IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 12, April 2026
Impact Factor: 8.2

Code Insight: An AI-Powered Code Evaluation and Learning Platform Using a Hybrid Intelligence Approach

Mr. Pratik A. Patil, Mr. Mandar P. Deshmukh, Mr. Uday K. Jadhav, Mr. Tanish A. Tambe
Students, Department of Computer Engineering,
Late G. N. Sapkal College of Engineering, Nashik

Abstract: *In modern technical education, irregular student attendance and limited hands-on practice often lead to poor programming proficiency, especially in practical examinations. To address this issue, this paper presents the design and implementation of an AI-driven code learning and evaluation platform that enables continuous coding practice independent of classroom presence. The proposed system allows academic institutions to upload problem statements, facilitating structured learning for students. The platform integrates automated code execution with an AI-based analysis module powered by a large language model, which evaluates submitted programs for time and space complexity and provides optimization suggestions. Additionally, the system maintains a repository of student submissions and incorporates a gamified leaderboard mechanism to encourage engagement and performance improvement. The backend is developed using Spring Boot with JPA and Hibernate for efficient data management, while the frontend utilizes React and Tailwind CSS for an interactive user experience. Experimental usage demonstrates that the system enhances coding consistency, provides meaningful feedback, and promotes competitive learning among students. The proposed solution bridges the gap between theoretical learning and practical skill development in programming education*

Keywords: AI – Assisted Code Evaluation, Computational Complexity Estimation, Continuous Coding Practice Platform, Pretrained LLMs

I. INTRODUCTION

Programming skills have become essential in modern technical education and the software industry. However, many students struggle to achieve proficiency due to irregular practice, limited access to structured learning environments, and lack of continuous feedback. Traditional teaching methods and existing coding platforms often focus mainly on program correctness, providing little insight into code efficiency, structure, or optimization. This creates a gap between writing functional code and understanding how to improve its performance in real-world scenarios.

To address these challenges, this paper presents an AI-powered code learning and evaluation platform that supports continuous and self-paced learning. The system enables institutions to assign coding problems while allowing students to practice independently. Submitted programs are automatically evaluated for correctness and performance within a controlled environment, ensuring immediate feedback.

A key contribution of this work is the integration of a hybrid AI approach. The platform combines a fine-tuned code-centric language model for structural analysis and complexity estimation with a generative AI component that produces clear, human-readable explanations and optimization suggestions. This combination ensures both analytical accuracy and intuitive understanding, helping users improve their coding skills effectively.

Additionally, the system maintains a repository of submissions to track student progress over time and incorporates a leaderboard mechanism to encourage engagement through healthy competition. By integrating automated evaluation, hybrid AI-driven analysis, and educational features, the proposed platform aims to bridge the gap between theoretical learning and practical coding proficiency, ultimately fostering more efficient and skilled programmers.

DOI: 10.48175/IJARSCT-33745

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 12, April 2026
Impact Factor: 8.2

The platform is implemented using a modern full-stack architecture, featuring a responsive frontend interface and a robust backend supported by efficient data management systems. By integrating automated code execution, hybrid AI-driven analysis, and educational support features, the proposed system aims to bridge the gap between theoretical understanding and practical application. The overarching goal is to create a scalable and learner-centric solution that enhances coding proficiency by enabling students to not only write correct programs but also develop an understanding of efficient and optimized coding practices.

II. LITERATURE REVIEW

Early work in automated code assessment primarily focused on rule-based systems and test case validation, where student programs were evaluated based on correctness of output against predefined inputs. While such systems proved effective in large-scale environments like online judges, they were limited in their ability to provide qualitative feedback regarding code efficiency and structure. Researchers have identified that merely validating correctness is insufficient for developing strong programming skills, as students also need guidance on optimization and best practices.

To overcome these limitations, static code analysis techniques were introduced, which examine source code without executing it. These approaches utilize syntax trees and predefined rules to detect inefficiencies, code smells, and potential errors. Although static analysis tools provide deeper insights compared to simple output matching, they often lack adaptability and struggle to handle diverse coding styles. Additionally, their feedback is typically rigid and may not be easily interpretable by novice programmers.

With the emergence of machine learning and natural language processing, recent research has shifted towards intelligent systems capable of understanding code semantics. Several studies have demonstrated the use of neural networks and deep learning models to predict code complexity, detect bugs, and recommend optimizations. More recently, large language models have shown promising results in analyzing programming logic and generating human-like explanations. These models leverage vast amounts of training data to understand patterns in code, enabling more flexible and context-aware evaluation compared to traditional methods.

Parallel to advancements in code analysis, educational platforms have also evolved to support continuous learning. Learning management systems and online coding platforms provide students with access to practice problems, submission tracking, and performance analytics. Gamification techniques, such as leaderboards and point-based scoring, have been widely adopted to increase student engagement and motivation. Studies indicate that such features encourage consistent participation and foster a competitive yet collaborative learning environment.

Despite these advancements, there remains a lack of integrated systems that combine real-time code execution, AI-driven complexity analysis, and continuous learning support within a single platform. Many existing solutions either focus solely on code correctness or provide limited feedback on performance optimization. Furthermore, few systems explicitly address the challenge of irregular student attendance by enabling uninterrupted access to structured coding practice.

The proposed system aims to bridge this gap by integrating automated code execution with AI-based complexity evaluation and a structured learning environment. By leveraging generative AI for code analysis and incorporating features such as problem management, submission tracking, and leaderboard-based engagement, the system provides a comprehensive solution for improving programming skills. This approach not only enhances technical learning but also aligns with modern educational requirements for flexibility, scalability, and personalized feedback.

III. PROPOSED SYSTEM

The proposed system is an AI-driven code learning and evaluation platform designed to facilitate continuous programming practice, automated execution, and intelligent feedback generation.

1. System Overview

The platform follows a multi-layered architecture consisting of a frontend interface, a backend processing layer, a database management system, and an AI-based analysis module. The frontend is developed using modern web

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 12, April 2026
Impact Factor: 8.2

technologies to provide an interactive coding environment where users can write, edit, and submit programs. The backend, built using a robust framework, handles request processing, code execution, and communication with the AI service. A relational database is used to store user data, problem statements, and submission history.

The system is designed to support two primary user roles: administrators and students. Administrators can upload and manage problem statements, while students can attempt these problems, submit their code, and view feedback along with performance metrics.

2. Methodology and Workflow

The working of the system is based on a structured pipeline that ensures seamless interaction between different components. The overall workflow can be described in the following stages:

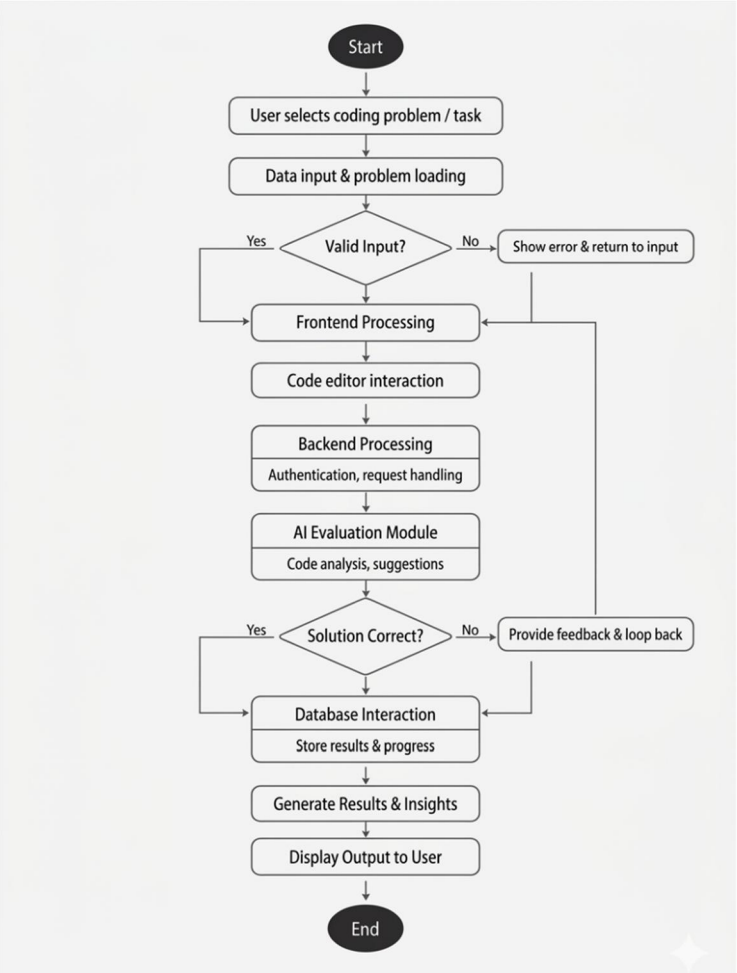

Fig. 1. Control Flow Chart

Step 1: Problem Assignment

Educational institutions or instructors upload coding problems to the platform through the administrative interface. Each problem includes a description, input-output format, and optional test cases. These problems are stored in the database and made accessible to students.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 12, April 2026
Impact Factor: 8.2

Step 2: Code Submission

Students log into the platform and select a problem to solve. Using the integrated code editor, they write their solution in Java and submit it for evaluation. The submitted code is transmitted to the backend server via secure API calls.

Step 3: Code Execution

Upon receiving the code, the backend initiates the execution process. The submitted program is first compiled using a Java compiler. If compilation is successful, the program is executed with predefined or user-provided inputs. The system captures the output, runtime errors, and execution status. To ensure system stability, execution constraints such as time limits and resource usage restrictions are applied.

Step 4: AI-Based Code Analysis

After execution, the code is processed through a hybrid AI module for advanced evaluation. In the first stage, a fine-tuned code-centric model analyzes the program structure to estimate time and space complexity and detect inefficiencies. In the second stage, the results are passed to a generative AI component via an API, which converts the analysis into clear, human-readable explanations and optimization suggestions.

This hybrid approach combines accurate structural analysis with intuitive feedback, providing more effective and context-aware insights than traditional methods.

Step 5: Result Processing and Storage

The results obtained from both execution and AI analysis are aggregated by the backend. These include program output, correctness status, complexity estimation, and suggested improvements. The complete submission data is then stored in the database, allowing for future reference and performance tracking.

Step 6: Feedback and Visualization

The processed results are sent back to the frontend, where they are presented to the user in an organized manner. Students can view execution output, identify errors, and understand optimization suggestions provided by the AI. This immediate feedback loop helps in improving coding skills iteratively.

Step 7: Performance Tracking and Gamification

To encourage continuous engagement, the platform implements a scoring mechanism based on correctness, efficiency, and consistency. Points are awarded to users for successful submissions, and a leaderboard is maintained to rank students based on their performance. This gamified approach promotes healthy competition and motivates users to practice regularly.

3. Key Methodological Features

One of the distinguishing aspects of the proposed system is the integration of real-time code execution with AI-based evaluation. Unlike conventional platforms that rely solely on test case validation, the system provides qualitative insights into code efficiency and structure. The use of generative AI allows for flexible and human-like explanations, making it easier for students to understand their mistakes and improve their solutions.

Another important feature is the support for continuous learning. By allowing students to access problem statements and submit solutions at any time, the system eliminates dependency on classroom attendance. The inclusion of submission history and performance analytics further enhances the learning experience by enabling self-assessment.

IV. SYSTEM ARCHITECTURE

The proposed system is designed using a modular and scalable architecture that integrates a user-friendly interface, a robust backend processing unit, a secure execution environment, and an AI-based analysis component. The architecture

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 12, April 2026
Impact Factor: 8.2

ensures efficient communication between components while maintaining reliability, security, and extensibility. The overall system is structured into four primary layers: the Presentation Layer, Application Layer, Execution & AI Analysis Layer, and Data Layer.

1. Architectural Overview

The system follows a client-server model where the frontend interacts with backend services through RESTful APIs. The backend acts as the central controller, managing user requests, executing code, communicating with the AI module, and handling database operations. The architecture is designed to support concurrent users and scalable deployment in academic environments.

2. System Architecture Diagram

Below is a conceptual diagram representing the system architecture:

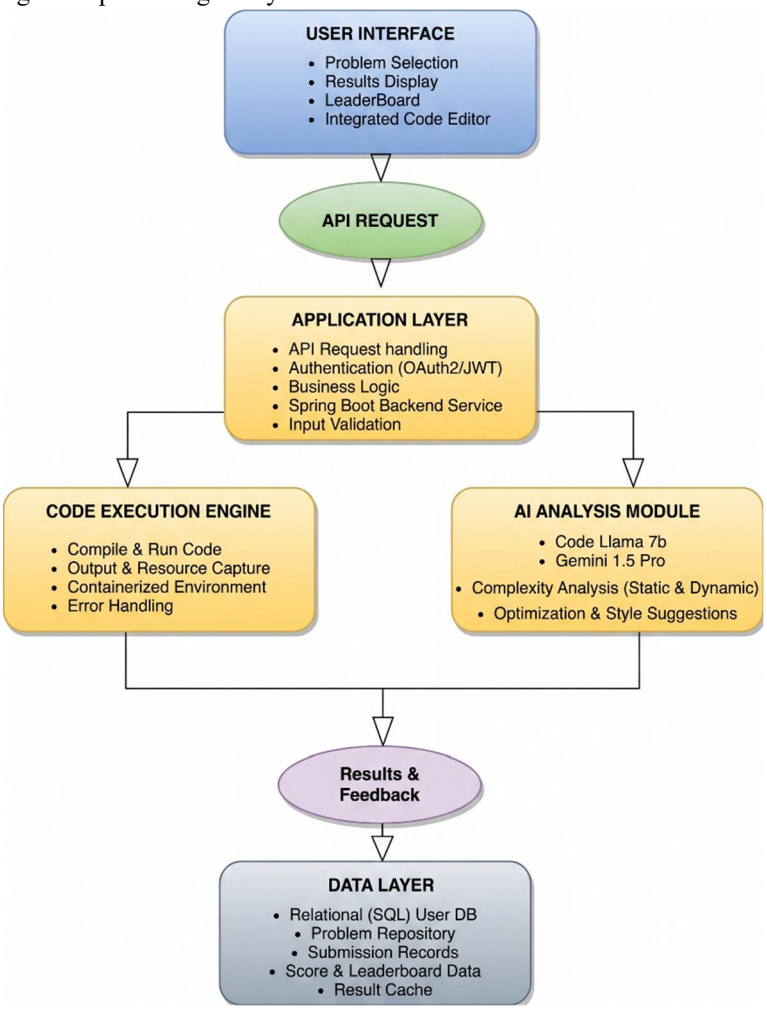

Fig. 2. System Architecture

3. Component Description

3.1 Presentation Layer

The presentation layer is responsible for user interaction and provides an intuitive interface for both students and administrators. It is developed using modern frontend technologies, enabling dynamic rendering and responsiveness. This

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 12, April 2026
Impact Factor: 8.2

layer includes features such as a code editor, problem statement display, submission interface, and result visualization. It also presents performance metrics and leaderboard rankings to users in a structured format.

3.2 Application Layer

The application layer forms the core of the system and is implemented using a backend framework that supports RESTful communication. It handles incoming requests from the frontend, processes user inputs, and coordinates operations between different modules. Key functionalities include user authentication, problem management, submission handling, and integration with external services. This layer ensures proper validation of inputs and maintains the logical flow of operations within the system.

3.3 Code Execution Engine

The code execution engine is a critical component responsible for compiling and running user-submitted programs. It processes the code in a controlled environment, ensuring that execution is performed securely and efficiently. The engine captures compilation errors, runtime exceptions, and program output. To maintain system stability, execution constraints such as time limits and resource restrictions are enforced. This module ensures that users receive accurate feedback regarding the correctness of their code.

3.4 AI Analysis Module

The AI analysis module uses a hybrid approach to provide intelligent insights into the submitted code. A fine-tuned code model first analyzes the program structure to estimate time and space complexity and detect inefficiencies. The results are then passed to a generative AI component via an API, which produces clear explanations and optimization suggestions.

This combination ensures accurate analysis along with easy-to-understand, context-aware feedback, enhancing the overall learning experience.

3.5 Data Layer

The data layer is responsible for persistent storage and management of all system-related information. It includes user profiles, problem statements, submission records, and performance data. The database is designed to support efficient querying and scalability. By maintaining a history of submissions, the system enables progress tracking and analytical insights into student performance over time.

4. Data Flow and Interaction

The system operates through a well-defined data flow. When a user submits code, the request is sent to the backend, which forwards it to the execution engine. After execution, the results are passed to the AI module for further analysis. The combined output is then stored in the database and returned to the frontend for display. This seamless interaction ensures minimal latency and a smooth user experience.

V. FUTURE SCOPE

The proposed system establishes a strong foundation for intelligent code evaluation and learning; however, several enhancements can further extend its capabilities. One key direction is the expansion to support multiple programming languages beyond Java, making the platform applicable to a broader range of users. This would involve extending the execution engine and adapting the analysis pipeline to handle diverse language constructs.

Another significant area for future improvement lies in advancing the hybrid AI module. While the current system combines a fine-tuned code model with a generative AI component, further refinement can focus on improving model accuracy through better fine-tuning, dataset expansion, and integration with lightweight static analysis techniques. This would enhance the precision of complexity estimation and provide more reliable optimization suggestions.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 12, April 2026
Impact Factor: 8.2

The platform can also evolve by incorporating adaptive learning features. By analyzing user performance and submission history, the system can recommend personalized problem sets and learning paths tailored to individual skill levels. This would create a more targeted and effective learning experience.

Additional features such as plagiarism detection, collaborative coding environments, and real-time guidance mechanisms can further enrich the platform. Enhanced analytics dashboards for instructors can also be introduced to monitor student progress and identify learning gaps more effectively.

From a technical perspective, future work may include deploying the system on scalable cloud infrastructure and strengthening secure sandboxing for code execution. These improvements would ensure better performance, security, and readiness for large-scale usage.

VI. CONCLUSION

This paper presented an AI-powered code learning and evaluation platform aimed at improving programming proficiency through continuous practice and intelligent feedback. The system addresses common limitations of traditional learning methods by enabling flexible coding practice and automated assessment beyond classroom constraints.

A key contribution of the proposed platform is the integration of a hybrid AI-based analysis module. By combining a fine-tuned code-centric model for structural and complexity analysis with a generative AI component for explanation generation, the system provides both accurate evaluation and clear, human-readable insights. This approach allows users to understand not only the correctness of their code but also its efficiency, design quality, and possible optimizations.

Furthermore, features such as submission tracking, performance monitoring, and a leaderboard mechanism encourage consistent engagement and progressive learning. The modular system architecture supports scalability and adaptability for academic use.

In summary, the proposed platform effectively bridges the gap between theoretical understanding and practical implementation by transforming code evaluation into a guided learning experience, thereby supporting the development of efficient and industry-ready programmers.

REFERENCES

[1]. Kumar and S. Singh, "Automated Code Evaluation Systems in Programming Education: A Review," *International Journal of Computer Applications*, vol. 182, no. 45, pp. 12–18, 2020.

[2]. Y. Liu, A. Jain, and K. Srinivasan, "Hybrid AI Approaches for Code Analysis and Optimization," *IEEE Access*, vol. 11, pp. 45000–45015, 2023.

[3]. P. Brown et al., "Language Models are Capable of Code Understanding and Generation," *Proceedings of the ACM Conference on AI and Software Engineering*, pp. 1–10, 2021.

[4]. M. Chen et al., "Evaluating Large Language Models Trained on Code," *arXiv preprint arXiv:2107.03374*, 2021.

[5]. K. Patel and D. Shah, "Gamification in Education: Enhancing Student Engagement through Leaderboards and Reward Systems," *International Journal of Educational Technology*, vol. 8, no. 2, pp. 55–63, 2022.

[6]. S. Verma and A. Mehta, "Online Coding Platforms for Skill Development: A Comparative Study," *Journal of Educational Computing Research*, vol. 58, no. 4, pp. 789–805, 2020