yes	Interactive coding exercises	Very important	Undecided	Very likely
24-26	Female	100 Level	Mechanical Engineering	Textbooks	Python	Web development	Daily	No	Gamification elements	Very important	No	Somewhat likely
24-26	Female	200 Level	Computer Science and Engineering	Online forums	Python	Data structures	Daily	Yes	Offline access	Very important	No	Somewhat likely
24-26	Male	100 Level	Computer Science and Engineering	Classroom instruction	Java	Algorithms	Rarely	No	Real-time feedback on code	Very important	Yes	Not very likely
18-20	Male	500 Level	Chemical Engineering	Classroom instruction	C++	Object-Oriented Programming	Less than once a week	No	Collaboration tools	Very important	Yes	Somewhat likely
18-20	Male	100 Level	Computer Science and Engineering	Online forums	C++	Algorithms	Less than once a week	No	Interactive coding exercises	Somewhat important	No	Very likely
21-23	Male	300 Level	Computer Science and Engineering	Textbooks	Other	Database management	Less than once a week	Yes	Video tutorials	Very important	Yes	Somewhat likely
21-23	Female	200 Level	Computer Science and Engineering	Classroom instruction	Python	Object-Oriented Programming	Rarely	Yes	Collaboration tools	Not at all important	Yes	Not very likely
24-26	Male	400 Level	Electrical Engineering	Textbooks	Other	Data structures	Daily	No	Real-time feedback on code	Very important	No	Somewhat likely
24-26	Female	100 Level	Computer Science and Engineering	Online tutorials	C++	Data structures	Several times a week	Yes	Collaboration tools	Very important	No	Somewhat likely
21-23	Female	300 Level	Electrical Engineering	Online tutorials	JavaScript	Algorithms	Several times a week	No	Real-time feedback on code	Somewhat important	No	Not at all likely
18-20	Male	200 Level	Computer Science and Engineering	Mobile applications	Python	Data structures	Less than once a week	No	Gamification elements	Very important	Undecided	Somewhat likely
24-26	Female	100 Level	Computer Science and Engineering	Online forums	Python	Database management	About once a week	Yes	Collaboration tools	Not at all important	Yes	Not at all likely
24-26	Male	200 Level	Computer Science and Engineering	Classroom instruction	Python	Data structures	Less than once a week	No	Real-time feedback on code	Not very important	No	Not at all likely
24-26	Male	300 Level	Mechanical Engineering	Mobile applications	Other	Object-Oriented Programming	About once a week	Yes	Real-time feedback on code	Not at all important	Undecided	Somewhat likely
24-26	Female	200 Level	Electrical Engineering	Classroom instruction	Python	Database management	About once a week	No	Video tutorials	Not at all important	No	Somewhat likely
21-23	Male	100 Level	Electrical Engineering	Mobile applications	C++	Object-Oriented Programming	About once a week	Yes	Interactive coding exercises	Very important	Undecided	Very likely
21-23	Female	200 Level	Computer Science and Engineering	Online tutorials	Python	Database management	Less than once a week	No	Gamification elements	Not at all important	Undecided	Not at all likely
21-23	Male	200 Level	Computer Science and Engineering	Mobile applications	Python	Algorithms	Several times a week	Yes	Collaboration tools	Not at all important	No	Very likely
18-20	Male	200 Level	Electrical Engineering	Mobile applications	Python	Object-Oriented Programming	Several times a week	Yes	Gamification elements	Somewhat important	Undecided	Not very likely

Fig. 1. Snippet of Data insight gathered from participants

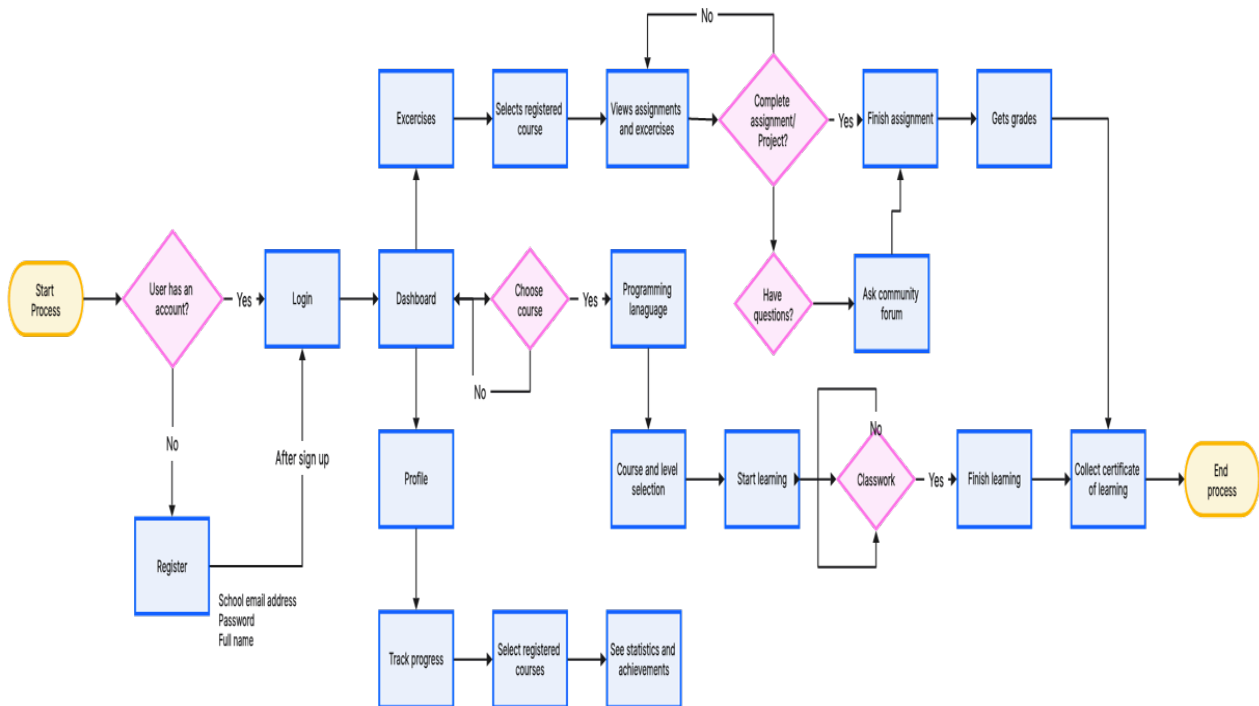


Fig. 2. Flow chart for the system

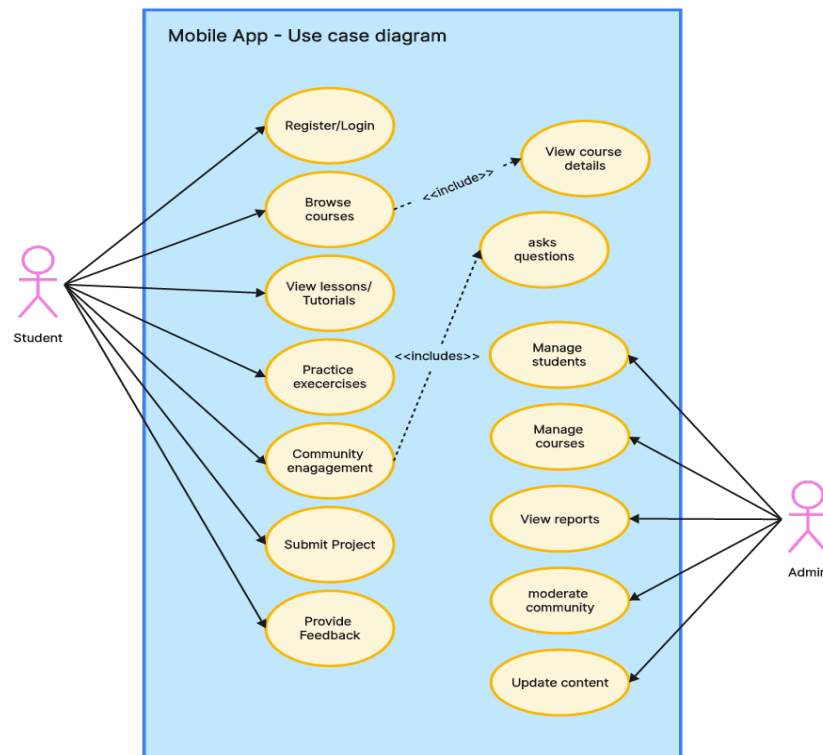


Fig. 3. Use case diagram of the mobile app

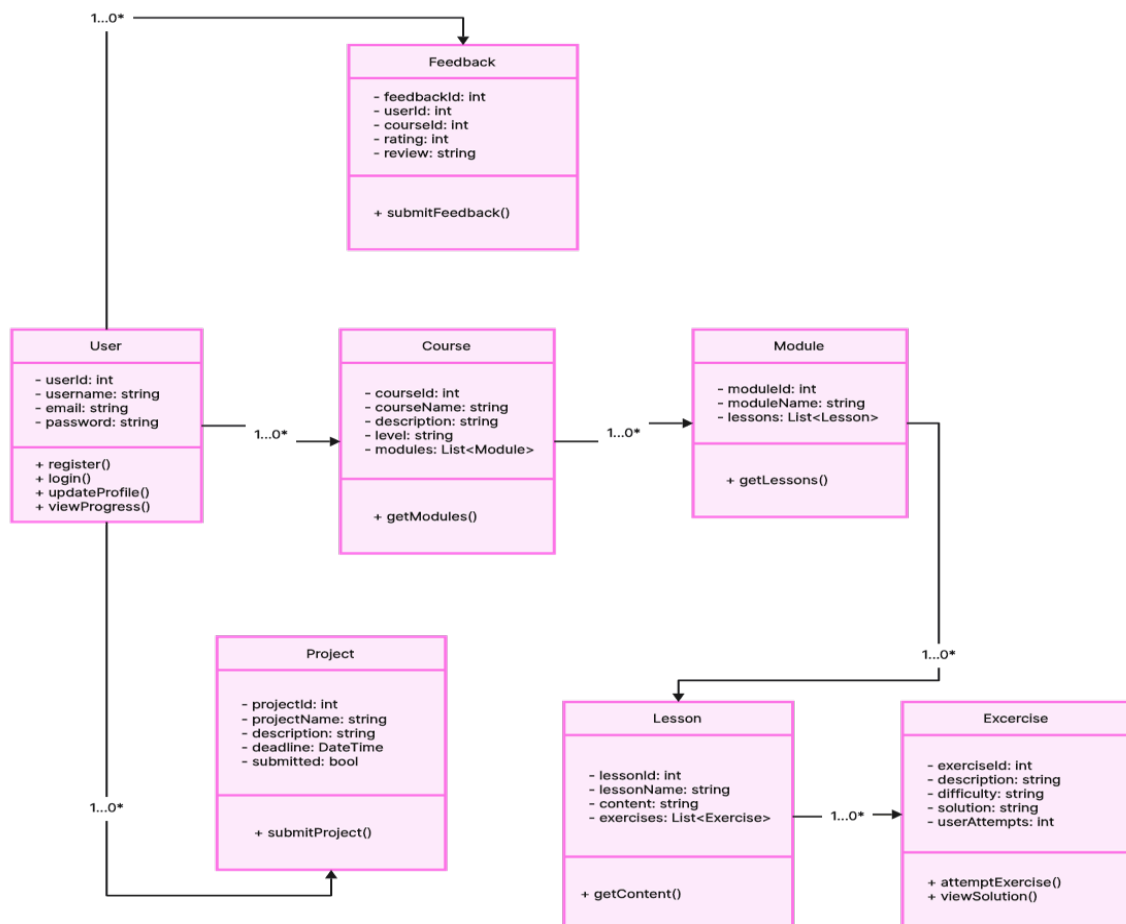


Fig. 4. Class diagram of the mobile app

Fig. 4 shows the class diagram, which is the static structure of the system, showcasing relationships between classes such as Course, User, Module, Feedback, Project, Lesson and Exercise, crucial for a well-structured database. Validation of the design specifications is conducted through expert reviews and user-centered testing. Feedback from experienced designers and users informs refinements, ensuring that the architecture meets technical and user requirements.

V. RESULT AND DISCUSSION

Given that the system involves user onboarding amongst other functional requirements, the graphical user interface (GUI) was designed to provide a seamless and user-friendly experience. Thus, the user onboarding process in this mobile

application begins with a user-friendly registration flow as shown in Fig. 5 where new users can create an account. During registration, users provide essential information, such as their name, email address, and preferred programming languages or areas of interest.

Once registered, the onboarding process gathers additional information related to the user's learning preferences and prior knowledge of programming, helping customize the user experience. The onboarding process familiarizes users with key features and culminates with a congratulatory pop-up animation, providing immediate positive feedback upon successfully completing the setup process.

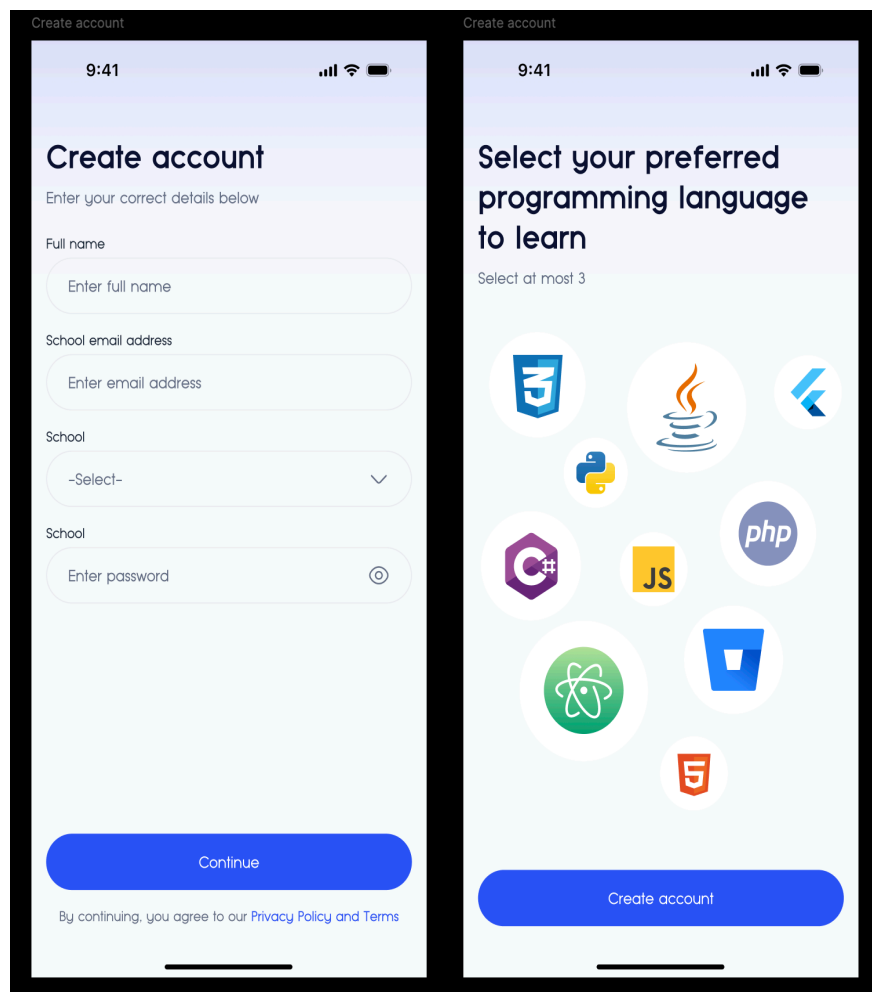


Fig. 5. User registration and onboarding

The mobile application includes a secure login screen where users authenticate using their registered email and password. Upon successful login, users can access the app's settings area to update their preferences, which were initially set during registration and onboarding. These preferences encompass learning styles, preferred programming languages, and course progress, enabling the app to continuously tailor content and exercises to the user's needs and skill level.

The settings menu also allows users to adjust their engagement preferences, such as the types of notifications they receive and the difficulty level of programming exercises. The login process is designed to be seamless and secure, providing clear feedback messages like "Login successful" or "Incorrect password" to assist users through any issues. As depicted in Fig. 6, this process ensures users can resume their programming education from where they left off, maintaining learning continuity.

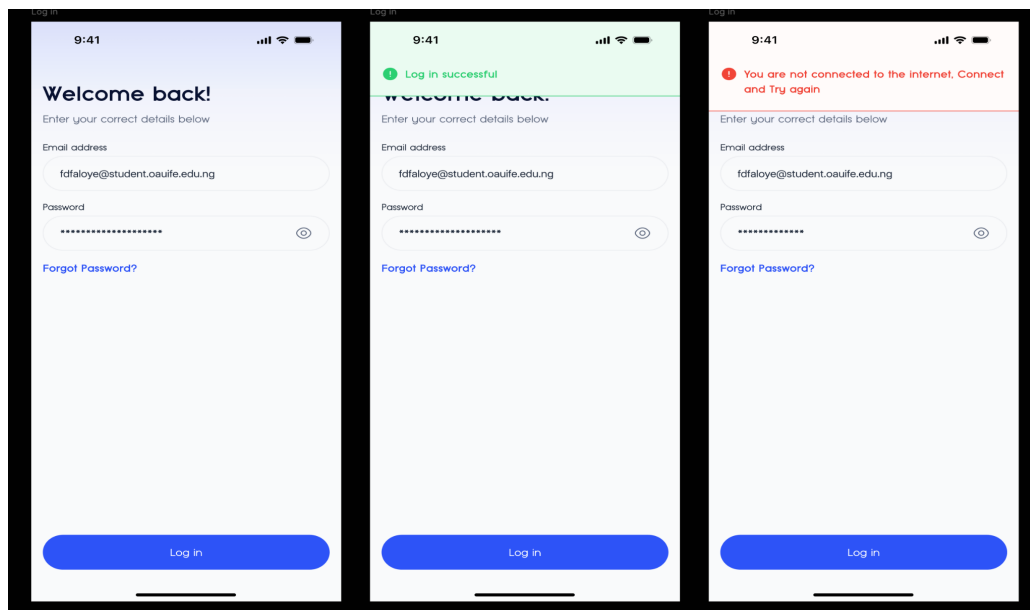


Fig. 6. Login with success and error feedback message.

User data, including preferences, progress, and settings, is securely stored in a cloud-based database like Firebase. Additionally, a local, encrypted copy of relevant data is maintained on the user's device for offline access and faster reference. This structure enhances the app's responsiveness while ensuring interactivity.

Regarding the learning and tutorials, access to learning content and module processes within the mobile application

begins after successful user authentication. Upon logging in as shown in Fig. 7, users are granted access to a tailored dashboard featuring an overview of their course progress, recommended courses based on their schools, and community threads. The app's structured pathway includes interactive lessons, coding exercises, and video tutorials designed to gradually build users' programming skills.

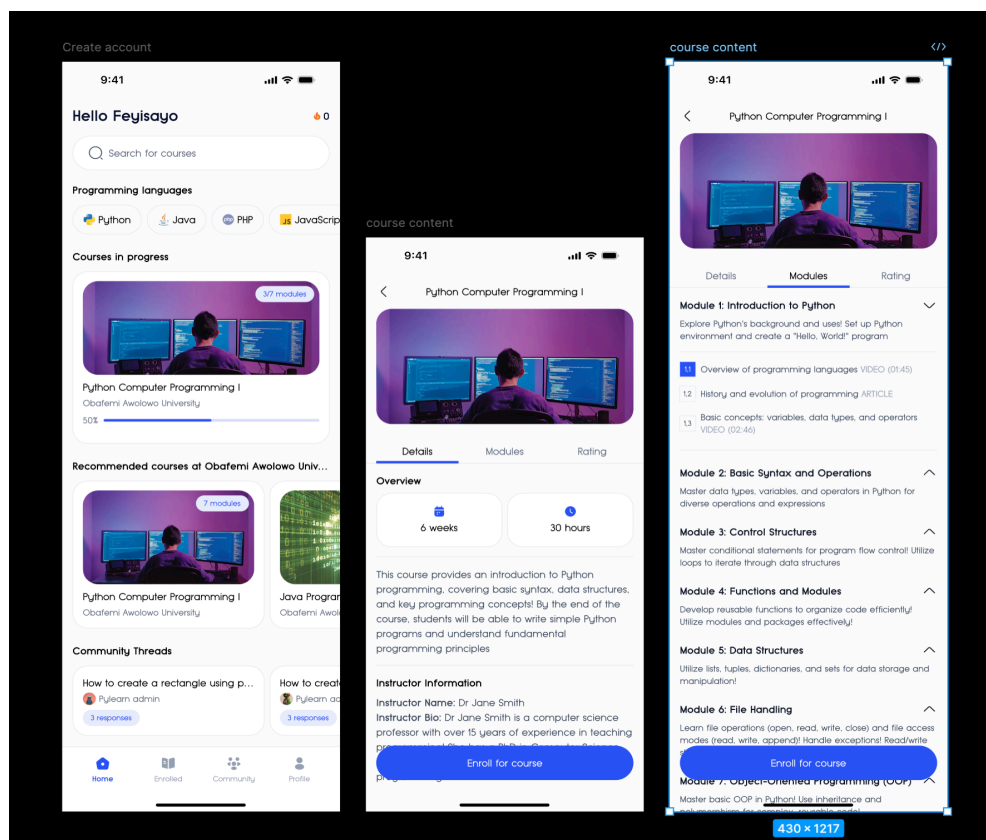


Fig. 7. Dashboard showing the course details

The learning process begins as shown in Fig. 8 with each module introducing core topics through multimedia content and practical examples. Users receive immediate evaluations on their coding submissions, enabling them to apply what they've learned effectively. As students' progress, the app adapts the difficulty level based on their performance, ensuring appropriate challenge levels. Furthermore, as shown

in Fig. 9 users are prompted to take assessments and receive feedback on their performance, with insights and recommendations provided. All user progress and results are securely stored in a cloud database, ensuring seamless learning continuity across devices and offline work.

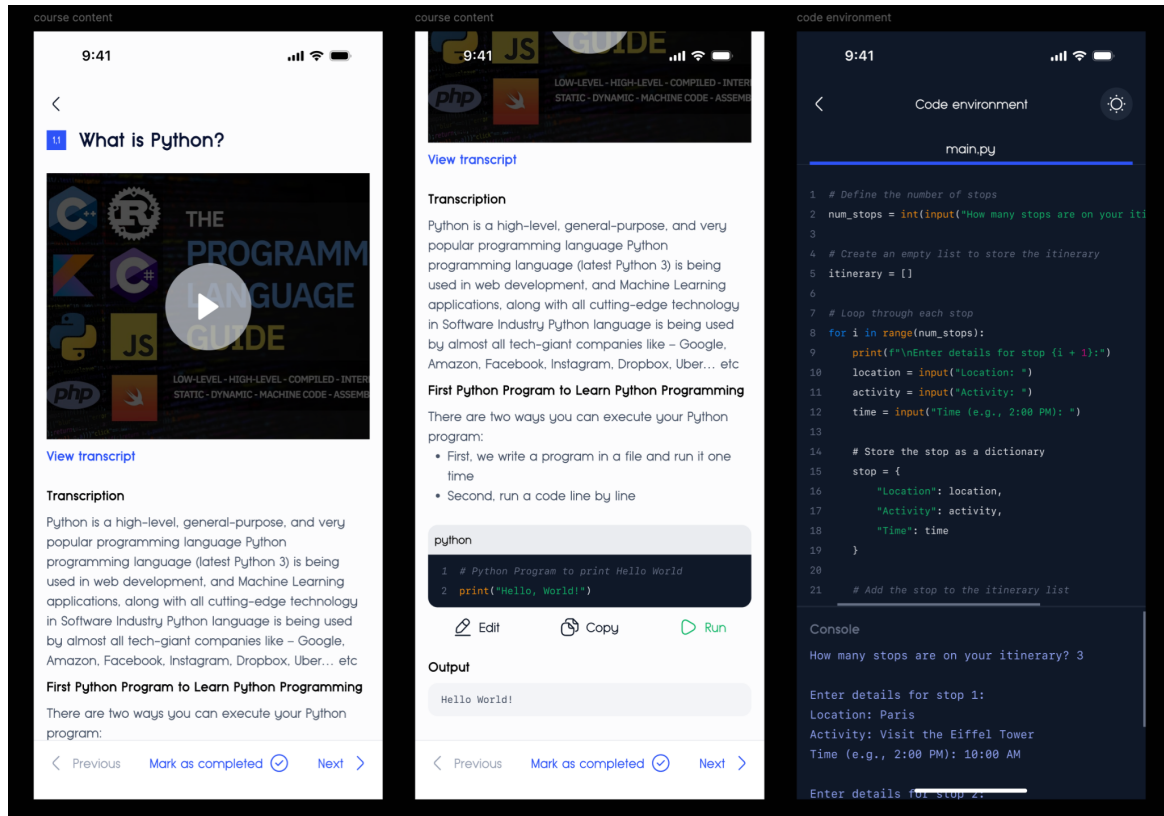


Fig. 8. Course module, content and a live coding environment

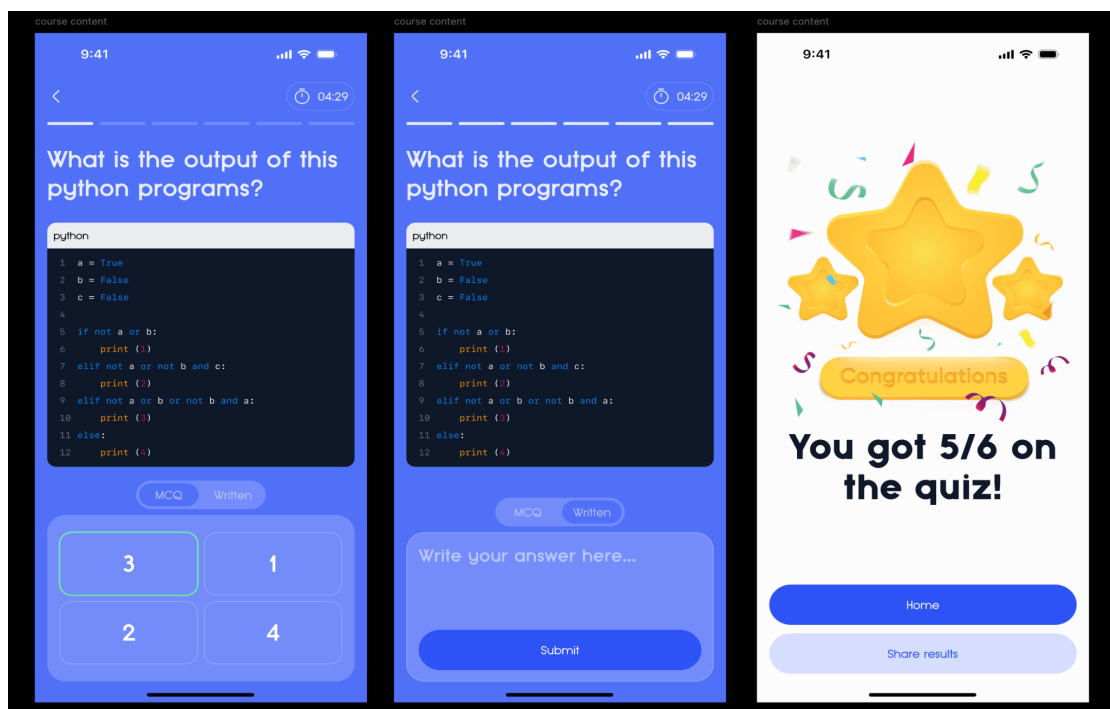


Fig. 9. Assessments, quiz and feedback

A. Statistics and Tracking Tool

The mobile application features a robust system for tracking student progress in learning programming languages, providing detailed insights into their development. Through an intuitive progress dashboard, students can view key metrics such as completed exercises, mastered concepts, and time spent coding. The app also incorporates streaks to encourage consistent practice, rewarding students for consecutive days of coding with badges and achievements. These streaks help students build daily habits essential for mastering programming, with reminders to maintain momentum.

In addition to tracking progress, the system as shown in Fig. 10 offers personalized insights, highlighting strengths and areas for improvement. It suggests exercises and resources based on individual performance, helping students focus on concepts that need reinforcement. The app also tracks progress across different programming languages, allowing students to see how their skills evolve over time. By enabling students to compare their progress with peers through community insights and leaderboards, the system fosters engagement, accountability, and motivation, creating an interactive and data-driven learning experience.

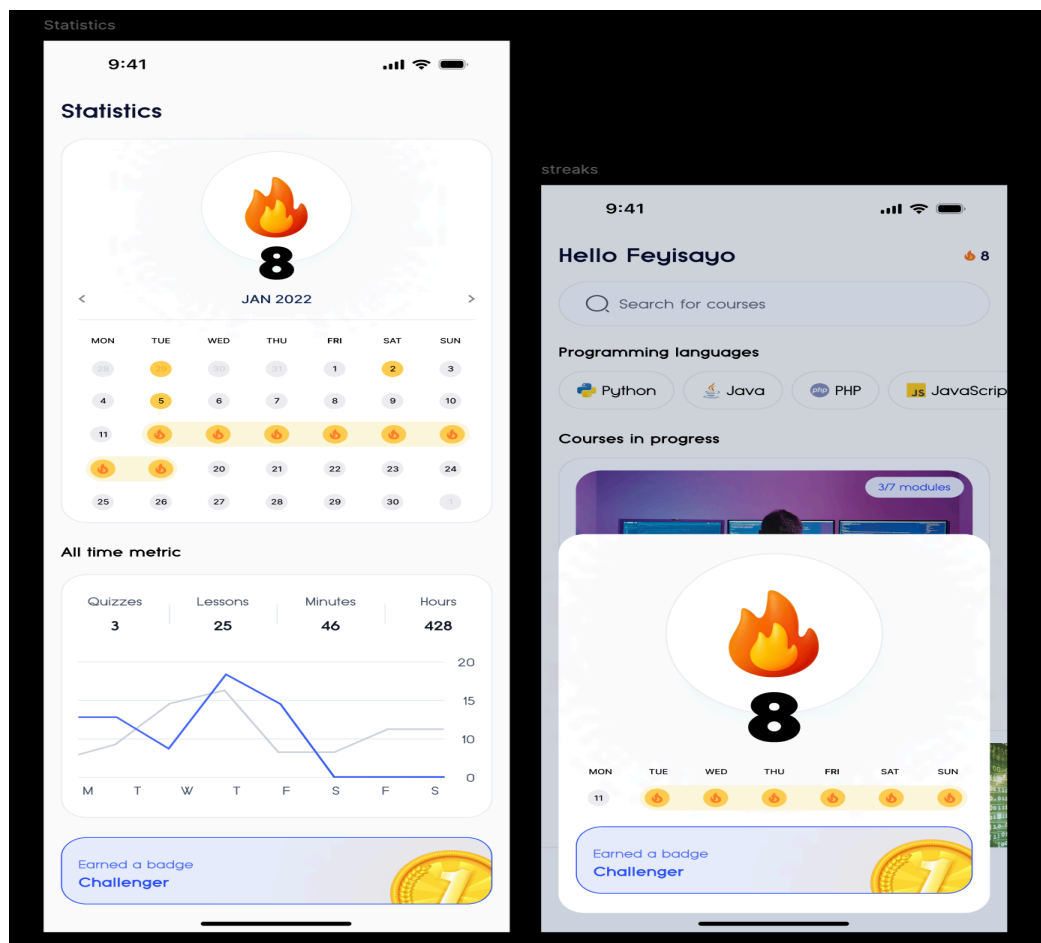


Fig. 10. Progress tracking

B. System Evaluation

This section outlines the usability testing conducted on the mobile application developed to teach programming languages to undergraduate students. A total of 7 participants were selected for the beta test of the app, representing the target demographic. These participants were asked to explore key features of the app, including user authentication, access to learning content, interactive coding, and community collaboration tools and also rate it out of 10. After interacting with the app, they were required to complete a feedback form aimed at evaluating the application's functionality, ease of

use, and overall user satisfaction. The result is shown in Fig. 11. Also, the snippet of usability testing results obtained from the 7 testers is shown in Fig. 12. As Fig. 12 reflects, the 7 testers provided feedback points, feedback summary, issues encountered while using the mobile application and the suggestions for improvement. The goal of the usability testing is to enhance the user experience, evaluate the effectiveness of the learning tool, identify some of the preferences and needs identify by the users, collect user feedback for iterative improvement, ensure accessibility, validate the design decisions made, and measure the learning outcomes.

From an educational management perspective, the usability testing of the mobile application designed to teach programming languages to undergraduate students serves several important functions, which include, (i) alignment with educational goals, (ii) focus on target audience

consideration, (iii) determination of quality assurance, (iv) feedback loop for educators, (v) enhancing student engagement, (vi) accessibility and inclusivity, and (vii) data-driven decision making.

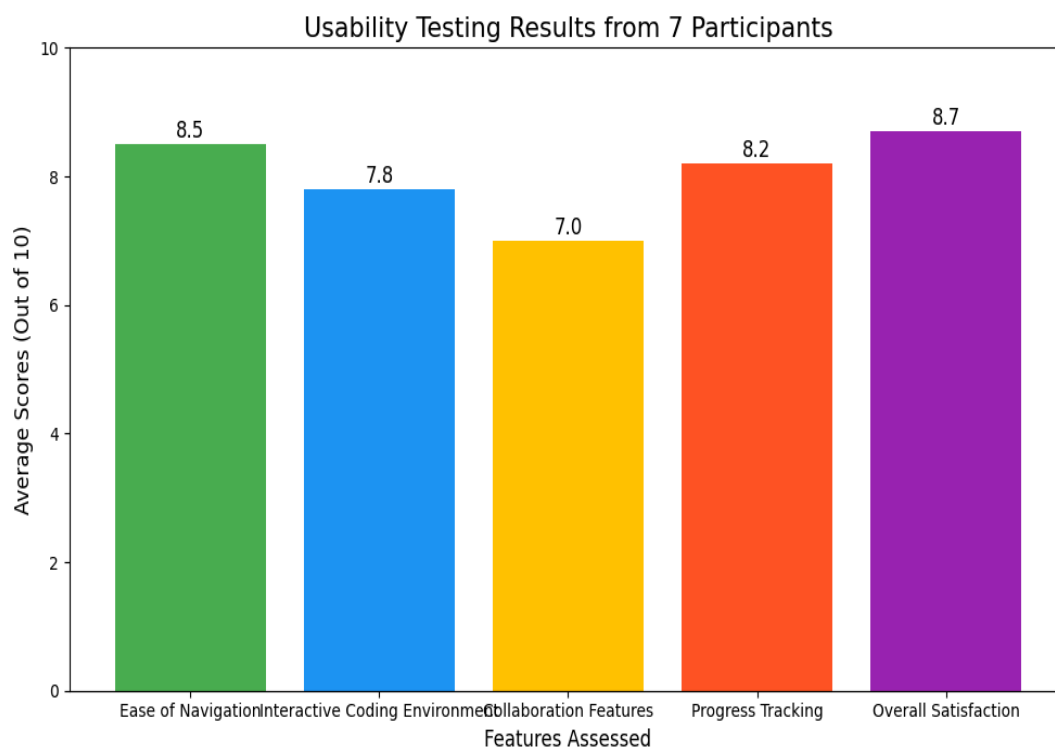


Fig. 11. Bar chart showing the usability tests results from the 7 participants.

Scenario Description	Objective	Feedback Points	Feedback Summary	Issues Encountered	Suggestions for Improvement
New User Onboarding and Initial Setup	Test user authentication and initial setup	Ease of registration, clarity of instructions, UI design	Registration was straightforward, but initial setup instructions were somewhat unclear. UI design is clean but could benefit from more visual guidance.	Difficulty understanding the steps for setting up the profile. Some fields were confusing.	Add more tooltips and visual aids during the setup process. Simplify instructions and provide examples.
Accessing and Navigating Learning Content	Test access to learning content and navigation	Speed, navigation usability, quality of materials	Content loaded quickly, but navigation between modules was occasionally slow. Materials are generally high-quality but need better organization.	Occasional delays in loading content. Difficulty finding specific topics due to complex navigation.	Optimize loading times and streamline the navigation structure. Implement a more intuitive content hierarchy.
Using the Interactive Coding Environment	Test coding editor functionality and effectiveness	Performance, tools/features, code execution experience	The coding environment is responsive with useful features like code hints. However, error detection is sometimes inaccurate. Communication tools worked well, but collaboration features had occasional glitches. Community engagement is moderate but could be more interactive.	Inaccurate error messages led to confusion during debugging. Some coding tools were not functioning as expected.	Improve error detection algorithms and refine code hint features. Ensure all tools are fully functional.
Engaging in Collaboration and Community Tools	Test collaboration and community features	Communication tools, collaboration ease, community engagement	Progress tracking is generally accurate and gamification elements are motivating. Feedback could be more detailed. Switching between languages was mostly smooth, but integration of cross-language resources was inconsistent.	Some progress metrics were incorrect. Gamified elements sometimes failed to update in real-time. Inconsistent integration of cross-language resources. Occasionally slow transitions between language modules.	Fix bugs in collaboration tools and enhance community features to encourage more interaction.
Tracking Progress and Achievements	Test progress tracking and gamified elements	Accuracy, gamification effectiveness, feedback quality	The app functioned well on smartphones but had layout issues on tablets. Performance was generally good, but some device-specific glitches were noted.	Layout problems on tablets. Minor performance issues on certain devices.	Ensure real-time updates for gamified elements and provide more detailed feedback on progress.
Navigating Between Different Programming Languages	Test switching between languages and integration	Transition smoothness, cross-language support, content integration			Improve cross-language resource integration and enhance the speed of transitions between language modules.
Accessing the Application on Different Devices	Test performance across different devices	UI consistency, performance, device-specific issues			Optimize UI for various screen sizes and resolutions. Address device-specific performance issues.

Fig. 12. Snippet of usability testing results obtained from 7 testers.

The form contained questions based on usability metrics such as effectiveness, efficiency, ease of use, and satisfaction, which were used to evaluate the user experience. Data gathered from these forms were analyzed and summarized to provide insights into the app's performance across different

aspects. The results were plotted using the System Usability Scale (SUS) as shown in Table 1 to visualize the levels of acceptance and areas that needed improvement. The table shows responses ranging from strongly disagree (1) to strongly agree (3).

Table 1: Questions and SUS for measuring the usability of the system

S/N	Questions	SA	N	SD
1	I think I would like to use this system frequently.	0%	43%	57%
2	I found the system unnecessarily complex.	86%	14%	0%
3	I thought the system was easy to use.	0%	29%	71%
4	I think I would need the support of a technical person to use this.	71%	14%	14%
5	I found the various functions in this system well-integrated.	0%	43%	57%
6	I thought there was too much inconsistency in this system.	29%	43%	29%
7	I would imagine that most people would learn to use this system quickly.	0%	14%	86%
8	I found the system very cumbersome to use.	71%	14%	14%
9	I felt very confident using the system.	0%	47%	53%
10	I needed to learn a lot of things before I could get going with this system.	43%	29%	29%

SA = Strongly Agreed; N = Neutral; SD = Strongly Disagree

The performance metrics indicate that the mobile application tool for undergraduate students learning programming languages meets the desired standards of efficiency, accuracy, and stability. However, several areas were identified for potential improvement:

- i. Refine Coding Error Detection Accuracy: Although the interactive coding environment performs well, further refinements could enhance its ability to handle complex or nuanced coding errors, providing more precise feedback and improving the overall user experience.
- ii. Optimize Response Times for Content Loading: Continued optimization of response times, especially during transitions between different programming languages and loading of complex exercises, could enhance the application's responsiveness and user satisfaction.
- iii. Address Device-Specific Layout Issues: While the application performs well on most devices, addressing the minor layout issues observed on tablets will ensure a more consistent experience across all screen sizes.
- iv. Monitor and Enhance Resource Utilization: As new features are added or updates are made, it will be important to continuously monitor and optimize resource utilization to maintain efficiency and smooth performance across a variety of devices.

Overall, the application performs effectively across all key metrics, providing a strong foundation for an engaging and educational tool for learning programming languages.

VI. FINDINGS

This section outlines the key outcomes from the research, design, implementation, and evaluation stages of the mobile application tool developed to enhance the learning experience for undergraduate students studying programming languages. The primary goal of this study was to systematically analyze and improve the user experience of programming education tools by addressing critical challenges identified through a structured approach.

The research began with an in-depth investigation into common issues faced by users of existing programming learning applications. This was accomplished through a comprehensive literature review, user feedback analysis, and

examination of current educational technology trends. The study identified several key issues, including difficulties with navigating educational content, inefficiencies in coding error detection, limited collaboration capabilities, and performance challenges related to device compatibility and responsiveness.

Based on these insights, a user-centered framework was developed to address the identified challenges. The framework prioritized improving usability by creating a more intuitive navigation structure, enhancing coding error detection accuracy, and incorporating advanced features such

as real-time progress tracking and gamified learning elements. It also aimed to optimize application performance across various devices and to strengthen collaborative features for peer interaction and support.

The framework was implemented through the creation of a prototype mobile application using modern technologies. The application provided an interactive coding environment with real-time feedback and error correction, utilizing Firebase for user authentication and content management. Furthermore, it was optimized for smooth performance on different devices, ensuring broad accessibility.

The evaluation phase involved rigorous usability testing, performance assessments, and comparisons with existing educational tools. The results demonstrated significant improvements in user satisfaction, system performance, and overall usability. The interactive coding environment and enhanced collaborative features were particularly well-received, and the application performed effectively across a range of devices. These findings affirm the effectiveness of the proposed framework in enhancing the learning experience for programming students, providing a solid foundation for further development and wider application in educational technology.

VII. RECOMMENDATIONS

Based on the findings and conclusions drawn from this study, several recommendations can be made for future work and improvements in the development of the mobile application tool for undergraduate students learning programming languages. These recommendations aim to enhance user experience, expand the app's capabilities, and address limitations observed during the testing phases:

- i. Broaden User Testing to Diverse Student Groups: The usability testing was conducted with a limited sample of students, which may not fully capture the app's effectiveness across different user demographics. It is recommended that future studies involve a more diverse group of testers, including students from various disciplines, levels of programming expertise, and learning needs. This approach will provide a more comprehensive understanding of the app's usability and ensure that design improvements address a wider range of user requirements.
- ii. Enhance Interactive Coding Environment Features: While the current coding environment performs well, there is potential for further refinement. Future development should focus on improving error detection accuracy, expanding debugging tools, and providing more nuanced feedback for complex coding scenarios. Incorporating advanced features such as intelligent code suggestions and real-time collaboration tools could significantly enhance the coding experience.
- iii. Integrate Advanced Learning Analytics and Personalization: To further engage users and tailor the learning experience, integrating advanced learning analytics and personalization features is recommended. By analysing student progress and preferences in real-time, the app can offer personalized learning paths, targeted content recommendations, and adaptive challenges that cater to individual learning styles and skill levels.
- iv. Optimise Performance for Scalability and Device Compatibility: As the user base grows, it is crucial to ensure that the app can handle increased traffic and maintain optimal performance. Future iterations should focus on optimizing performance, particularly regarding load times and resource utilization, to ensure smooth operation across a wide range of devices. Implementing scalable infrastructure and cloud-based solutions can help manage high volumes of data and user interactions effectively.

Enhance Accessibility Features: While the current accessibility features provide a good start, there is room for further improvement. Future versions of the app should incorporate customizable font sizes, high-contrast modes, and gesture-based controls to support users with various disabilities. Collaboration with accessibility experts and ongoing user feedback will be essential to create a more inclusive and user-friendly application.

VIII. CONCLUSION

This paper aimed to enhance the user experience of a mobile application designed for undergraduate students learning programming languages by systematically addressing critical challenges identified through both literature review and user feedback. These challenges included inefficiencies in content navigation, inaccuracies in coding feedback, and inconsistent performance across different devices, all of which were key barriers to effective learning and engagement.

To address these issues, a comprehensive framework was proposed and implemented, focusing on improving usability, accuracy, and overall engagement. Key innovations involved optimizing the interactive coding environment to provide more accurate error detection, reducing content loading times, and ensuring a seamless user experience across multiple devices. The application was developed using advanced technologies such as Flutter, Dart, and Firebase, alongside custom solutions to enhance performance and functionality.

The evaluation of the framework, although conducted in a controlled testing environment, provided insightful results. The application demonstrated significant improvements in user satisfaction, coding accuracy, and overall system performance. Enhancements in error detection and faster content delivery were particularly well-received, contributing to a more efficient and engaging learning experience.

In conclusion, this study highlights the importance of thoughtful design and the integration of advanced technologies in developing educational mobile applications. The framework successfully addressed common usability challenges, offering a practical solution for improving the learning experience in programming education. While the scope of testing was limited, the findings suggest that broader implementation and further refinement could yield even greater benefits, underscoring the role of user-centered design in advancing educational technology. Future research should explore the long-term impacts of such interventions on learning outcomes across diverse educational settings.

REFERENCES

- [1]. Jaxongirmirzo, M. (2024). Revolutionizing Education with Interactive Tech Boosts Global Learning Outcomes. *Indonesian Journal of Innovation Studies*.
- [2]. Scherer, R., Siddiq, F., & Sánchez Viveros, B. (2020). A meta-analysis of teaching and learning computer programming: Effective instructional approaches and conditions. *Computers in human behavior*, 109, 106349.
- [3]. Zhou, S., Lin, L., He, Z., & Xiong, N. (2021). Reform and Practice of Experimental Teaching of Programming Courses Based on Internet. *2021 4th International Conference on Information Systems and Computer-Aided Education*.
- [4]. Lin, H.C. (2006). An Internet-based graphical programming tool for teaching power system harmonic measurement. *IEEE Transactions on Education*, 49, 404-414.
- [5]. Seta HB, Afrizal S, Hidayanto AN. Analysis of a Mobile Learning Adoption Model for Learning Improvement Based on Students. *Perception Journal of Information Technology Education: Research*. 2022;21:169–196 10 28945 4955.
- [6]. Katijah Mohd Jamil, S.N., & Shareena Abdul Rahim, N.F. (2021). Mobile Educational Game Application for Middle School Student in Learning Bahasa Melayu (Peribahasa) Using Interactive Method. *2021 IEEE 9th Conference on Systems, Process and Control (ICSPC 2021)*, 153-156.
- [7]. Muthuprasad, T., Aiswarya, S., Aditya, K. S. & Jha, G. K. (2021). Students' perception and preference for online education in India during COVID -19 pandemic. *Social Sciences & Humanities Open*, 3(1), 100101.
- [8]. Rangel-de Lazaro, G., & Duarte, J. (2023). Moving Learning: A Systematic Review of Mobile Learning Applications for Online Higher Education. *Journal of New Approaches in Educational Research*, 12(2), 198-224.

- [9]. Butler, A., Camilleri, M.A., Creed, A. and Zutshi, A. (2021), "The Use of Mobile Learning Technologies for Corporate Training and Development: A Contextual Framework", Camilleri, M.A. (Ed.) *Strategic Corporate Communication in the Digital Age*, Emerald Publishing Limited, Leeds, pp. 115-130.
- [10]. Usmonova G.A. (2024). Exploring the Conceptual Framework of Mobile Learning Technology. *Texas Journal of Philology, Culture and History*.
- [11]. Hung, P., Lam, J., Wong, C., & Chan, T. (2015, July). A study on using learning management system with mobile app. In *2015 international symposium on educational technology (ISET)*, Wuhan, China, 2015, pp. 168-172.
- [12]. Siddiquah, A., Khan, S., Hwang, G., Abbas, M.A., & Ajayi, S. (2021). Usage patterns and effects of mobile learning activities using social learning apps on the achievement of undergraduate students in a history of art course. *Int. J. Mob. Learn. Organisation*, 15, 448-465.
- [13]. Yu, Z., Yu, L., Xu, Q., Xu, W., & Wu, P. (2022). Effects of mobile learning technologies and social media tools on student engagement and learning outcomes of English learning. *Technology, Pedagogy and Education*, 31, 381 - 398.
- [14]. Ahmad Faudzi, M., Che Cob, Z., Omar, R., Sharudin, S.A., & Ghazali, M. (2023). Investigating the User Interface Design Frameworks of Current Mobile Learning Applications: A Systematic Review. *Education Sciences*.
- [15]. Kumar, B.A., & Mohite, P. (2017). Usability of mobile learning applications: a systematic literature review. *Journal of Computers in Education*, 5, 1 - 17.
- [16]. Drolia, M., Papadakis, S., Sifaki, E., & Kalogiannakis, M. (2022). Mobile Learning Applications for Refugees: A Systematic Literature Review. *Education Sciences*.
- [17]. Rosmani, A.F., Abdul Mutalib, A., & Sarif, S.M. (2022). Conveying Asmaul Husna's Knowledge Via Mobile App For Undergrads: Hybridizing Signaling Principle And Nielsen's Design Guidelines. *Journal of Information System and Technology Management*.
- [18]. Hikmah, R., Soepeno, B., Puji, R.P., & Sugiyanto (2021). The Effectiveness of WhatsApp Mobile Learning Shows The Existence of Kamal Historical Site Towards Student Interest. *IOP Conference Series: Earth and Environmental Science*, 747.
- [19]. Coelho, R.C., Marques, M.F., & de Oliveira, T. (2023). Mobile Learning Tools to Support in Teaching Programming Logic and Design: A Systematic Literature Review. *Informatics in Education*.
- [20]. Ortiz, O.O., Alcover, P.M., Sánchez, F., Pastor, J.A., & Herrero, R. (2015). M-Learning Tools: The Development of Programming Skills in Engineering Degrees. *IEEE Revista Iberoamericana de Tecnologías del Aprendizaje*, 10, 86-91.
- [21]. Matthews, R., Hew, S.H., & Choo, K.A. (2009). Multimedia learning object to build cognitive understanding in learning introductory programming. *Advances in Mobile Multimedia*.
- [22]. Falode, O. C., Dome, K., Chukwuemeka, E. J., & Falode, M. E. (2022). Development of an Interactive Mobile Application for Learning Undergraduate Educational Technology Concepts. *International Journal of Professional Development, Learners and Learning*, 4(1), ep2204.
- [23]. Countryman, J., & Hoopes, A. (2011). The Technovation Challenge: Girls Creating Mobile Apps.
- [24]. Patil, P.A., Mane, P.D., & Shinde, P.N. (2022). Incorporating Visualization Tools and Active Learning Approach for Programming Courses. *Journal of Engineering Education Transformations*.