Design and Development of an Interactive Learning Application for Basic

Jaziyah Ramadhan Ali, Hudi Amhimmid Basheer Ateeyah, Masarrah Saeid Mousa Ibrahim

Department of Computer Science, University of Sabratha, Sabratha, Libya
E-mail: huda.eatia@sabu.edu.ly; Masarazr001@gmail.com

Corresponding Author: E-mail: jazia.zreg@sabu.edu.ly

Abstract:

Programming is a crucial skill in fields like AI, software development, and data science, driving high demand for qualified professionals. However, many students feel apprehensive about learning programming due to its abstract and complex nature. This project introduces a simplified, interactive educational application tailored for beginners in general and higher education. Featuring an integrated execution engine and an online code editor (PaizaCloud), the system delivers immediate feedback on coding errors, enabling students to learn through iterative, hands-on practice without frustration. Additionally, the platform empowers teachers to efficiently design practical lessons aligned with their curricula. Ultimately, the application simplifies complex concepts, boosts learning efficiency, and saves time. Future enhancements will focus on expanding the platform to cover advanced programming languages and modern software frameworks.

Keywords: MirrorCode; JavaScript; Monaco Editor; PaizaCloud.

Disclosure of Artificial Intelligence (AI) Tools: In this research, QuillBot was used as an artificial intelligence (AI) tool that relies on Natural Language Processing (NLP) and generative AI techniques to paraphrase texts and improve writing. It rewrites sentences in multiple ways while preserving the original meaning and correcting grammatical and spelling errors.

1. Introduction

The importance of acquiring programming skills has grown significantly across various domain contexts, driving an urgent need to design effective and engaging strategies that make learning to program an enjoyable experience. The era of traditional, one-way learning where students passively listen and are expected to memorize theoretical concepts verbatim is obsolete. Instructional design makes a tangible difference by determining the optimal ways to structure, divide, and deliver educational content (Ghofur and Youhanita, 2020).

Traditional teaching strategies often fall short in maintaining learner motivation and engagement, especially for beginners who face difficulties in grasping novel programming paradigms. To address this gap, the proposed system combines theoretical explanations with practical, real-time application using PaizaCloud a fast, cloud-based Integrated Development Environment (Cloud IDE) that operates directly within the web browser.

PaizaCloud supports multiple programming languages and frameworks (such as Python, Node.js, and PHP/LAMP), features a built-in terminal, offers AI-assisted functionality, and enables users to write, execute, and debug code instantly without installing software on their local devices. As a web-based platform, the application is cross-compatible and accessible across all smart devices and desktop computers.

The system enables learners to master fundamental coding principles such as variables, loops, and conditional statements while offering multi-language support and collaborative tools that encourage peer interaction and discussion. Applicable to both general and higher education levels, the platform enables students to engage with interactive content through activities, quizzes, immediate feedback, and personalized learning pathways (Ghofur and Youhanita, 2020).

1.1 Literature Review

To contextualize the system within current academic research, several key studies on adaptive and technology-enhanced learning were analyzed:

- Čep et al. (2025) analyzed a total of sixteen studies using thematic synthesis to identify key methodological trends, outcomes, and emerging themes. The findings reveal a strong preference for mixed-methods

approaches, combining system usage data with surveys, interviews, and performance metrics to capture both measurable impacts and user experiences. AI integration, real-time feedback, and gamification were identified as key enablers of adaptive learning system (ALS) success. The review highlights a clear trend toward scalable, learner-centered ALS solutions in academia and identifies the need for robust evaluation frameworks and ethical governance.

- Similarly, Dinata et al. (2024) developed an interactive learning application utilizing the Technological Pedagogical and Content Knowledge (TPACK) framework, incorporating a pedagogical paradigm in its design through simulation elements and animated visuals.
- Nasution and Nusa (2021) developed a UI/UX design prototype of a learning web application named "IdeIn" using the Design Thinking method, which consists of empathize, define, ideate, prototype, and test stages. The main focus of this learning web application is to provide an opportunity for the whole community to support education in Indonesia by conducting online classes.

2. Methodology

The research methodology relies on designing and developing the application using modern software engineering principles and web frameworks, utilizing Python and Java as primary languages. Featuring an integrated execution engine and an online code editor (PaizaCloud), the system delivers immediate feedback on coding errors, enabling students to learn through iterative, hands-on practice without frustration. Additionally, the platform empowers teachers to efficiently design practical lessons aligned with their curricula. Ultimately, the application simplifies complex concepts. Furthermore, QuillBot was utilized as an artificial intelligence tool that relies on Natural Language Processing (NLP) and generative AI techniques. The system's effectiveness is evaluated by deploying it to a target sample of beginners, where empirical data are collected through structured questionnaires and assessment tests, and then statistically analyzed to measure learning efficacy.

2.1 System Analysis and Design

The analysis and design phase is a critical foundation in the Software Development Life Cycle (SDLC). This phase encompasses defining the software architecture, network infrastructure, database schema, file systems, user interface layouts, and exact data flow specifications.

2.2 Technical Architecture and Tools

The system utilizes a robust set of tools and technologies to deliver a seamless, interactive experience:

- **Backend Infrastructure:** JavaScript with Node.js is utilized on the server side for processing client requests and executing core business logic.
- **Frontend Interface:** Built using modern HTML5, CSS3, and the Bootstrap framework to ensure responsive, accessible, and visually clean UI across all device screen sizes.
- **Database Management:** MySQL serves as the central relational database management system, handling user data, course records, and progress logs efficiently.
- **Interactive Code Editors:** Integrated code editors, specifically Monaco Editor and MirrorCode, permit users to write, run, and preview code output directly within the browser UI.
- **Development Environment:** Visual Studio Code was used as the primary local IDE during development due to its rich debugging tools and extensible ecosystem.

2.3 Database Design

The system database (learning platform) consists of five logical tables connected via relational constraints. Tables 1, 2, and 3 detail the main entity structures:

Table (1): Structure of Users Table

Field Name	Type	Length	Key	Description
UserID	INT	11	PK	Unique identifier for user
Username	VARCHAR	50	Unique	User account handle
Password	VARCHAR	100	-	Hashed password
Role	ENUM	-	-	User role ('User', 'Admin')

Table (2): Structure of Lessons Table

Field Name	Type	Length	Key	Description
LessonID	INT	11	PK	Unique identifier for lesson
Title	VARCHAR	100	-	Lesson title
Content	TEXT	-	-	Detailed lesson instructional body
CreatedBy	INT	11	FK	Author ID (FK referencing Users)

Table (3): Structure of User Progress Table

Field Name	Type	Key	Description
ProgressID	INT (11)	PK	Unique progress entry record ID
UserID	INT (11)	FK	FK referencing Users table
LessonID	INT (11)	FK	FK referencing Lessons table
Status	ENUM	-	Lesson state ('Not_Started', 'In_Progress', 'Completed')
CompletionDate	DATE	-	Date of lesson completion

2.4 Implementation and System Features

The system was successfully built along with all interactive interfaces. Key features include:

- **Embedded Interactive Code Editor:** Enables learners to write code (HTML/JS/Python) and view real-time rendering in a live output panel.
- **Instant Feedback System:** When code contains structural or syntax errors, diagnostic suggestions guide the student toward self-correction.
- **Admin Dashboard:** Provides administrators with seamless tools for managing course material rapidly.

2.5 Core System Features

- **Course & Lesson Catalog:** Organized structured lists categorized by programming language, allowing learners to easily navigate and access targeted content.
- **Embedded Live Code Editor:** Embedded code editor supporting HTML, JavaScript, and Python. Features a live execution window showing output immediately upon code execution.
- **Automatic Progress Tracking:** Automatically records user code state and lesson completion metrics, ensuring seamless continuation across sessions.
- **Administrative Dashboard:** A comprehensive administrative control panel for instructors to publish new lessons, modify existing content, manage users, and curate curriculum materials

2.6 Testing Methodology and Field Evaluation

The system was evaluated through three complementary forms of testing:

- **Functional Testing:** Verifying core operations including registration, lesson navigation, code execution, and data storage.
- **Non-Functional Testing:** Measuring editor execution speed, interface usability, and data security.
- **User Field Testing:** Deploying the platform to a sample of novice computer science students and gathering feedback via standardized surveys.

3. Results and Discussion

3.1 Results Analysis

The evaluation and user feedback revealed the following key findings: Table 4 summarizes the survey results.

Table (4): User evaluation of the application

Survey Item	Positive Rating (%)	Notes
Interface clarity and ease of use	60%	
Effectiveness of hints and guidance	70%	
Visual design attractiveness	70%	
Ease of navigation between lessons/pages	75%	
Accessibility of programming tools	60%	
Application stability and performance	45%	Lowest-rated aspect
Support for understanding programming concepts	75%	
Integration of code editor with explanations	80%	Highest-rated aspect
Live code preview window	65%	
Short quizzes and assessments	80%	Highest-rated aspect
Overall positive evaluation	85%	Excellent + Very Good

- Over 70% of respondents confirmed that integrating the live code editor significantly facilitated their comprehension of abstract concepts compared to static lecture notes.
- Most participants commended the instant feedback feature, which drastically reduced debugging time and fostered independent learning.
- The survey sample reported high satisfaction regarding UI/UX clarity and intuitive navigation across lessons and quizzes.

3.2 Discussion

Previous studies involved the design of adaptive learning systems (ALS) based on immediate feedback (Čep et al., 2025) and the development of interactive systems grounded in the Technological Pedagogical Content Knowledge (TPACK) framework to implement a model incorporating simulation and animation elements (Dinata et al., 2024); additionally, a preliminary educational model (IdeIn) was designed using the design thinking approach (Nasution and Nusa, 2021). In our study, we developed an interactive code editor system featuring immediate feedback; it allows learners to write code (HTML, JS, or Python) and preview it in a live output panel, while providing instant diagnostics regarding structural and syntax errors to guide students toward self-correction.

4. Conclusion

This study demonstrates that shifting from passive traditional instruction to direct interactive practice is essential for novice programming education. The research successfully presented a comprehensive Arabic platform combining tutorial explanations, live coding, and automated diagnostic feedback, significantly enhancing learning outcomes while lowering initial entry barriers.

4.1 Recommendations

Based on empirical findings and field trials, the researcher recommends:

- Expanding application scope to include advanced tracks and extra languages (e.g., Python, Java, Data Structures).
- Developing dedicated mobile/tablet applications to improve accessibility anywhere.
- Optimize the code engine performance and rendering pipeline for low-spec hardware and constrained network conditions to guarantee a smooth, accessible experience for all learners.
- Integrating AI algorithms to deliver tailored adaptive feedback matching individual learning paces.
- Expand the educational content to cover intermediate and advanced programming concepts and popular frameworks, facilitating long-term progressive learning pathways.

References

Ghofur, A. and Youhanita, E. (2020). "*Interactive media development to improve student motivation*", IJECA (International Journal of Education and Curriculum Application), 3 (1), pp. 1–6.

Kose, U., Koc, A. and Yucesoy, A. (2013). "*Design and development of a sample 'computer programming' course tool via story-based e-learning approach*", Educational Sciences: Theory & Practice, 13 (2), pp. 1235–1250.

Nasution, W.S.L. and Nusa, P. (2021). "*UI/UX design web-based learning application using design thinking method*", Ars: Jurnal Seni Rupa dan Desain, 24 (1), pp. 24–35.

Radan, P. and Design, M.S.E. (2020). "*Design and development of an interactive educational app for learning programming*", International Journal of Computer Science and Mobile Computing, 9 (5), pp. 112–118.

Čep, A., Bernik, A. and Tomičić, I. (2025). "*Adaptive learning systems in higher education: Challenges, trends, and outcomes*", Journal of Information and Organizational Sciences.