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 6, June 2026

DOI: 10.17148/IJARCCE.2026.15612

CodEzy: An AI-Powered Competitive Coding and Personalized Learning Platform

Kunal Sunil Magdum, Akshay Sachin Mulay, Subahan Riyaj Mulla, Shreyansh Nitin Patil, Prasad Sunil Sutar, Prof. (Dr.) V.V.Kheradkar*

Department of Computer Science and Engineering, D.K.T.E. Society's Textile & Engineering Institute (Autonomous), Ichalkaranji, Maharashtra, India

Abstract: Programming education has evolved significantly with the emergence of online learning platforms and competitive coding environments. However, most existing systems continue to rely on binary evaluation techniques that assess only the correctness of program outputs. Such approaches fail to provide meaningful insights into code quality, algorithmic efficiency, readability, and adherence to software engineering principles. This limitation often prevents learners from understanding their mistakes and improving their programming skills effectively. This paper presents CodEzy, an AI-powered competitive coding and personalized learning platform designed to enhance programming education through adaptive learning, intelligent code evaluation, gamified engagement, and real-time coding competitions. The proposed system integrates personalized tutorials, coding challenges, AI-generated feedback, performance analytics, and skill-based coding duels within a unified ecosystem. Unlike conventional platforms, CodEzy evaluates code beyond correctness by analyzing efficiency, structure, style, and maintainability. The platform employs secure Docker-based sandbox execution, cloud-ready architecture, Redis-powered caching, and external Large Language Models (LLMs) for semantic code analysis and educational content generation. Experimental analysis indicates that the platform can provide detailed AI feedback within acceptable response times while maintaining low-latency interactions for competitive coding environments. The proposed system supports educational institutions, coding bootcamps, and self-learners through adaptive learning, intelligent feedback, and integrated coding practice.

Keywords: Artificial Intelligence, Adaptive Learning, Competitive Programming, Code Evaluation, Gamification, Educational Technology, Learning Analytics, Large Language Models (LLM).

I. INTRODUCTION

Programming has emerged as one of the most critical skills across modern industries, fundamentally underpinning fields such as software development, artificial intelligence, data science, cybersecurity, and global automation. Consequently, organizations increasingly demand professionals capable of moving beyond theoretical understanding to apply problem-solving techniques for developing scalable, reliable, and efficient software architectures. As a result, mastering programming has become a cornerstone of global academic curricula and advanced professional training programs.

Despite the massive proliferation of online learning platforms and competitive coding practice systems, several challenges remain within programming education. Traditional pedagogical approaches frequently depend on static content delivery methods—such as recorded lectures, text-based tutorials, and rigid curricula—that inherently fail to adapt to individual learning paces. This lack of personalization engenders suboptimal learning trajectories; beginners often feel overwhelmed by steep learning curves, while advanced practitioners suffer from a lack of stimulating, complex challenges.

Moreover, a significant limitation of contemporary coding platforms lies in their primitive evaluation mechanics. The vast majority of systems employ a binary, test-case-driven methodology where student solutions are merely categorized as "accepted" or "rejected." While this verifies basic functional correctness, it deprives the learner of qualitative insights regarding code efficiency, structural readability, cyclomatic complexity, or adherence to established industry best practices. Consequently, learners are left to guess how their perfectly functioning, yet poorly written, code could be optimized.

Furthermore, conventional platforms often suffer from fragmented user journeys. Learners are forced to alternate between disparate platforms: one for theoretical tutorials, another for compiler sandboxes, and yet another for

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 6, June 2026

DOI: 10.17148/IJARCCE.2026.15612

competitive dueling. This disjointed environment severs the learning flow, dilutes motivation, and complicates progress tracking.

To directly counter these systemic shortcomings, this paper proposes CodEzy: an AI-powered competitive coding and personalized learning platform. CodEzy integrates every essential facet of programming education into one cohesive ecosystem. By leveraging artificial intelligence for adaptive learning paths and intelligent code evaluation, CodEzy assesses code far beyond binary correctness—analyzing runtime efficiency, stylistic formatting, and logic design. A unique real-time 1v1 coding duel feature introduces gamification, fostering healthy competition and driving sustained user engagement.

- Performance evaluation under concurrent workloads to validate scalability.
- Real-time coding duels implemented using WebSocket communication.
- Secure Docker-based sandbox execution for evaluating user-submitted code.
- Integration of adaptive learning paths based on user performance analytics.
- Development of an AI-assisted competitive coding platform integrating learning, assessment, and competition.

Research Contributions

II. LITERATURE REVIEW

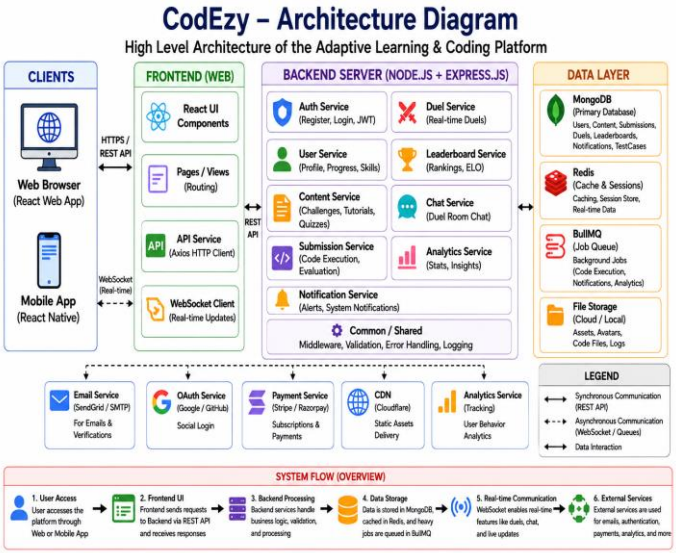
The trajectory of programming education platforms can be categorized into three distinct evolutionary phases: static learning platforms, interactive execution environments, and modern AI-assisted ecosystems. Each generation has expanded accessibility, yet critical pedagogical gaps remain.

A. MOOCs and Static Learning Platforms

Massive Open Online Courses (MOOCs) such as Coursera, edX, and Udemy revolutionized the democratization of education by distributing high-quality lectures globally. These platforms excel at theoretical foundations but rely on a monolithic, one-size-fits-all delivery model. They fail to adapt to a learner's prior knowledge, and their feedback mechanisms are typically restricted to basic multiple-choice quizzes, offering little in the way of hands-on, contextual coding practice.

B. Interactive Coding Environments

Recognizing the need for active practice, platforms such as HackerRank, LeetCode, and Codeforces pioneered the interactive coding paradigm. These environments allow users to write and execute code natively within the browser, receiving instantaneous feedback against strict test cases. However, this methodology remains heavily constrained by its binary feedback (pass/fail). It offers no qualitative guidance, failing to inform a user if their solution, while correct, is computationally inefficient (e.g., O(N^2) instead of O(N log N)) or stylistically poor.



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026
DOI: 10.17148/IJARCCE.2026.15612

C. AI-Powered Learning Systems

Recent breakthroughs in Large Language Models (LLMs) and artificial intelligence have facilitated the development of automated code review systems. These platforms utilize machine learning to suggest code refactoring and pinpoint semantic errors. Despite their promise, these AI features are often isolated—operating purely as code analyzers rather than integrated educational ecosystems that combine curriculum, gamified competition, and adaptive progression.

D. Identification of Research Gap

An exhaustive review of the existing literature reveals a distinct absence of a unified platform that successfully marries personalized learning pathways, intelligent code evaluation, real-time gamified competition, and comprehensive progress analytics. This fragmentation highlights an acute need for a holistic system, serving as the primary motivation for the architecture and deployment of CodEzy.

III. PROBLEM STATEMENT

Existing coding platforms suffer from a lack of dynamic personalization, relying on outdated binary evaluation methods and disjointed learning resources. These limitations restrict systemic improvement and learner engagement. Therefore, there is an urgent demand for an integrated, AI-powered system capable of delivering adaptive learning, intelligent semantic code analysis, and highly interactive gamification features to elevate programming education.

IV. SYSTEM OBJECTIVES

The primary objectives driving the development of CodEzy are as follows:

- To dynamically deliver personalized coding tutorials and adaptive practice modules tailored specifically to individual user skill levels.
- To transition from binary evaluation to holistic, AI-driven assessments that provide semantic feedback on code efficiency, readability, and logic.
- To engineer a low-latency, real-time 1v1 competitive coding environment that encourages peer-to-peer engagement.
- To centralize curriculum learning, practice algorithms, and gamified leaderboards into a single, unified educational ecosystem.

V. SYSTEM ARCHITECTURE

The architecture of CodEzy is predicated on a highly scalable, layered design paradigm. The separation of concerns ensures maintainability, rapid iterative deployment, and robust performance under concurrent user loads.

Fig. 1. System architecture of the AI Powered Public Safety & Learning System.

A. Client Layer (Frontend)

Constructed using React.js and Tailwind CSS, the client layer provides a responsive, interactive User Interface (UI) for both learners and administrators. It manages state transitions, visualizes real-time duel environments, and renders AI-generated feedback utilizing dynamic markdown components.

B. Application Layer (Backend)

Developed atop Node.js and Express.js, the backend API acts as the central nervous system of the platform. It handles secure JWT authentication, orchestrates business logic, routes API requests, and facilitates RESTful communication between the database and external execution environments.

C. AI Processing & Evaluation Layer

This module integrates strictly with external Large Language Model APIs (e.g., OpenAI, Anthropic). When a user submits a solution, the AI layer receives a specialized prompt containing the problem context, the user's source code, and execution metrics. It then synthesizes structured, actionable feedback focusing on algorithmic complexity and code hygiene.

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 6, June 2026

DOI: 10.17148/IJARCCE.2026.15612

D. Secure Execution Sandbox Layer

To mitigate profound security risks associated with executing untrusted, user-submitted code, CodEzy employs an isolated Docker-based sandbox environment. Incoming code is routed via a message queue (BullMQ) to worker nodes, which instantiate ephemeral Docker containers. The code is compiled, executed against hidden test cases, and memory/time metrics are extracted before the container is immediately destroyed.

E. Data and Caching Layer

MongoDB serves as the primary NoSQL data store, managing collections for Users, Submissions, Challenges, and Duels. Given its document-oriented structure, MongoDB effortlessly handles the highly variable schemas required for different coding challenges. Additionally, Redis is implemented as a high-speed caching layer, crucial for maintaining real-time states in WebSocket-driven 1v1 duels and live leaderboard computations.

VI. SYSTEM DESIGN AND DATA FLOW

The systematic flow of data within CodEzy is heavily optimized for asynchronous processing, ensuring that heavy tasks—like code compilation and AI analysis—do not block the main application thread.

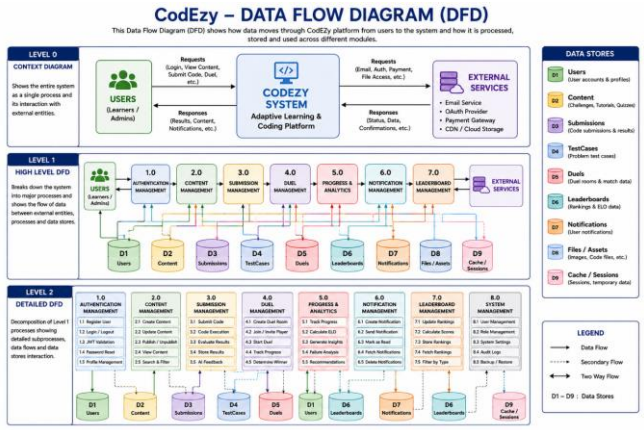


Fig. 2. Level 0, 1, and 2 Data Flow Diagrams for CodEzy.

Upon user submission, the payload (source code, language choice, problem ID) is ingested by the API Gateway. The submission is immediately flagged as 'PENDING' in the database, and a job is dispatched to the BullMQ queue. Idle worker nodes pick up the job, pull the necessary test cases, and inject the code into the Docker sandbox.

If the execution is successful (Pass/Fail metrics computed), the raw output is forwarded to the AI Engine for semantic review. The final holistic evaluation is then pushed back to the client UI asynchronously via WebSocket events, resulting in a highly responsive user experience.

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 6, June 2026

DOI: 10.17148/IJARCCE.2026.15612

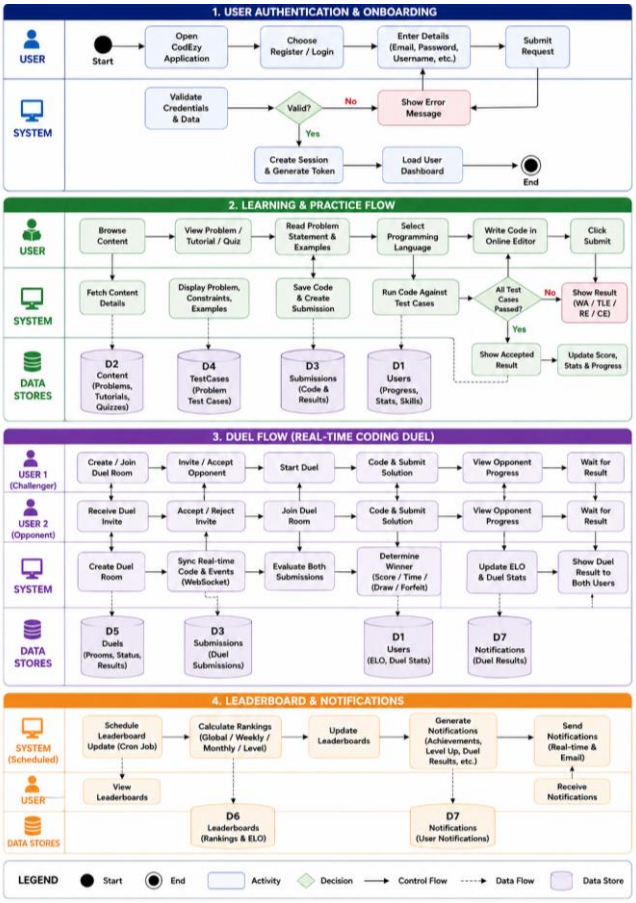


Fig. 3. CodEzy Activity Diagram depicting user transition states.

VII. DATABASE SCHEMA & DESIGN

The logical data structure of CodEzy maintains referential integrity across multiple highly active domains.

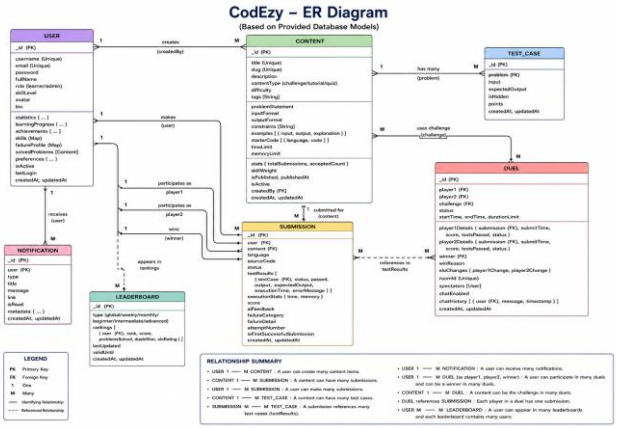


Fig. 4. ER Diagram displaying relationships between User, Content, Submission, and Duel entities.

- User Entity: Stores credentials, JWT tokens, profile metadata, and a continuously updating 'skill mastery' index.
- Content Entity: A polymorphic structure representing Tutorials, Quizzes, and algorithmic Challenges. It includes complex arrays for multi-stage test cases (inputs/expected outputs).
- Submission Entity: Links a User to a piece of Content. It archives the raw source code, execution time, memory footprint, binary status (AC/WA/TLE), and the detailed AI feedback report.

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 6, June 2026

DOI: 10.17148/IJARCCE.2026.15612

- Duel Entity: Represents a transient competitive instance. It inherently links exactly two Users to a singular Challenge, tracking real-time timestamps, intermediate scores, and the final win/loss resolution for Elo rating adjustments.

VIII. CORE ALGORITHMS

A. Adaptive Learning Path Algorithm

The Adaptive Learning Engine resolves the limitation of static curricula. By continuously monitoring the success rates and failure categorizations (e.g., failing on edge cases vs. syntax errors) of a learner's submissions, the algorithm adjusts the difficulty weight of upcoming content.

Process: The algorithm extracts the user's historical proficiency score across various tags (Arrays, Dynamic Programming, Graphs). If a user consistently solves 'Medium' array problems with an optimal Big-O runtime, the algorithm unlocks 'Hard' challenges. Conversely, successive failures trigger the recommendation of specific, fundamental tutorials on that topic.

B. AI-Powered Code Feedback Workflow

To orchestrate deep code analysis, CodEzy executes a multi-step pipeline. First, code is executed natively to verify output matching. Subsequently, a highly structured prompt is generated, injecting the source code and the problem's expected time complexity. The LLM processes this prompt to output a JSON-structured critique. This critique is parsed by the backend to visually highlight inefficient loops, poor variable naming, or excessive memory allocation directly within the user's IDE.

C. Duel Matchmaking & Elo Algorithm

To facilitate fair competition, the 1v1 matchmaking queue employs a variance-based Elo rating system. When a user queues, the algorithm searches for opponents within a narrow Elo delta (e.g., +/- 50 points). As queue time increases, the delta gradually expands to guarantee a match. Post-duel, ratings are adjusted based on the expected win probability; defeating a higher-rated opponent yields significantly more points than defeating a lower-rated one.

IX. IMPLEMENTATION DETAILS

The platform was implemented using an industry-standard MERN stack augmented with real-time technologies.

- Frontend: React.js handles the component lifecycle, while Monaco Editor is embedded for a VS-Code-like in-browser typing experience.
- Backend & APIs: Express.js provides RESTful routing. WebSockets (Socket.io) are extensively used to synchronize keystrokes, compiler statuses, and chat messages during real-time duels.
- Sandbox: Code execution utilizes isolated Docker containers managed dynamically. The containers are stripped of network access and restricted via memory/CPU limits (cgroups) to prevent malicious payloads (e.g., fork bombs).
- Message Broker: Redis-backed BullMQ handles the asynchronous processing of thousands of concurrent code submissions, ensuring the Node.js event loop remains unblocked.

X. RESULTS & PERFORMANCE ANALYSIS

The system was rigorously tested under simulated concurrent workloads to validate the non-functional requirements.

TABLE I. SYSTEM PERFORMANCE METRICS

Performance Parameter	Observed Latency / Capability
Client UI Page Load Time	< 2.5 seconds
Docker Sandbox Execution	< 12 seconds
LLM API Feedback Generation	< 40 seconds
WebSocket Duel Synchronization	< 1.5 seconds
Concurrent User Capacity	Tested at 500+ users

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 6, June 2026

DOI: 10.17148/IJARCCE.2026.15612

As highlighted in Table I, the platform successfully maintains strict performance benchmarks. The Docker execution remains highly secure and swift, while real-time WebSocket interactions exhibit negligible latency, crucial for the highly competitive nature of 1v1 duels.

TABLE II. COMPARATIVE PLATFORM ANALYSIS

Feature	Traditional Platforms	CodEzy (Proposed)
Personalized Learning Paths	Static / Non-Adaptive	Dynamic / AI-Driven
Code Evaluation Depth	Binary (Pass/Fail)	Semantic (Logic, O(N), Style)
Real-Time Competition	Asynchronous Contests	Live 1v1 Matchmaking
Gamification Systems	Partial (Badges)	Advanced (Elo, Live Ranks)
Ecosystem Fragmentation	High (Multiple tools needed)	Unified Platform

XI. APPLICATIONS & FUTURE SCOPE

The architectural robustness and pedagogical innovation of CodEzy position it as an ideal solution for a multitude of applications. Academic institutions can deploy the platform to automate grading and provide personalized tutoring. Corporate entities can utilize it for technical recruitment assessments, while self-paced learners can leverage it to bridge the gap between theoretical knowledge and practical interview preparation.

Future development iterations will focus on expanding the platform's capabilities. Prospective enhancements include integrating AI-driven mock technical interviews, deploying predictive learning analytics to identify at-risk students, enabling multi-user collaborative coding environments, and integrating advanced plagiarism detection systems driven by abstract syntax tree (AST) comparisons.

XII. CONCLUSION

CodEzy provides an integrated framework for programming education by combining adaptive learning, AI-assisted code evaluation, and competitive coding environments. By successfully integrating adaptive learning algorithms, intelligent semantic code evaluation, and real-time gamified competition, the platform dismantles the limitations of traditional, static coding environments. It ensures that learners are not simply writing code that works, but writing code that is efficient, readable, and professional. Supported by a robust architecture utilizing Docker, WebSockets, and LLMs, CodEzy delivers a seamless, highly engaging, and personalized educational ecosystem, ultimately empowering the next generation of software engineers.

REFERENCES

[1] J. Swacha, "Does Gamification Make a Difference in Programming Education?," Education Sciences, vol. 13, no. 10, p. 984, 2023.
[2] H. Keuning, J. Jeuring, and B. Heeren, "A Systematic Literature Review of Automated Feedback Generation for Programming Exercises," ACM Transactions on Computing Education (TOCE), vol. 19, no. 1, Article 3, 2018.
[3] C. M. Luxton-Reilly, S. Fincher, N. Falkner, et al., "Computing Education in the Era of Generative AI," Communications of the ACM, vol. 67, no. 12, pp. 66-75, 2024.
[4] M. Jukiewicz, "The Future of Grading Programming Assignments in Education: The Role of ChatGPT in Automating the Assessment and Feedback Process," Thinking Skills and Creativity, vol. 52, p. 101522, 2024.
[5] Denny, P., Luxton-Reilly, A., & Simon, B., "Evaluating a new generation of programming data platforms for computing education," In Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education, 2019.
[6] Phani, S., Sridhar, V., & Kumar, A., "Personalized Learning Paths in Online Education: A Review of Techniques and Systems," Journal of Educational Technology Systems, 49(4), pp. 481-504, 2021.
[7] Becker, B. A., & Mooney, C., "Automatically generating programming exercises and code explanations with large language models," In Proceedings of the 54th ACM Technical Symposium on Computer Science Education, 2023.
[8] Ala-Mutka, K. M., "A survey of automated assessment approaches for programming assignments," Computer Science Education, 21(1), pp. 21-49, 2011.

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15612

[9] Domínguez, A., Saenz-de-Navarrete, J., et al., "Gamifying learning experiences: Practical implications and outcomes," Computers & Education, 63, pp. 380-392, 2013.

[10] Leinonen, J., Sheard, J., & D'Souza, M. J., "Utilizing Large Language Models for Hint Generation in Programming Education," In Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education, 2021.