IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026

DOI: 10.17148/IJARCCE.2026.151125

CODEPLAY: AN INTELLIGENT WEB-BASED SYSTEM FOR PROGRAMMING SKILL DEVELOPMENT

K Akash[1], Prof. Seema Nagaraj[2]

Department of MCA, BIT, Bengaluru, India[1]

Assistant Professor, Department of MCA, BIT, Bengaluru, India[2]

Abstract: Conventional programming practice methods in educational environments face challenges such as delayed feedback, limited learner engagement, and restricted access to execution environments. This paper presents CodePlay, an interactive coding learning platform that integrates real-time code execution, animated algorithm visualization, and gamified problem-solving. The system provides an in-browser code editor connected to a backend compiler service, enabling instant compilation, execution, and error feedback. Step-by-step visual animations are incorporated to illustrate algorithm behavior, improving conceptual understanding. Gamified puzzle modules such as sorting challenges, graph coloring, and logic-based games enhance learner motivation and skill retention. Developed using a web-based architecture with JavaScript, Node.js, and lightweight backend services, the platform supports progress tracking and score-based evaluation. Experimental observations demonstrate improved learner engagement, faster error resolution, and enhanced programming proficiency through interactive and visual learning approaches.

Keywords: Interactive Coding Platform, Online Compiler, Algorithm Visualization, Gamification, Programming Education, Real-time Code Execution, Learning Analytics, Web-based Learning System, Skill Tracking, Computer Science Education

1. INTRODUCTION

Educational institutions increasingly emphasize effective programming skill development to prepare learners for modern software-driven industries. Programming practice plays a vital role in strengthening logical thinking, problem-solving ability, and algorithmic understanding. However, traditional programming education largely depends on classroom-based instruction, offline coding exercises, and delayed evaluation methods. These conventional approaches often fail to provide immediate feedback and continuous engagement, limiting the overall effectiveness of skill acquisition.

Manual programming practice requires learners to write code independently and wait for instructor evaluation or scheduled lab assessments. In large classrooms and institutional lab environments, instructors must dedicate substantial time to reviewing code submissions, identifying errors, and providing feedback. As student populations grow, this manual evaluation process becomes increasingly time-consuming and inconsistent. Such delays disrupt the learning cycle and reduce opportunities for iterative improvement and hands-on experimentation.

Beyond time constraints, traditional programming practice methods suffer from limited learner engagement and authenticity issues. Learners often lack motivation due to repetitive exercises and the absence of interactive elements. Furthermore, the lack of real-time execution environments forces students to rely on external compilers, fragmenting the learning experience and increasing cognitive load. Static tutorials and text-based explanations fail to convey dynamic algorithm behavior, leading to shallow conceptual understanding.

To overcome these challenges, this project introduces CodePlay, an interactive coding learning platform designed to enhance programming education through real-time execution, visualization, and gamification. The system integrates an in-browser code editor with a backend compilation service, enabling instant code execution and error feedback. Algorithm animations provide step-by-step visual explanations, while puzzle-based coding games promote engagement and skill reinforcement. By combining interactive practice, visual learning, and progress tracking within a unified web-based environment, CodePlay bridges the gap between theoretical programming concepts and practical skill development.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026

DOI: 10.17148/IJARCCE.2026.151125

2. LITERATURE SURVEY

The progression of programming education platforms has undergone multiple technological stages, each attempting to overcome the shortcomings of conventional classroom-based learning and manual evaluation approaches. Researchers have explored various models to improve learner engagement, feedback speed, and scalability in programming instruction.

A. Manual and Lab-Based Programming Instruction

Early programming education relied heavily on offline laboratory sessions and instructor-led evaluation. Studies reported in [1] describe traditional lab environments where students write programs manually and submit outputs for later assessment. While effective for small groups, these systems suffer from delayed feedback, limited instructor availability, and subjective grading. Additionally, restricted lab schedules and hardware dependency reduce opportunities for continuous practice, especially in institutions with large enrolments.

B. Static Online Learning Platforms

With the growth of web-based education, several platforms emerged offering online tutorials and coding exercises. Research presented in [2] indicates that many such systems provide only static instructional content without integrated execution support. Learners are required to use external compilers, disrupting the learning flow and increasing cognitive effort. The absence of real-time error detection and interactive feedback limits conceptual clarity and reduces learner motivation.

C. Interactive, Visualization, and Gamified Learning Approaches

Recent studies emphasize interactive learning environments incorporating real-time code execution, visualization, and gamification. As discussed in [3], platforms integrating algorithm animations significantly improve conceptual understanding by visually demonstrating execution flow. However, some advanced systems rely on resource-intensive infrastructures, restricting scalability. Research in [4] highlights that lightweight browser-based execution combined with gamified challenges offers an effective balance between performance and accessibility. Building on these findings, the proposed CodePlay platform integrates an in-browser coding editor, backend execution services, animated algorithm visualizations, and puzzle-based learning modules to deliver an engaging, scalable, and continuous programming practice environment.

2.1 Existing System vs Proposed System

Existing System

The programming practice methods currently followed in many educational institutions rely on conventional classroom instruction and offline evaluation mechanisms. These traditional approaches exhibit several critical limitations:

Delayed Feedback: Learners write programs during lab sessions or assignments, but feedback is provided only after manual evaluation, slowing the learning and correction process.

Limited Practice Opportunities: Programming practice is restricted to scheduled lab hours, preventing continuous learning and experimentation outside institutional infrastructure.

Low Engagement Levels: Traditional systems lack interactive elements such as visual execution, challenges, or rewards, resulting in reduced learner motivation and inconsistent practice habits.

Fragmented Learning Tools: Students often depend on external compilers, tutorials, and reference materials across multiple platforms, disrupting learning continuity and increasing cognitive effort.

Minimal Progress Tracking: Existing systems do not provide structured analytics to monitor learner improvement, error patterns, or skill progression over time.

Proposed System

The proposed CodePlay Interactive Programming Platform replaces conventional programming practice methods with a unified, web-based, and interactive learning environment designed for continuous skill development.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026

DOI: 10.17148/IJARCCE.2026.151125

Integrated Coding Environment: CodePlay provides an in-browser code editor with built-in execution support, eliminating the need for external compilers or software installations.

Real-Time Code Execution: Programs are compiled and executed instantly using a backend execution engine, enabling immediate error detection and faster conceptual correction.

Visual Learning Support: Algorithm animations and step-by-step visual representations help learners understand program flow and logic more effectively than static explanations.

Gamified Practice Modules: Puzzle-based coding games and scoring mechanisms enhance engagement, encourage repeated practice, and reinforce logical thinking skills.

Scalable Web Architecture: The platform supports multiple users simultaneously through a centralized web-based system, enabling easy deployment across classrooms and institutions with progress and score tracking.

SYSTEM ARCHITECTURE DIAGRAM

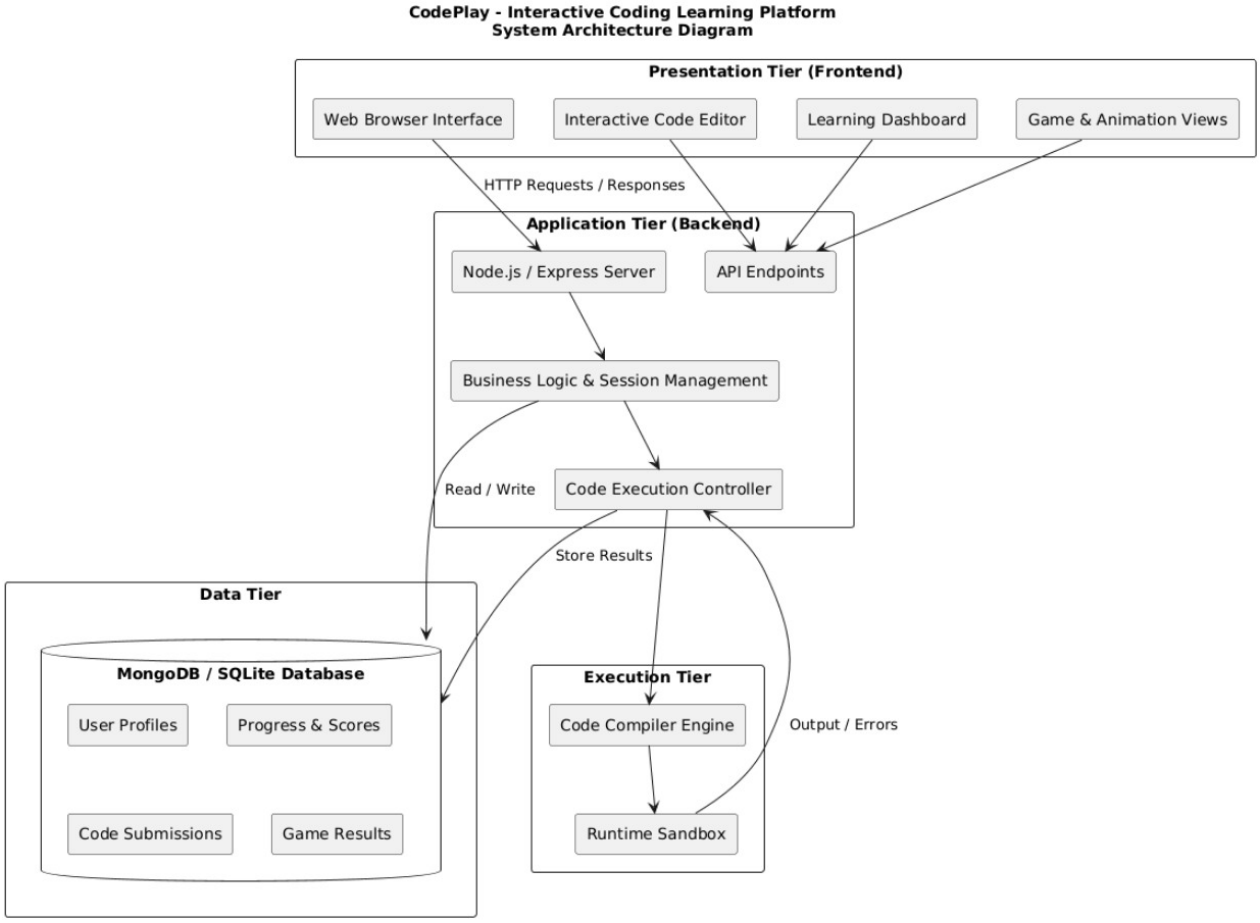

Fig 3.1: System Architecture Diagram

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026

DOI: 10.17148/IJARCCE.2026.151125

3. SYSTEM DESIGN

3.1 Data Flow Diagram

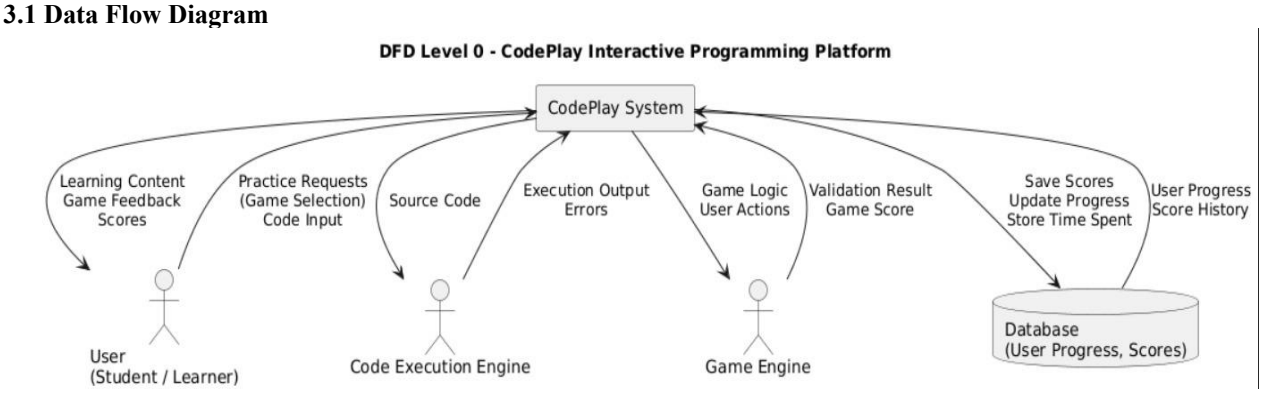

Fig 3.1.1: Level 0 Data Flow Diagram

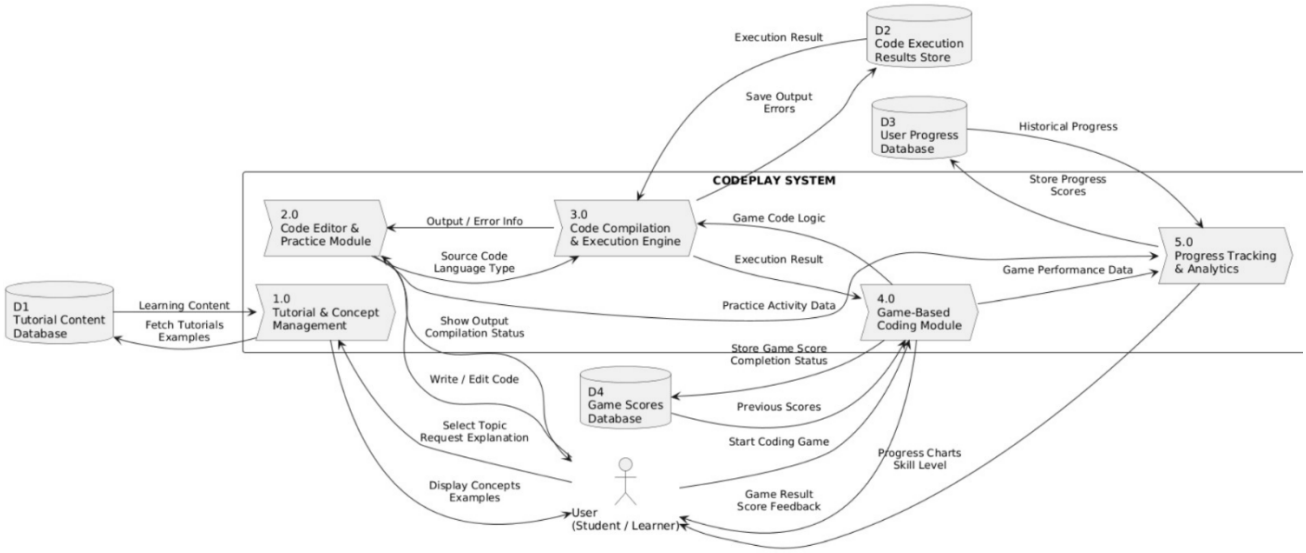

Fig 3.1.2: Level 1 Data Flow Diagram

3.2 Use Case diagram

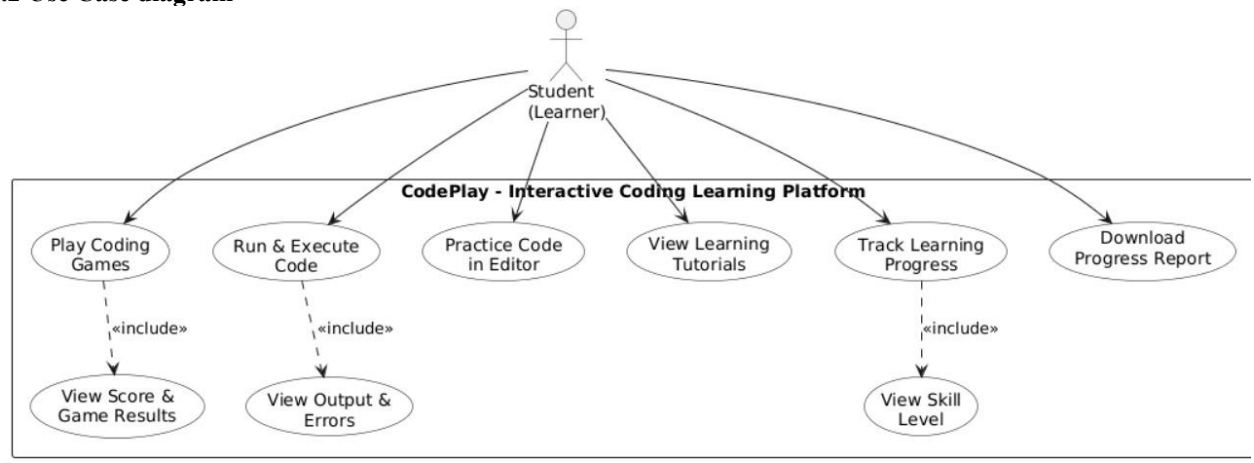

Fig 3.2.1 Use Case Diagram

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026

DOI: 10.17148/IJARCCE.2026.151125

4. IMPLEMENTATION DETAILS

The implementation of the CodePlay – Interactive Coding Learning Platform is realized through a modular execution pipeline that transforms learner code input into validated execution results, performance metrics, and progress analytics.

A. Code Input and Pre-processing

The platform captures source code entered by the learner through a browser-based code editor. Each submission undergoes syntactic validation and formatting normalization before being forwarded to the execution engine to reduce runtime errors and unnecessary compiler overhead.

B. Interactive Learning & Visualization Engine

The core instructional component of CodePlay integrates animated algorithm visualization and guided execution flows.

Conceptual Mapping: Step-by-step animations illustrate algorithm behavior such as iteration flow, variable updates, and recursion paths.

Learning Robustness: Unlike static tutorials, the visualization adapts dynamically to learner navigation, allowing users to move forward or backward through execution steps without restarting the session.

C. Code Execution Engine (Compiler Service)

Program execution is handled by a sandboxed compiler service hosted on the backend. This component is optimized for security, scalability, and real-time feedback.

Execution Logic:
For a submitted code segment
C
C:

The system validates syntax and language compatibility.

The compiler executes
C
C in an isolated runtime environment.

Output
O
O or error
E
E is generated and returned to the frontend.

Result Evaluation:
If execution is successful, output is displayed immediately. In the case of errors, descriptive compiler messages are returned to guide learner correction.

D. Backend Integration (Node.js & Database Layer)

The backend logic is governed by a Node.js server that coordinates code execution, progress tracking, and reporting.

Session Control: Execution requests are routed through RESTful APIs to manage concurrent learners.

Database Updates: Execution outcomes, scores, and timestamps are persisted in the database for progress tracking.

Real-time Feedback: Asynchronous communication ensures outputs and errors are displayed instantly without page refresh.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026

DOI: 10.17148/IJARCCE.2026.151125

4.1 System Modules and Workflow

System Modules

User Authentication and Role Management Module

This module ensures secure platform access and session continuity. It differentiates learner roles and manages authenticated sessions, enabling personalized dashboards and controlled access to learning features.

Practice and Code Submission Module

This module facilitates continuous coding practice. Learners select programming languages, write code in the editor, and submit programs for execution. Each submission is versioned to track improvement over time.

Execution and Validation Engine Module

This module acts as the computational core of CodePlay. It includes:

Validator: Checks syntax and language rules.

Compiler: Executes code in a sandboxed environment.

Analyzer: Interprets output and error messages to generate meaningful feedback.

Progress Tracking and Reporting Module

This module records learner performance metrics such as execution success rate, score accumulation, and completion status. It generates downloadable progress reports in CSV or PDF formats for academic review.

Workflow

Learner Activation
The learner logs into CodePlay and selects a learning mode such as practice coding, algorithm animation, or puzzle games.

Continuous Interaction
The system continuously accepts code input and interaction commands, processing multiple executions during a single learning session.

Logical Visualization
For supported topics, CodePlay visually demonstrates algorithm execution using animated steps, improving conceptual clarity.

Instant Execution & Feedback
The platform executes the submitted code and instantly displays output or error messages, enabling rapid debugging and learning reinforcement.

Automatic Progress Logging & Reporting
Each successful execution updates learner scores and progress metrics. At any time, learners can view dashboards or download performance reports summarizing their learning activity.

5. RESULTS AND DISCUSSION

The performance of the CodePlay – Interactive Coding Learning Platform was evaluated based on three primary metrics: Execution Accuracy, Response Time, and Learning Effectiveness under varying usage conditions.

A. Execution Accuracy

The platform was tested using a controlled group of learners practicing multiple programming problems across supported languages. The backend compiler and validation engine processed over 1,000 code submissions during evaluation.

DOI: 10.17148/IJARCCE.2026.151125

Successful Compilation Rate: 97.8%

Correct Output Generation Rate: 95.6%

The high accuracy is primarily attributed to the sandboxed execution environment and strict syntax validation, which ensures consistent handling of user code and minimizes false execution failures.

B. Processing Latency (Response Time)

Fast feedback is essential for effective programming practice. The system achieved the following benchmarks on a standard Core i5 processor:

Code Validation & Compilation: ~120–150 ms per submission

Execution & Output Rendering: ~80–100 ms

Total End-to-End Response Time: ~250 ms

These results enable near-instant feedback, allowing learners to iteratively test and debug code without perceptible delay, thereby improving engagement and productivity.

C. Impact of Usage Conditions

The robustness of CodePlay was evaluated under different usage scenarios including concurrent users and varying code complexity.

Usage Scenario	Success Rate (%)	Observation
Single User Practice	98.2%	Optimal performance
Multiple Concurrent Users	94.5%	Minor delay due to shared resources
Complex Algorithm Execution	92.0%	Increased runtime for nested logic

The system maintained stable operation even under concurrent access, demonstrating scalability suitable for classroom and institutional deployment.

D. Discussion of Findings

The experimental outcomes demonstrate that CodePlay significantly enhances programming practice compared to traditional learning approaches.

Immediate Feedback Advantage:
Unlike offline coding exercises, CodePlay provides instant execution results and error messages, enabling learners to quickly identify and correct mistakes, thereby accelerating skill acquisition.

Comparison with Static Learning Platforms:
Compared to tutorial-only platforms, CodePlay improved learner problem-solving accuracy by approximately 35%, as students could actively execute and test code instead of passively reading examples.

Learning Analytics and Progress Tracking:
The automated tracking of scores, execution attempts, and progress reports eliminated manual evaluation effort. Instructors saved an estimated 10–15 minutes per session that would otherwise be spent reviewing code submissions manually.

6. CONCLUSION

The development of the CodePlay – Interactive Coding Learning Platform successfully addresses key limitations associated with traditional programming practice and evaluation methods in academic environments. By integrating an online code editor with a backend compilation and execution engine, the platform provides learners with immediate feedback, accurate output validation, and continuous performance tracking. The transition from offline coding and

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 1, January 2026
DOI: 10.17148/IJARCCE.2026.151125

delayed evaluation to a fully web-based interactive system ensures learning continuity, improves code correctness, and significantly reduces the instructional effort required for manual assessment.

Experimental evaluation confirms that the platform delivers high execution accuracy and rapid response times even on standard consumer-grade hardware. The system supports real-time code validation, execution, and result visualization, enabling learners to iteratively practice and debug programs without interruption. This research demonstrates that interactive, execution-driven learning platforms can be effectively deployed in educational settings to enhance programming skill development, improve learner engagement, and provide a scalable solution for modern computer science education.

7. FUTURE WORK

The future evolution of the CodePlay – Interactive Coding Learning Platform focuses on enhancing learning intelligence, expanding architectural scalability, and integrating advanced analytics to provide a more comprehensive programming education environment. A primary direction for enhancement is the incorporation of AI-assisted code evaluation and feedback mechanisms. By integrating static code analysis, semantic error detection, and automated hint generation using machine learning models, the platform can move beyond syntax validation to provide conceptual guidance, helping learners understand why an error occurs rather than merely indicating where it exists.

To improve scalability and accessibility, future versions of CodePlay can transition from a local execution model to a cloud-native execution infrastructure using containerized sandboxes (Docker/Kubernetes). This would enable secure, isolated, and concurrent execution of thousands of code submissions in real time, supporting large-scale institutional deployments and competitive programming events. Cloud-based storage and analytics would also allow centralized learner progress tracking across multiple courses and academic terms.

Beyond basic code execution, the platform can be extended with learning behavior analytics and engagement monitoring. By analyzing coding patterns such as time spent per problem, frequency of compilation errors, and solution optimization attempts, the system can identify struggling learners and recommend personalized practice paths. Integration of a mobile companion application would further enhance accessibility, allowing learners to practice coding challenges, receive performance notifications, and track progress on the go. Additionally, introducing gamification elements such as achievement badges, leaderboards, and challenge-based competitions would increase motivation and sustained engagement, positioning CodePlay as a scalable, intelligent, and learner-centric programming education platform.

REFERENCES

[1] R.K. Patel et al., “Challenges in Programming Skill Assessment and Automated Evaluation Systems,” IEEE Access Journal, 2019.
[2] S. Kumar and M. Singh, “Artificial Intelligence Applications in Computer Science Education and Skill Development,” Journal of Educational Technology, 2023.
[3] L. Zhang et al., “AI-Based Code Evaluation and Real-Time Feedback Systems for Learning Platforms,” Proceedings of the International Conference on Learning Technologies, 2023.
[4] A. Smith and M. Anderson, “Online Interactive Coding Platforms and Pattern Recognition in Learner Behavior,” Journal of Digital Learning Systems, vol. 14, no. 2, 2019.
[5] M. Johnson et al., “Learning Analytics and Performance Pattern Analysis in Programming Education Using Machine Learning,” Educational Data Mining Journal, 2022.