SCIENTIFIC OASIS

Decision Making: Applications in Management and Engineering

Journal homepage: www.dmame-journal.org
ISSN: 2560-6018, eISSN: 2620-0104

Optimising Programming Skill Acquisition Using AI-Enhanced Problem Based Learning and Decision Support Models

Mona Esmat[1,*], Nahed Amasha[2], Shuhui Liu[3] Amira Atta[3], R. E Ladly[4], W.K.El Said[5]

1 Computer Department, Mansoura University, Egypt
2 Computer Department, Mansoura University, Egypt
3 Computer Department, Mansoura University, Egypt
4 Computer Department, Mansoura University, Egypt
5 Computer Department, Mansoura University, Egypt

ARTICLE INFO

Article history:
Received 17 April 2024
Received in revised form 21 October 2024
Accepted 12 November 2024
Available online 30 December 2024

Keywords:
Decision Support System; Problem-Based Learning; Multi-Criteria Decision Making

ABSTRACT

A significant number of first-year computer science students struggle to acquire programming skills, largely due to challenges such as varying levels of logical complexity, limited capacity for algorithmic thinking, and the absence of tailored feedback. In response to these issues, a mobile learning application was designed, incorporating the Problem-Based Learning (PBL) approach. This application integrated interactive exercises, automated feedback mechanisms, and collaborative elements to facilitate the learning of C# programming. The effectiveness of this intervention was evaluated using a quasi-experimental research design, involving 60 participants divided into control and experimental groups. Findings demonstrated that the experimental group achieved statistically significant gains in programming competence when compared with the control group, with notable improvements observed between pre-test and post-test scores. In addition to the empirical analysis, the study employed the Analytic Hierarchy Process (AHP) to inform instructional strategy selection. Three pedagogical approaches—traditional lecture-based learning, non-technological PBL, and mobile-supported PBL—were compared based on usability, learner engagement, skill enhancement, and scalability. Weights were assigned to each criterion, and AHP analysis revealed that the mobile-supported PBL approach ranked highest, indicating a pedagogical advantage in this context. The outcomes of the study provide valuable insights for educators, researchers, curriculum designers, and academic policymakers aiming to enhance programming instruction methodologies.

1. Introduction

Programming courses often require students to undertake coding tasks [3]. However, it appears that students prioritise task completion over the underlying learning process. This misalignment between instructional objectives and student engagement becomes evident when learners are unable to complete assignments. Previous findings indicate that many students face difficulties in

* *Corresponding author.*
E-mail address: monaesmat1980@gmail.com

https://doi.org/10.31181/dmame7220241459

planning and implementing solutions, which often leads to frustration [21] and learning setbacks. Programming, beyond being an academic discipline, represents a vital skill that enables individuals to comprehend and influence the digital environment. Mastery of programming fosters essential problem-solving capabilities, requiring learners to deconstruct complex problems and formulate structured, logical solutions. Furthermore, it promotes adaptability, creativity, and innovation in the development of software systems and applications, contributing to technological advancement [26].

Despite its significance, acquiring programming proficiency presents numerous challenges. A prominent difficulty reported by students involves problem analysis, effective problem representation, the selection of appropriate teaching methods, and a general lack of understanding of programming syntax and structures [13]. Addressing these issues through real-world problem-solving and programming design using code as an expressive medium presents a compelling pedagogical strategy. Programming typically involves a structured approach that includes exploring potential solutions, using various representational tools (including visual aids), formulating a response, and evaluating and refining that solution until the problem is resolved [24]. Identifying the key pedagogical elements of this approach enables students not only to learn programming but also to cultivate the analytical and critical thinking required to engage meaningfully with technology [28].

PBL is an instructional method that immerses students in solving real-life problems, thereby encouraging self-directed learning and the development of critical thinking. Unlike traditional lectures, PBL positions the instructor as a facilitator who supports students through open-ended, complex scenarios [6]. Originally established for medical education, PBL has since been successfully adapted for use across various disciplines and instructional formats. Rather than focusing solely on content delivery, PBL aims to build competencies such as analytical thinking, problem-solving, collaboration, communication, and digital skills. It further encourages learners to critically assess research materials and embrace an approach rooted in organisational insight and continuous learning, in contrast to the passive memorisation practices still common in many academic settings. Through the application of PBL, students may cultivate a professional mindset that is both flexible and resilient, well-suited to navigating the uncertainties and evolving demands of modern workplaces [10]. By shifting the emphasis from passive knowledge acquisition to active inquiry, PBL supports greater student engagement, long-term retention of knowledge, and interdisciplinary learning [2]. As a teaching and curriculum model, PBL equips students with practical problem-solving capabilities, preparing them to address complex, real-world challenges effectively.

Programming education encompasses multiple learning domains: the cognitive domain, which involves the understanding and evaluation of knowledge; the affective domain, which pertains to learners' attitudes and organisation of information; and the psychomotor domain, which concerns the development of practical skills [16]. The traditional instructional methods still dominant in many classrooms offer only a formal environment, posing difficulties for novice programmers in reading, tracking, writing, and developing basic code. This frequently leads to over-reliance on peers to complete assigned tasks, fostering passivity and diminishing learners' motivation, with some students merely expecting lenient grading from instructors [9]. In recent years, mobile applications have become increasingly prevalent in educational contexts, particularly as tools to support teaching and learning [25]. Mobile learning (m-learning) offers a distinctive benefit through its flexibility, enabling learners to access educational content at any time and location [15]. It promotes a self-directed approach, giving students autonomy over when, where, and how they engage with their studies [18]. Learners have expressed a preference for mobile technologies as learning tools, citing their usability and relevance to contemporary digital culture [27].

This study investigates a central challenge in programming education: the high incidence of

errors among beginners and the limited availability of personalised support due to constraints in teaching resources. The significance of this research lies in its integration of a problem-solving instructional approach within a mobile application, which supports students in acquiring practical experience with C# programming. The mobile platform enables learners to write code, address programming problems, debug errors, and develop problem-solving competencies in a flexible and structured learning environment. While previous research on instructional strategies has largely relied on quantitative metrics (such as test scores) or subjective teaching evaluations, there remains a need for a structured framework to evaluate and prioritise multiple pedagogical options across diverse criteria. This study addresses this gap by employing the AHP as a decision-making model to compare instructional methods. AHP facilitates the systematic evaluation of alternatives—namely, traditional lectures, non-digital PBL, and mobile-supported PBL—based on criteria including usability, learner engagement, skill acquisition, and scalability. The use of this multi-criteria framework enhances educational planning by combining empirical evidence with structured decision-making processes.

1.1 Research Questions

"What is the degree of effectiveness of utilising a mobile application designed according to the PBL approach in enhancing programming abilities among first-year computer science students enrolled at the Faculty of Specific Education, Mansoura University?"

1. This overarching inquiry is further explored through the following sub-questions:
2. What are the methodological steps involved in constructing a mobile application grounded in the principles of the PBL approach?

How effective is the implementation of a PBL-based mobile application in improving the programming competencies of first-year computer science students at the Faculty of Specific Education, Mansoura University?

1.2 Research Hypotheses

A. There is a difference at the level of significance (≤ 0.05) between the mean the control group and the experimental group, in the post-application of the test measuring programming skills as a whole, in favour of the experimental group.
B. There is a difference at the significance level (≤ 0.05) between the mean scores of the experimental group students in the pre- and post-application of the test measuring programming skills, in favor of the post-application.

2. Related Works

This research enabled the integration of a structured problem-solving strategy into a mobile application, offering learners a dynamic and interactive platform for acquiring C# programming skills. A PBL approach facilitates the development of both technical and interpersonal skills, promoting deeper understanding by shifting from isolated instructional sessions to an integrated curriculum. Real-world projects foster teamwork and independent learning while discouraging academic dishonesty, as skills must be applied rather than reproduced from external sources. Large group-based assignments, conducted over extended periods, familiarise students with realistic development cycles and highlight that failure in such contexts is often managerial rather than technical in nature [5].

A structured framework has been proposed to guide the systematic implementation of PBL in programming education. Drawing on conceptual reviews and qualitative insights from educational stakeholders, the model addresses the human, procedural, and product aspects of instructional

practice. This framework supports critical thinking, autonomous learning, and enhanced programming capability, aligning pedagogy with the demands of the computing industry [7]. Another model combining PBL with block-based programming was developed to promote programming proficiency. It includes core instructional elements such as input, learning stages, feedback, and assessment, offering learners a cohesive experience. The structured stages—problem definition, resource exploration, concept mapping, design, coding, testing, and presentation—encourage critical thinking and experiential learning. Evaluations confirmed the model's relevance and effectiveness in enhancing programming instruction within undergraduate courses [14].

A hybrid PBL-CBL model was examined in the context of ICD coding education. In a controlled study, students instructed using this model demonstrated notable improvements in knowledge, operational skills, and cognitive development. The approach, based on information processing theory, showed potential for broader use beyond its original discipline due to its positive effects on academic performance and critical reasoning [29]. Further exploration into the readiness of institutions to adopt PBL highlighted both enabling factors and challenges. Increased learner engagement was evident, though some resistance was noted from those accustomed to traditional, memorisation-based learning. A diagnostic tool was created to evaluate institutional preparedness, offering a strategic plan for successful PBL integration in varied educational contexts [20].

Introductory programming courses employing PBL methods revealed increases in student motivation, collaboration, and problem-solving. Learners participated in group discussions, prior knowledge assessments, and practical projects such as creating mobile applications from scratch. Positive learning outcomes and high pass rates demonstrated that PBL enhanced learner autonomy, facilitated instructor-student engagement, and supported the attainment of course objectives [1]. An online adaptation of PBL applied in secondary programming education demonstrated strong effectiveness in improving problem-solving abilities, programming skills, and higher-order thinking. Over an eight-week period, students engaged in collaborative activities using social media platforms, supported by facilitators. Post-intervention assessments confirmed that online PBL environments support active learning and increase overall performance [4].

A blended PBL approach incorporating visualisation tools was implemented in a mobile app development course. Through scaffolded, hands-on tasks, students advanced from basic projects to functional applications. Empirical evaluation using knowledge tests, student surveys, and project outcomes demonstrated marked improvements in both perceived and actual learning in mobile programming contexts [19]. A mobile application was also designed to streamline communication and academic processes within educational institutions. The system integrated educators, learners, and administrative staff to create a connected environment that prioritised usability, accessibility, and functionality. While it improved engagement and efficiency, concerns remained regarding child device usage, requiring careful design considerations to ensure safety alongside educational benefit [23].

Graduate students with no prior programming experience were introduced to mobile app development through a web-based visual platform. The environment promoted creativity, peer collaboration, and practical achievement. Despite minor limitations in interface design for complex projects, learners valued the flexibility and empowerment offered by the platform, which also informed refinements in instructional delivery [12]. To address academic inefficiencies, a mobile application was implemented at a university to unify access to course information and reduce administrative dependency. This centralised platform improved both student and staff experiences, streamlining operations and enhancing educational outcomes, thereby demonstrating the value of mobile technologies in academic support structures [8].

A mobile learning application was evaluated for its impact on learner engagement and

motivation in programming contexts. Feedback from users and experts highlighted its relevance and instructional effectiveness. By embedding real-world tasks, instant feedback, and collaborative elements within a PBL framework, the application fostered deeper involvement, enhanced skill acquisition, and encouraged ownership of the learning process [17]. In summary, mobile learning applications offer an engaging and interactive educational environment that promotes problem-solving and supports the development of programming skills through the completion of real-world tasks. These tools deliver a scaffolded instructional structure, allowing learners to engage with authentic challenges, receive timely feedback from the system, and collaborate with peers. The integration of PBL within such applications not only enhances learner engagement and participation but also fosters greater autonomy and skill acquisition in programming contexts.

3. Method

This study adopted a dual research trajectory. The first involved an empirical investigation through a quasi-experimental design, while the second employed a decision-support framework grounded in the AHP, facilitating a multi-method approach to systematically gather information and assess instructional innovations during the research process. Beyond statistical validation through the quasi-experimental method, AHP provided a structured mechanism for evaluating the instructional strategies used. Specifically, it enabled a comparative analysis of three pedagogical approaches to teaching programming: conventional lecture-based instruction, non-digital PBL, and the mobile-supported PBL model developed in this study. The AHP evaluation was based on four key criteria—usability, learner engagement, enhancement of programming skills, and scalability. These criteria were derived from prior educational research and refined through interviews with relevant stakeholders. Five experts specialising in educational technology and programming pedagogy participated in a structured pairwise comparison exercise, assessing both the criteria and the instructional alternatives. Their responses were then analysed using AHP to determine relative weights and consistency ratios, thereby ensuring logical coherence in the prioritisation process. Based on the calculated weights, a decision matrix was employed to assess and score each instructional method against the four established criteria. A composite ranking was subsequently derived to identify the most preferred instructional strategy. This layer of analysis enriched the empirical findings by offering a structured rationale to guide decision-making in the presence of competing pedagogical models within the complex domain of programming education.

3.1 Mobile App Architecture Layers

3.1.1 Presentation Layer

The application was developed using Java, a widely adopted programming language for Android development due to its flexibility in server integration and its capacity to structure entire systems. The design approach was informed by both User Interface (UI) and User Experience (UX) principles. Within the UI framework, familiar colour schemes were intentionally chosen to create visual consistency, and the night mode function was disabled to ensure a uniform design experience across different devices. Iconography was deliberately kept simple and intuitive to represent application functions without the need for extensive explanatory text. From a UX perspective, the educational content was presented in a clear, concise, and easily readable format, supplemented with illustrative examples to support understanding of each programming function. The application includes a wide range of exercises aligned with the learning material, reinforcing key concepts and supporting autonomous learning. A built-in self-assessment feature allows students to complete short quizzes at the end of each unit, with immediate automated feedback provided to identify

areas requiring further improvement. Collectively, these features contribute to a comprehensive learning environment designed to support effective and sustainable development of programming skills.

3.1.2 Business Layer

The business layer is responsible for defining the logic, procedural rules, and workflows involved in data exchange and system processes. Its functions include the following:

Security: It ensures the confidentiality of individual student data, allowing access only to the respective user. This includes personal statistics such as assessment scores, completed exercises, and teacher-assigned tasks.

Data Caching: Temporary storage is applied for critical information, such as academic results or communication records, to maintain accessibility during internet disruptions. Caching is particularly essential when real-time communication with the server is interrupted, allowing users continued access to essential functions.

Logging: Account activity is regulated to prevent duplication. Each account is limited to a single registration, and the logging mechanism oversees functions such as user registration, contest enrolment, and access control. It ensures that users can only view incomplete test records, while completed assessments remain inaccessible to preserve academic integrity.

Data Validation: During account creation and login, the application cross-checks user credentials with the server to prevent duplication. Only when the input data matches an existing account is access granted; otherwise, entry is denied to ensure the system maintains account accuracy and exclusivity.

3.1.3 Data Layer

The data layer encompasses all components responsible for data-related operations, including data services, utilities, service agents, and access modules that facilitate data transactions. It comprises two main segments:

Persistence: This component manages data access between the application and its data sources, ensuring consistent and secure retrieval and storage of information via the mobile interface.

Network: This segment oversees network communication, routing mechanisms, and the handling of network-related error messages. In addition to these core functions, the data layer incorporates mechanisms for data validation and maintenance, ensuring the integrity and functionality of the system.

Each of these components operates independently, with information exchanged through clearly defined interfaces, thereby ensuring modularity and streamlined system integration.

Figure 1 presents the structural framework of a mobile application designed using a PBL approach to support the learning of C# programming functions. The key components are outlined as follows:

- User Interaction: Learners interact with the application through login processes, content navigation, and response submission. The system records user inputs and provides feedback on individual learning progress.
- Problem Identification: Coding challenges are presented to prompt learners to identify and analyse programming problems.
- Knowledge and Skills Development: Users engage with instructional materials focused on C# functions, facilitating the acquisition of essential programming knowledge and skills.
- Application and Solution: Acquired knowledge is applied to solve programming tasks, encouraging active problem-solving.

- Reflection and Feedback: The system delivers reflective insights and feedback, allowing users to refine their understanding and improve performance.
- System Database: The application connects to a backend database, transmitting user queries related to tasks and learning progress while retrieving appropriate instructional content and performance feedback.

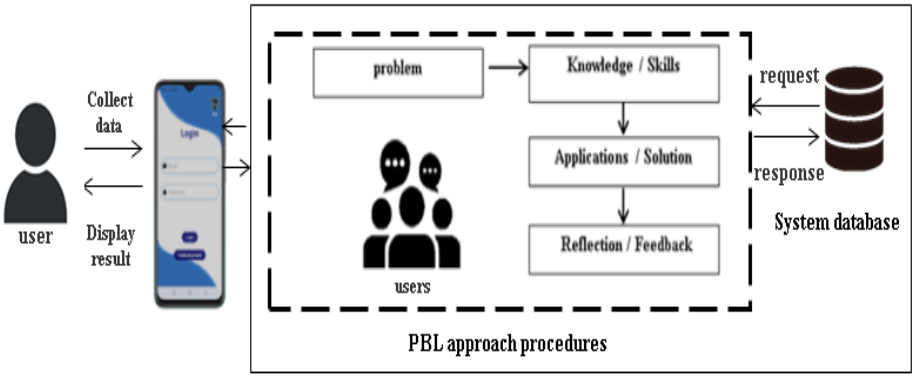


Fig. 1: System Architecture of the Proposed Application

The application includes a built-in group communication feature aimed at fostering collaborative learning through an integrated chat environment. This functionality enables students to engage in real-time discussions, allowing them to collaborate on exercises, exchange ideas, debate potential solutions, and raise questions. Users have the option to create or join groups, facilitating the sharing of information and promoting joint work on assignments and projects. This feature enhances student engagement and encourages active participation in the learning process. Moreover, it supports peer-to-peer interaction and contributes to the development of teamwork and problem-solving skills. Figure 2 illustrates the user interface of the mobile application, highlighting both the registration screen and the main operational screens for students and teachers. In Figure 2(a), the registration interface requires users to set up a username and password, which will be used for subsequent access to the application.

(a) (b) (c)

Fig. 2: (a) User Enters His Data and Presses "Login" or Use Help (b), (c) Main Screen for Student and Teacher

A help icon is also integrated into this screen, granting users access to a user guide (as shown in Figure 3) that outlines the application's functions and features. Figures 2(b) and 2(c) depict the primary interfaces for students and teachers, respectively. The student interface allows learners to monitor their academic progress and navigate through various educational sections, while the teacher interface is designed to track student performance and manage instructional tasks and

assignments.

Fig. 3: User Guide Screens

Figure 4 presents the pre-test interface, which students encounter immediately after logging into the application.

Fig. 4: Pre-Test Screens

This diagnostic assessment is intended to evaluate their baseline proficiency prior to engaging with the instructional content within the app. Upon completion of the pre-test, users are redirected to the main interface, where they can explore different functional sections of the application. By navigating to the "Book" section, learners gain access to educational materials focused on C# programming functions, as shown in Figure 5. This structured learning pathway enables the delivery of content aligned with each student's initial performance, thereby supporting a more personalised and effective educational experience.

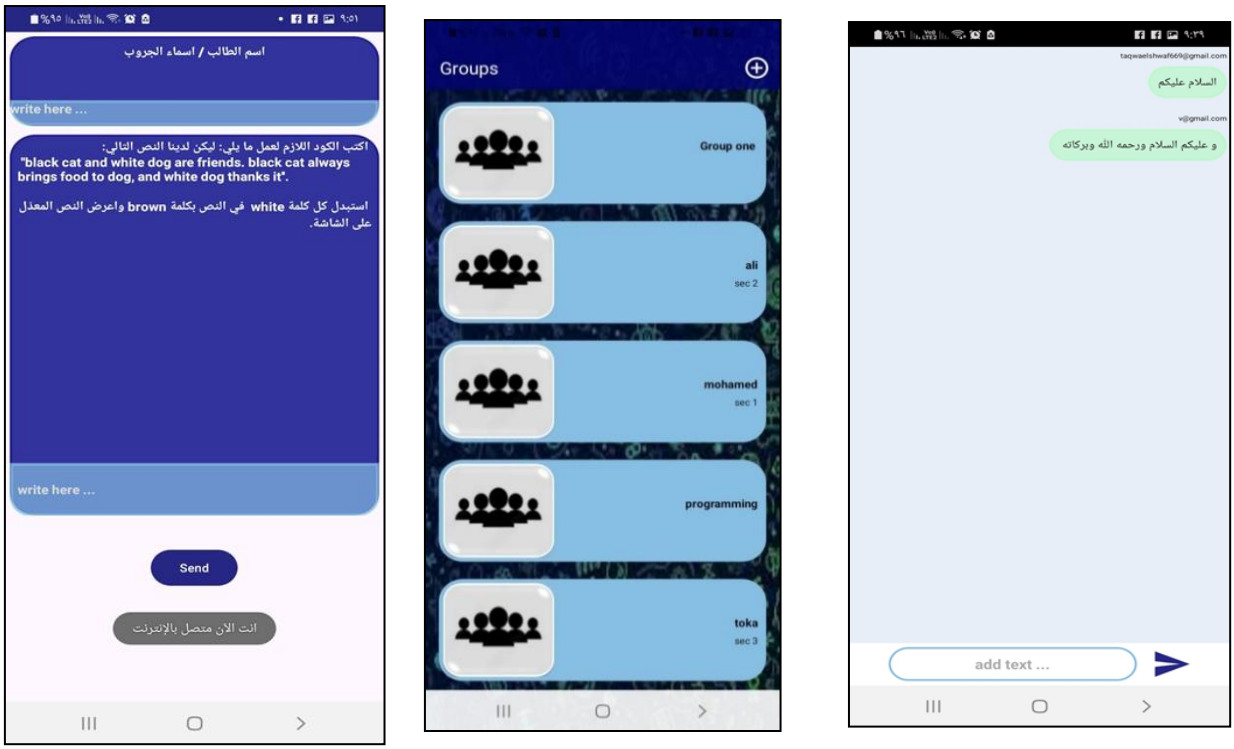
Fig. 5: Book Section

Figure 6 displays the book section of the application, where students can select specific functions from a presented list. Upon selection, the application provides illustrative examples accompanied by interactive exercises intended to reinforce conceptual understanding and develop problem-solving abilities. Through continuous engagement with these tasks, learners are able to gradually enhance their proficiency in C# programming by applying their knowledge in guided and contextually relevant scenarios.

(a) (b) (c) (d)

Fig. 6: (a), (b) Explanation of the Functions and (c), (d) Examples and Exercises

Furthermore, Figure 7(a) presents the Task Section of the application, where learners engage with a series of questions related to C# programming functions. This section is designed to facilitate the evaluation and reinforcement of students' understanding by offering interactive problem-solving activities that require users to complete coding tasks with their responses. This engagement allows learners to assess their comprehension, monitor their progress, and develop their programming abilities through experiential learning. Figures 7(b) and 7(c) illustrate the Chat section, which enables real-time interaction between learners and instructors. This feature supports group creation, allowing students to collaborate with peers, exchange ideas, resolve programming challenges collectively, and work together on exercises. The chat function enhances peer-to-peer learning and provides a direct communication channel for students to pose questions and receive guidance from both classmates and teachers. Furthermore, it facilitates immediate feedback and encourages knowledge exchange in multiple directions, thereby fostering active engagement and sustained participation in the learning process.



(a) (b) (c)

Fig. 7: (a) Task Section Displayed Various Questions on Functions and (b), (c) Chat Section

Figure 8 presents the "Show Result" interface, which displays the outcomes of both the pre-tests and exercises for learners and instructors. This feature serves as a performance monitoring tool, offering tailored feedback to support progress evaluation and instructional planning.

- For Students: The "Results" screen presents individual performance indicators, including assessment scores and progression metrics. This enables learners to identify areas of strength, recognise aspects needing improvement, and monitor their development over time.
- For Teachers: The same interface provides instructors with comprehensive data on students' performance. This includes trends in learning achievement, insight into specific strengths and weaknesses, and the ability to provide focused support where necessary to promote improved outcomes.

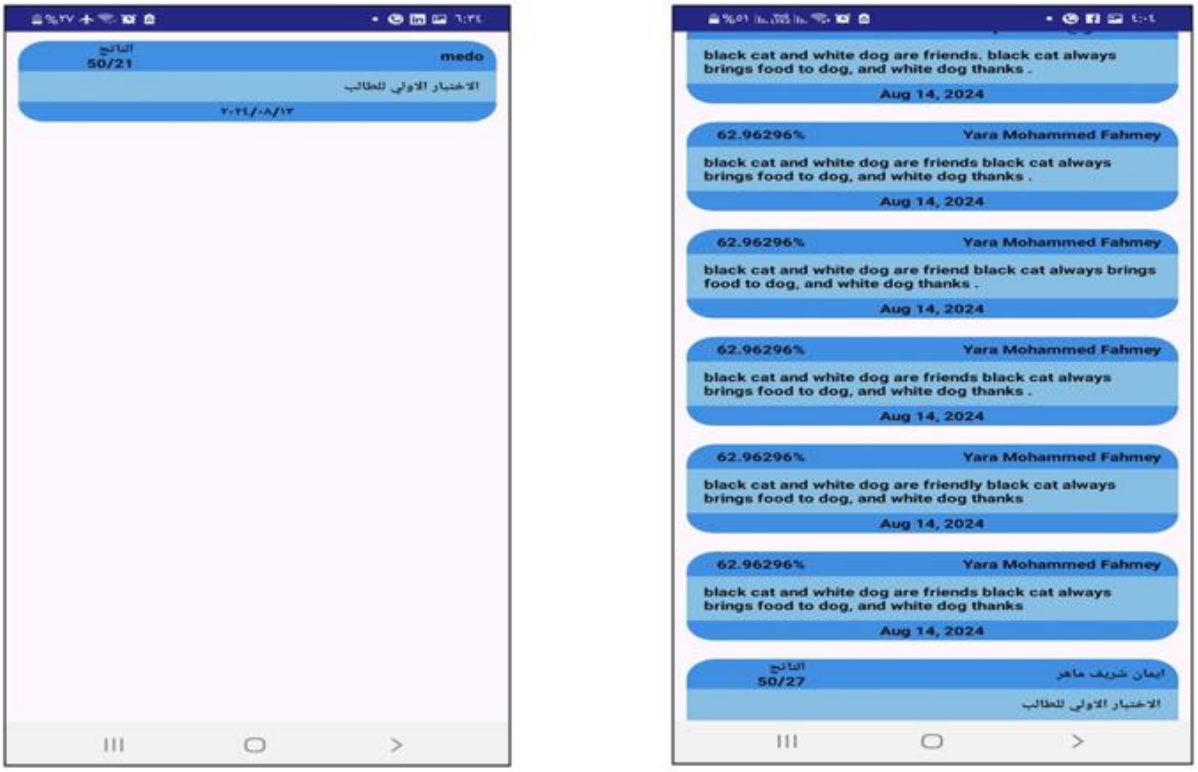


Fig. 8: (a) Result Screen for Teacher and (b) Result for Students

Figure 9 illustrates the "Add Task" section, which allows instructors to design and upload coding exercises for student engagement, thus fostering a structured and interactive learning process. When setting a task, the instructor formulates a programming question and designates two correct solutions as reference answers. This approach ensures clarity in instructional intent and supports automated evaluation.

Fig. 9: (a), (b) Add Task and Answers, (c) Uploading Task

By incorporating predefined solutions, the system can assess student submissions instantly,

provide immediate feedback, and maintain an accurate record of learning progression, as outlined in the procedural flow shown in Figure 10. Moreover, Figure 11 displays the Task Section from the student perspective, where learners are assigned exercises requiring the submission of written code in a designated response area. The system automatically evaluates the submitted work based on the defined answer criteria.

- Correct Answer: When the student's code matches the expected solution, the system awards a full score of 100%, and the outcome is recorded in the "Result" section for tracking academic progress.
- Incorrect Answer: If the code contains errors, the system highlights the specific issues in red and adjusts the score accordingly. This immediate feedback mechanism enables students to identify mistakes and revise their work.

This automated evaluation process enhances the learning experience by offering timely, specific feedback, thereby supporting the development of coding accuracy and reinforcing conceptual understanding through practical application.

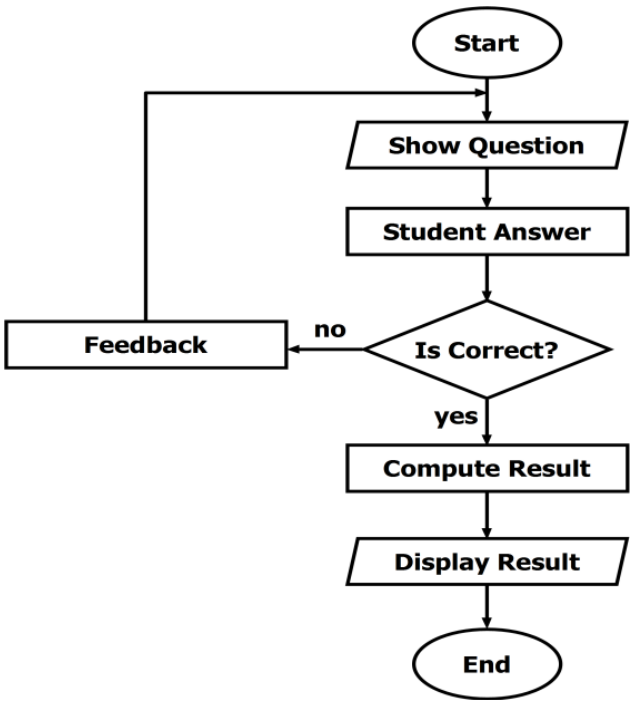


Fig. 10: Flowchart of Student Answer

Exam Section: Following the completion of the instructional content, students are required to undertake a final examination to assess their learning outcomes. The application automatically captures and stores the exam scores, allowing for detailed statistical analysis of learner performance. This functionality enables comparisons between pre-test and post-test results, facilitates the identification of individual progress, and provides a basis for evaluating the overall impact of the PBL-based approach on the development of programming skills. A group of subject matter experts systematically reviewed the application to determine its effectiveness, usability, and alignment with educational needs. Their evaluation confirmed that the application complies with academic standards, effectively incorporates PBL principles, and offers students a structured and engaging learning experience. The design and refinement of the application were informed by expert input throughout the development process. This feedback contributed to enhancing the usability, structure, and instructional effectiveness of the tool, resulting in a solution optimised for teaching C# programming functions.

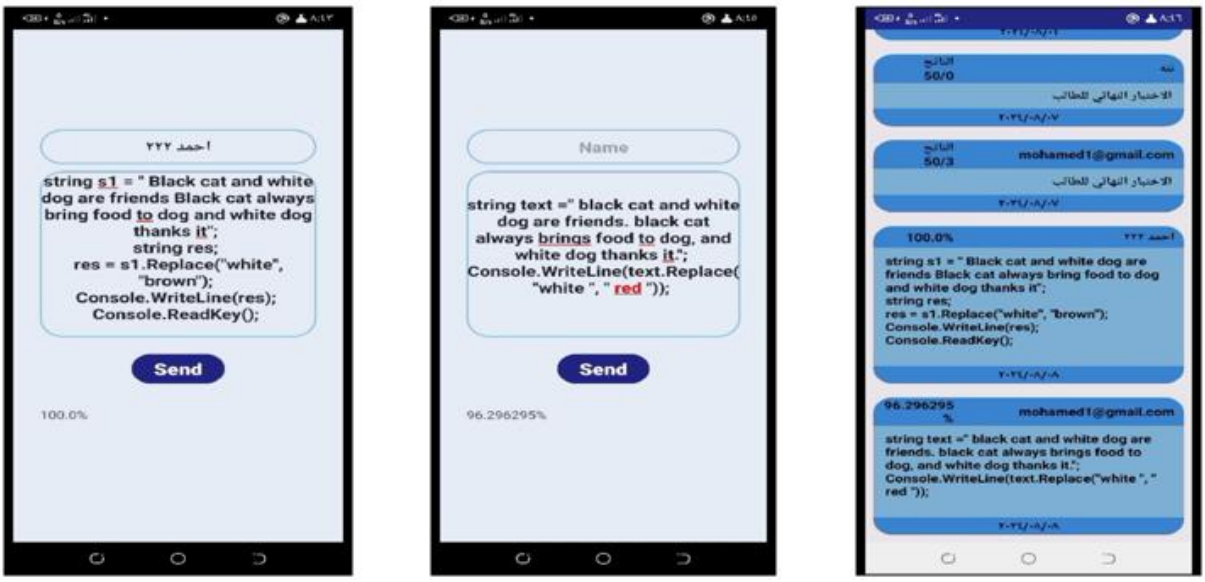


Fig. 11: (a) Correct Answer, (b) Incorrect Answer and (c) Student Results

3.2 Developing and Validating the Research Tool

The Programming Skills Assessment Test was initially composed of 55 items. To determine its validity, the test underwent expert review by three specialists in the field. Based on their feedback, several items were revised for clarity and alignment with the intended learning objectives, while others were rephrased or removed entirely due to their lack of relevance or insufficient alignment with the assessment criteria. Following these revisions, the test was refined to include 50 items deemed appropriate for assessing programming competencies. The reliability of the finalised assessment tool was established using the test–retest method. This procedure involved administering the test to a pilot group of thirty first-year students, followed by a second administration two weeks later to the same sample. The scores from both testing phases were analysed using Pearson correlation to estimate the consistency of results over time. As presented in Table 1, the overall reliability coefficient for the assessment instrument was calculated at 0.83, which is statistically significant at the 0.01 level. This outcome indicates that the test demonstrates a high degree of reliability for evaluating programming skills.

Table 1
Reliability Coefficient

Reliability Coefficient	Significance Level
0.83	0.01

4. Research Sample

The study sample included 60 first-year students enrolled in a computer science programme. These participants were divided into two equal groups, with 30 students assigned to the control group and the remaining 30 students allocated to the experimental group.

4.1 Equivalence between the Control and Experimental Groups in Programming Skills

To ensure equivalence in programming proficiency between the control and experimental groups prior to the intervention, a pre-test on programming skills was administered to both sets of participants. Following the data collection, an independent samples t-test was performed to compare the mean scores and standard deviations of the two groups. The analysis proceeded with the calculation of the t-value representing the difference between group means, along with the

corresponding level of statistical significance to determine whether any initial disparity existed between the groups. Table 2 demonstrates that the difference in mean scores between the experimental and control groups on the programming skills pre-test is not statistically significant at the 0.05 level. This outcome confirms that both groups exhibited comparable levels of programming proficiency prior to the intervention.

Table 2
Pre-Test for Programming Skills

Dimensions	Group	N	Mean	Std. Deviation	DF	T-Value	Sig. Level
1	Control	30	12.10	5.42	58	0.76	0.44
	Experimental	30	10.73	8.12			Non-Sig.

4.2 The Experimental Study

- The Programming Skills Assessment Test was administered to the control group using a traditional paper-based format, whereas the experimental group completed the same pre-test through the mobile application designed with a PBL framework. The resulting scores were systematically recorded for subsequent statistical analysis.
- Instruction for the control group followed conventional teaching methods, while the experimental group received instruction through the mobile application that incorporated the PBL strategy.
- The post-test was conducted using the same modality as the pre-test for each group: paper-based for the control group and application-based for the experimental group, maintaining consistency in the assessment approach across the study.

5. Results and Discussion

5.1 Testing the Validity of the First Hypothesis

To examine the validity of the first hypothesis, the difference between the group means was subjected to statistical analysis by calculating the t-value along with its corresponding significance level. The outcomes, including the mean scores, t-value, and level of significance, are presented in Table 3. The data presented in Table 3 indicate the following findings:

- A statistically significant difference at the 0.01 level was identified between the mean post-test scores of the experimental and control groups, with the experimental group exhibiting higher levels of programming proficiency.
- The experimental group's performance in overall programming skills showed a marked improvement compared to the control group following instruction through the mobile application, which was based on a PBL approach and incorporated pedagogical practices centred on solving authentic programming problems.

These findings affirm the acceptance of the study's first hypothesis. The outcomes can be attributed to several instructional elements embedded in the application:

- A major contributing factor to the improved performance is the app’s real-time communication functionality, which enabled continuous interaction between students and teachers. This allowed for immediate feedback, peer collaboration, and problem discussion, fostering a dynamic and supportive learning environment that enhanced comprehension and performance.

Regular engagement in programming tasks and consistent practice through sequential projects facilitated the gradual development of students’ programming capabilities. The relevance of the instructional content became more evident as learners applied their knowledge practically.

Furthermore, structured feedback and targeted guidance helped students to recognise their errors and improve their coding performance, ultimately contributing to the development of programming skills. These results are consistent with the findings reported in previous studies [19, 21], which support the use of mobile applications as effective tools for enhancing self-directed learning and improving academic outcomes. Key benefits of mobile-based instruction include:

- Anywhere Anytime Access: Mobile learning platforms provide learners with flexible access to educational content, allowing them to engage with instructional material at their convenience and promote independent study.
- Interactive Learning: The integration of quizzes and other interactive elements reinforces student engagement and deepens understanding of the subject matter.
- Personalised Learning: The application supports individual learning preferences by enabling students to select specific topics for further practice and monitor their progress according to personal goals.
- Immediate Feedback: Instant responses to student inputs allow for quick identification of errors and facilitate continuous performance improvement.
- Support for Self-Directed Learning: The application fosters essential self-regulated learning skills, including time management, task organisation, and independent study habits.

Table 3
Post-Test for Overall Programming Skills

Group	N	Mean	Std. Deviation	DF	T-Value	Sig. Level
Control	30	26.43	5.31	58	4.47	0.01
Experimental	30	34.40	8.16			

5.2 Testing the Validity of the Second Hypothesis

To test the second hypothesis, a paired samples t-test was employed. The analysis involved calculating the mean and standard deviation of programming skill scores for the experimental group before and after exposure to the instructional intervention. The corresponding t-value for the difference between the means was determined, alongside its statistical significance. Table 4 presents the findings of this comparison. As presented in Table 4:

- A statistically significant difference was observed at the 0.01 level between the experimental group's mean scores on the pre- and post-tests, with the post-test scores being notably higher.
- Programming proficiency among students in the experimental group significantly increased in the post-test when compared to their pre-test results.
- The reduction in score variability on the post-test suggests that students' performance levels became more uniform, indicating that the majority reached a high level of competence after completing the programme via the mobile application based on the PBL model.

These findings support the acceptance of the second hypothesis. The application not only assesses task completion but also provides immediate corrective guidance in cases of error, particularly on proper coding practices. This feature is interactive in nature, which plays a crucial role in facilitating understanding and retention. Interactivity has been shown to enhance learning outcomes, and in this context, it enabled students to strengthen their comprehension, recall, and application of programming concepts. These conclusions are in agreement with earlier research findings [1; 11; 20; 23], which have demonstrated that learners tend to acquire programming skills more effectively when they engage in collaborative environments. The involvement in team-based learning activities, problem-solving tasks, and peer collaboration significantly contributes to improved learning outcomes and overall skill development.

Table 4
Pre-Test and Post-Test for Programming Skills

Group	N	Mean	Std. Deviation	DF	T-Value	Sig. Level
Control	30	10.73	8.12	29	18.15	0.01
Experimental	30	34.40	8.16			

5.3 The Effectiveness of a Mobile Application Based on the PBL Strategy in Developing Programming Skills

To assess the impact of the mobile application designed with the PBL strategy on enhancing programming proficiency among first-year students, the T-value, eta squared, and effect size were computed. The summary of these statistical measures is presented in Table 5. According to the data in Table 5, the eta squared value associated with overall programming skills is 0.26. This result signifies that the influence of the mobile application, underpinned by the PBL approach, on the experimental group's programming skill development is substantial. Based on established interpretation guidelines [22], an eta squared value of 0.15 corresponds to a large effect size of 0.84. In this case, the effect size exceeds that threshold, reaching a value of 1, indicating a highly significant effect. These findings confirm that integrating a PBL-based mobile application into programming instruction serves as an effective method for significantly improving students' programming capabilities.

Table 5
Effect Size

T-Value	Eta Squared	Effect Size
4.47	0.26	High

Figure 12 presents a comparative analysis of the three instructional approaches based on the predetermined evaluation criteria. The mobile application supported by the PBL strategy demonstrated the highest performance in usability, student engagement, and skill enhancement, reflecting a strong alignment with targeted learning objectives.

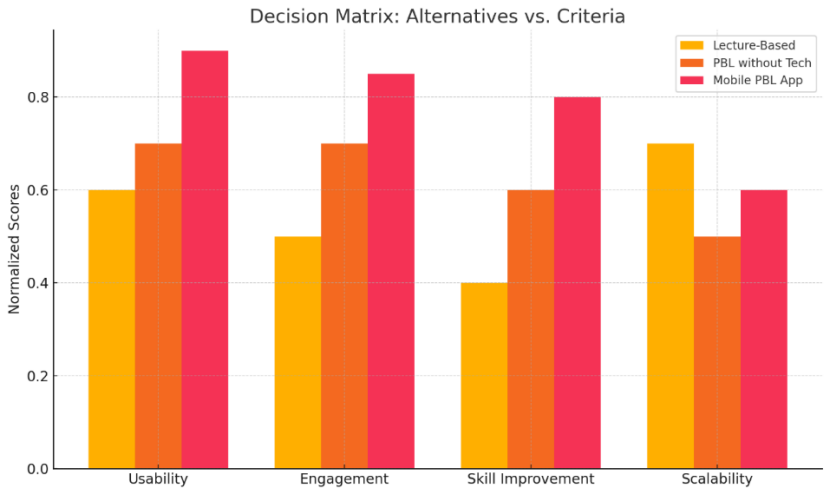


Fig. 12. Decision Matrix

The PBL approach without technological integration also showed favourable outcomes, particularly in fostering learner engagement, though it was less effective in the remaining areas. In contrast, the traditional lecture-based method ranked lowest across most dimensions, with the exception of scalability, where it displayed relatively stronger performance. These standardised

scores were subsequently utilised to compute the final results through the AHP, offering a comprehensive basis for instructional strategy evaluation.

Figure 13 displays the overall weighted scores for the three instructional methods. Among them, the mobile application utilising the PBL strategy attained the highest composite score, indicating its superiority over the other alternatives. These findings validate the outcomes of the experimental phase by integrating expert evaluations with a multi-criteria decision-making approach. PBL without technological support ranked second, followed by the lecture-based method, which received the lowest score. The results reinforce the conclusion that a technology-integrated PBL model delivers the most substantial educational benefits under the defined study conditions.

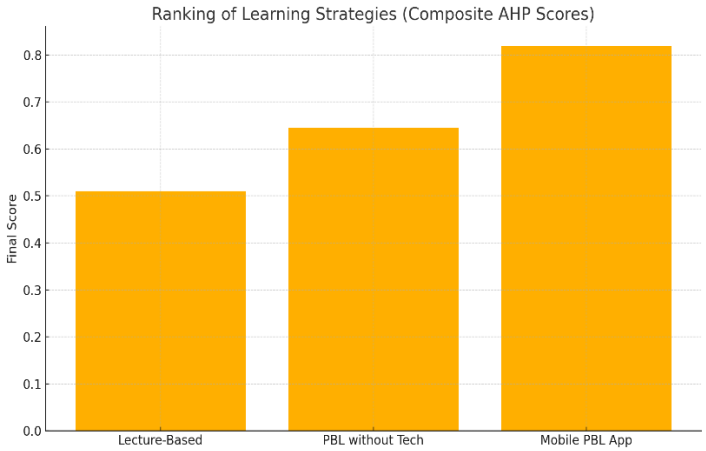


Fig. 13. Ranking of Learning Strategies

Figure 14 presents the distribution of priorities that guided the decision-making process within the AHP analysis. The criterion of skill improvement received the highest weighting (0.479), indicating that stakeholders regarded it as the most critical factor in programming education. This was followed by engagement (0.199) and usability (0.165), suggesting that participants also valued students' motivation to learn and the ease with which content could be accessed and understood. Scalability was assigned the lowest priority (0.157), which may reflect the localised nature of the programme and its alignment with teacher professional development within a specific institutional context. Collectively, the prioritisation outcomes mirror stakeholder perceptions of what is most essential in programming instruction, and these weightings directly influenced the final ranking of the instructional methods under consideration.

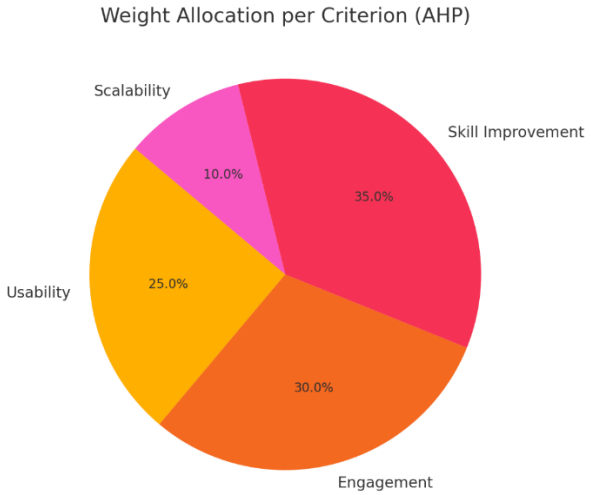


Fig. 14. Weight Allocation Per Criterion

The integration of statistical analysis with decision support mechanisms in this research has demonstrated the potential for more effective educational decision-making. Although the experimental findings confirmed that the mobile PBL approach significantly enhanced programming skills, the inclusion of the AHP contributed an additional layer of evaluative depth to the comparative analysis of instructional strategies. By allowing multiple pedagogical criteria such as usability, learner engagement, skill development, and scalability to be collectively assessed, AHP moves beyond traditional evaluative approaches that typically focus on a single outcome. This permits more nuanced, strategy-specific comparisons. For instructional designers, the implications are clear: the use of structured decision-making models during the course design phase enables a more context-sensitive and pedagogically aligned planning process. Rather than reverting to conventional instructional practices or adopting emerging technologies on the assumption of inherent superiority, decision models such as AHP support alignment with institutional goals, learner needs, and contextual priorities. In this regard, the mobile PBL application emerged as a scalable and effective model for fostering problem-solving capabilities in foundational programming modules.

Policymakers and educational programme implementers may also benefit from this approach, particularly when making informed decisions concerning resource distribution. The ability to quantitatively prioritise teaching methods enhances the justification for investing in mobile learning technologies where pedagogical merit is evident. Furthermore, the model offers a replicable framework for evaluating educational technologies in future planning cycles, particularly under constrained budgets. By applying such models, institutions can optimise resource allocation while maintaining educational quality and maximising learner outcomes. The study's findings deliver valuable guidance for a range of stakeholders including instructors, academic programme coordinators, and institutional decision-makers. These actors are frequently required to select instructional modes or technologies that can lead to meaningful improvements in student learning, particularly in technical subjects such as programming. Drawing upon empirical data from pre- and post-assessments, and supported by the structured evaluative capabilities of AHP, this research offers a rigorous yet accessible methodology for informed instructional decision-making. AHP enables the ranking of instructional methods based on a variety of educational dimensions, thereby reducing reliance on anecdotal preferences or single-metric evaluations. The evidence-based prioritisation afforded by this model provides a robust mechanism for curriculum planning, technological adoption, and resource allocation, ensuring that chosen instructional methods remain effective and contextually relevant over time.

6. Conclusion

In the field of programming, the ability to abstract problems is a fundamental step towards creating functional code. Therefore, equipping novice programmers with structured strategies to navigate the programming process and cultivate professional habits is of critical importance. This study aimed to evaluate the effectiveness of a mobile application designed using a PBL approach to foster programming proficiency. The application incorporates a problem-solving methodology that guides learners through educational tasks by engaging them in scientific reading, problem identification, and responding to guiding questions. This method encourages learners to develop autonomy and engage in reflective, analytical thinking. The application was developed through a user-centred design process that actively involved both students and academic staff. To determine the differences in performance between the experimental and control groups, multiple statistical techniques were applied. The results of the study indicate promising potential for extending the application to support additional programming languages such as Python, Java, and JavaScript,

enabling its applicability across broader educational contexts. Future longitudinal studies could examine the impact of the application on the long-term retention of programming skills, while comparative investigations could evaluate its effectiveness against other instructional models, including traditional lectures and flipped classrooms. Enhancing the application with adaptive features that respond to student progress could further personalise the learning experience. Moreover, the system may be adapted for use in specialised fields such as data science and cybersecurity, thereby assessing its flexibility across various domains within computer science education.

Importantly, this research contributes meaningfully to the literature in programming pedagogy by not only reporting evaluation outcomes but also incorporating rigorous empirical evidence. The inclusion of the AHP as a formal evaluative framework allowed for a structured comparison of instructional strategies based on multiple pedagogical criteria. This dual approach—merging experimental findings with expert-informed, multi-criteria evaluation—provides a sound foundation for decision-making in educational design. The model introduced through this research offers practical value for educators and curriculum developers seeking evidence-based justification for adopting specific teaching technologies or methods. The convergence of experimental data and AHP analysis reinforces the validity of the findings and supports strategic planning within programming education.

References

[1] Aires, J. P., Aires, S. B. K., Pereira, M. J., & Alves, L. M. (2023). Using the methodology problem-based learning to teaching programming to freshman students. http://hdl.handle.net/10198/28517

[2] Ali, S. S. (2019). Problem based learning: A student-centered approach. *English language teaching*, *12*(5), 73-78. http://dx.doi.org/10.5539/elt.v12n5p

[3] Allen, J. M., & Vahid, F. (2021). Concise graphical representations of student effort on weekly many small programs. Proceedings of the 52nd ACM Technical Symposium on Computer Science Education, 349-354. https://doi.org/10.1145/3408877.3432551

[4] Alshaye, I. A., Tasir, Z., & Jumaat, N. F. (2023). The effectiveness of online problem-based learning tasks on riyadh's secondary school students' problem-solving ability and programming skills. *Open Education Studies*, *5*(1), 20220208. https://doi.org/10.1515/edu-2022-0208

[5] Barg, M., Fekete, A., Greening, T., Hollands, O., Kay, J., Kingston, J. H., & Crawford, K. (2000). Problem-Based Learning for Foundation Computer Science Courses. http://dx.doi.org/10.1076/0899-3408(200008)10:2;1-C;FT109

[6] Barrows, H. S., & Tamblyn, R. M. (1980). *Problem-based learning : an approach to medical education*. Springer Pub. Co. https://cir.nii.ac.jp/crid/1130282273373464320

[7] Bawamohiddin, A. B., & Razali, R. (2017). Problem-based learning for programming education. *Learning*, *13*, 15. https://doi.org/10.18517/ijaseit.7.6.2232

[8] Condori-Obregon, P., Huallpa-Juarez, C., & Palomino-Vidal, C. (2025). Mobile application for distributing information to students at the Sciences and Humanities University. *Indonesian Journal of Electrical Engineering and Computer Science*, *37*(2), 1085-1092. http://doi.org/10.11591/ijeecs.v37.i2.pp1085-1092

[9] Derus, S. R. M., & Ali, A. Z. M. (2012). Difficulties in learning programming: Views of students. http://dx.doi.org/10.13140/2.1.1055.7441

[10] Duch, B. J., Groh, S. E., & Allen, D. E. (2001). *The power of problem-based learning : a practical "how to" for teaching undergraduate courses in any discipline* (1st ed.). Stylus Pub.

https://cir.nii.ac.jp/crid/1130000795577223552

[11] Faja, S. (2014). Evaluating Effectiveness of Pair Programming as a Teaching Tool in Programming Courses. *Information Systems Education Journal*, *12*(6), 36-45. http://isedj.org/2014-12/n6/ISEDJv12n6p36.html

[12] Hsu, Y.-C., & Ching, Y.-H. (2013). Mobile app design for teaching and learning: Educators' experiences in an online graduate course. *International Review of Research in Open and Distributed Learning*, *14*(4), 117-139. https://doi.org/10.19173/irrodl.v14i4.1542

[13] Ismail, M. N., Ngah, N. A., & Umar, I. N. (2010). Instructional strategy in the teaching of computer programming: a need assessment analyses. *The Turkish Online Journal of Educational Technology*, *9*(2), 125-131. https://www.researchgate.net/publication/228765283

[14] Janpla, S., & Piriyasurawong, P. (2018). The development of problem-based learning and concept mapping using a block-based programming model to enhance the programming competency of undergraduate students in computer science. *Tem Journal*, *7*(4), 708. https://www.ceeol.com/search/article-detail?id=717326

[15] Jumaat, N. F., & Tasir, Z. (2013). Integrating project based learning environment into the design and development of mobile apps for learning 2D-animation. *Procedia-Social and Behavioral Sciences*, *103*, 526-533. https://doi.org/10.1016/j.sbspro.2013.10.369

[16] Khakim, A. A. (2019). Problem-Based learning in programming lesson. 2nd International Conference on Intervention and Applied Psychology (ICIAP 2018), 529-536. https://doi.org/10.2991/iciap-18.2019.44

[17] Kusuma, F. I., Suryani, N., & Sumaryati, S. (2022). Mobile application-based media learning and its' effect on students' learning motivation. *International Journal of Evaluation and Research in Education*, *11*(3), 1353-1359. http://doi.org/10.11591/ijere.v11i3.22481

[18] N. M. Zahid, e. a. (2015). Problem based Learning through Mobile Application. *Innovative Practices in Higher Education Expo*. https://eprints.utm.my/62146/

[19] Nurbekova, Z., Grinshkun, V., Aimicheva, G., Nurbekov, B., & Tuenbaeva, K. (2020). Project-based learning approach for teaching mobile application development using visualization technology. *International Journal of Emerging Technologies in Learning (iJET)*, *15*(8), 130-143. https://doi.org/10.3991/ijet.v15i08.12335

[20] Powell, L. M., & Wimmer, H. (2016). Evaluating Students' Perception of Group Work for Mobile Application Development Learning, Productivity, Enjoyment and Confidence in Quality. *Information Systems Education Journal*, *14*(3), 85-95. http://isedj.org/2016-14/

[21] Prather, J., Reeves, B. N., Denny, P., Becker, B. A., Leinonen, J., Luxton-Reilly, A., Powell, G., Finnie-Ansley, J., & Santos, E. A. (2023). "It's weird that it knows what i want": Usability and interactions with copilot for novice programmers. *ACM transactions on computer-human interaction*, *31*(1), 1-31. https://doi.org/10.1145/3617367

[22] Richardson, J. T. (2011). Eta squared and partial eta squared as measures of effect size in educational research. *Educational research review*, *6*(2), 135-147. https://doi.org/10.1016/j.edurev.2010.12.001

[23] Rivera, M., Dávila, G., & Ruiz-Lizama, E. (2021). Design and development of an educational mobile application to optimize communication and interaction between members of educational institutions in real time. *Industrial Data*, *24*(1), 277-292. https://doi.org/10.15381/idata.v24i1.19421

[24] Rudder, A., Bernard, M., & Mohammed, S. (2007). Teaching programming using visualization. Proceedings of the Sixth IASTED International Conference on Web-Based Education, 487-492. https://www.researchgate.net/publication/228880750

[25] Sánchez-Morales, L. N., Alor-Hernández, G., Rosales-Morales, V. Y., Cortes-Camarillo, C. A., & Sánchez-Cervantes, J. L. (2020). Generating educational mobile applications using UIDPs identified by artificial intelligence techniques. *Computer Standards & Interfaces*, *70*, 103407. https://doi.org/10.1016/j.csi.2019.103407

[26] Simões, A., & Queirós, R. (2020). On the nature of programming exercises. *arXiv preprint arXiv:2006.14476*. https://doi.org/10.48550/arXiv.2006.14476

[27] Sung, Y.-T., Yang, J.-M., & Lee, H.-Y. (2017). The effects of mobile-computer-supported collaborative learning: Meta-analysis and critical synthesis. *Review of educational research*, *87*(4), 768-805. https://doi.org/10.3102/0034654317704307

[28] Turner, S., & Hill, G. (2007). Robots in problem-solving and programming. 8th Annual Conference of the Subject Centre for Information and Computer Sciences, 82-85. http://dx.doi.org/10.13140/2.1.2425.6800

[29] Yang, W., Li, H., Su, A., & Ding, L. (2023). Application of problem based learning (PBL) and case based learning (CBL) in the teaching of international classification of diseases encoding. *Scientific Reports*, *13*(1), 15220. https://doi.org/10.1038/s41598-023-42175-1