Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 04, ICTEM 2.0 (2026)

Development of a Centralized Competitive Coding Platform for College Students

1st Ruchika Gupta
Dept. of Computer Science
Moradabad Inst. of Tech.
Moradabad, India
ruchikagupta.mit@gmail.com

2nd Devesh Gupta
Dept. of Computer Science
Moradabad Inst. of Tech.
Moradabad, India
deveshgupta916@gmail.com

3rd Armaan Siddiqui
Dept. of Computer Science
Moradabad Inst. of Tech.
Moradabad, India
armaansiddiqui.pms@gmail.com

4th Akshdeep Singh
Dept. of Computer Science
Moradabad Inst. of Tech.
Moradabad, India
akshsingh977@gmail.com

5th Somya Kaushik
Dept. of Computer Science
Moradabad Inst. of Tech.
Moradabad, India
somyakaushik0911@gmail.com

***Abstract*—All industries were increasingly requiring a basic level of programming skills from nearly every employee. The wide availability of learning materials was not enough to ensure success in achieving a suitable level of skills. As a result, a significant gap was noted between most college graduates and what employers expected regarding programming. Consequently, there was a clear need for a dedicated platform to build a strong coding community through structured and ongoing education.**

In this project, a specialized web application was created to support college students and help them improve their skills in Competitive Programming and Data Structures Algorithms (DSA). The web application served as a single online resource where faculty members could assign coding challenges for students to complete. The website also included features that allowed students to compare their performance with their peers. To boost motivation and engagement, challenges, leaderboards, and gamification elements were added to coding activities.

The web application used a modern technology stack. The front end was built with React and Tailwind CSS, while the back end was implemented using Express.js. Data was stored and managed with MongoDB. The primary functionality of the app allowed users' submitted code, along with the necessary driver code, to be sent to the Judge0 API. The code was then tested against predefined unit test cases, and a status ID was returned to help with error tracking and provide immediate feedback to users.

Additionally, a chat room, a discussion forum, and gamification features were included to create interactive learning experiences for college students involved in coding. This platform effectively addressed the urgent need for students to gain coding skills relevant to the industry, supporting their preparation for careers in technology.

***Keywords*— competitive programming, data structures, algorithms, gamification, Judge0, React, MongoDB, web application**

I. Introduction

Learning how to program often feels intimidating at first, but it's important to remember that it's never too late to learn. Programming is mainly about thinking through problems. Each problem needs to be broken down into smaller parts so that a sequence of steps, called algorithms, can be created to find a solution. It's also important that the code written for the solution is both correct and efficient.

Many people don't realize that becoming a skilled programmer takes a lot of time, effort, and practice. Mastery of programming doesn't happen overnight. Unfortunately, many individuals stop learning to code because they aren't aware of the commitment required.

As more work and business activities shift to digital platforms, it's crucial for everyone to have at least a basic understanding of coding. Young professionals can boost their competitiveness by acquiring coding skills. This knowledge gives them better control and understanding of the technology they use. Being able to innovate and stay competitive in one's field relies heavily on coding skills, especially with the ongoing demand for software development. Additionally, those who can write code are seen as valuable assets, creating more opportunities for career growth.

With regular practice in writing code, along with focus and deliberate effort, both the speed and quality of code production can greatly improve.

II. Literature Review

In this literature review, an extensive study of 10–12 research papers related to competitive programming platforms, online coding assessment systems, and gamified learning environments was conducted. The findings of these papers highlighted the growing importance of coding platforms in skill development, interview preparation, and talent identification. Various studies emphasized the role of gamification, structured problem sets, community interaction, and contest-based evaluation in improving logical thinking and problem-solving abilities among learners.

A. Existing Systems

The following observations were made based on the review of the existing systems.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 04, ICTEM 2.0 (2026)

1) HackerRank: HackerRank was identified as a well-known platform where coding challenges were solved by programmers from around the world. A wide range of computer science topics and programming languages such as Java, C++, and PHP were supported. When code was submitted by a programmer, it was primarily evaluated based on the correctness of the output. Rankings on the global leaderboard were determined according to these scores. In addition to individual practice, short competitive events called ”Code Sprints” were organized, in which participants competed to solve a defined set of problems. Competitive coding was made more engaging through gamification techniques. Partnerships with companies were also established to facilitate talent recruitment based on demonstrated coding skills. The practice section of the platform was made freely accessible to users.

2) CodeChef: CodeChef was recognized as a global platform for competitive programming, supported by a large and active community. More than 50 programming languages were supported. A space was created for students and professionals to practice, compete, and continuously improve their coding skills. Practice contests were conducted regularly, often in preparation for major competitions such as the ACM-ICPC. Monthly contests were organized, and prizes were offered to increase motivation. Special emphasis was placed on encouraging younger students in India to develop a strong programming foundation from an early stage.

3) LeetCode: LeetCode was identified as a leading platform primarily focused on mastering Data Structures and Algorithms (DSA), which were considered essential for technical interviews at top technology companies. Thousands of problems were provided and were organized according to difficulty levels and categories. Unlike platforms that focused heavily on competitive contests, greater emphasis was placed on comprehensive problem libraries and community-driven discussions. Through the ”Discuss” section, multiple solutions, explanations, and complexity analyses were shared by users. This approach was found to support deeper conceptual understanding. The platform was widely used by students and professionals preparing for technical interviews and was considered a standard reference for skill assessment in the industry.

TABLE I
COMPARISON OF EXISTING CODING PLATFORMS

Feature	HackerRank	CodeChef	LeetCode
Primary Focus	General coding challenges, gamification, and corporate recruiting	Competitive programming culture and contests	Mastery of Data Structures & Algorithms (DSA) for interviews
Problem Type	Wide variety of computer science topics	Primarily competitive programming style problems	Core DSA and algorithmic concepts
Learning Style	Solving and ranking through short-term contests	Regular contests and community-based practice	Structured problem solving with detailed discussions
Gamification Ranking	Global leaderboards, badges, and scoring system	Monthly contests with prizes	Focus on individual progress and problem-solving count
Key Differentiator	Strong corporate recruiting integration	Large community and grassroots programming focus	Extensive interview-oriented problem library and solution analysis
Accessibility	Free for core practice	Free	Free

III. METHODOLOGY

The methodology adopted for the development of the centralized competitive coding platform followed a systematic and modular approach. The entire development process was divided into requirement analysis, system design, implementation, integration, and testing phases to ensure reliability and scalability.

A. Requirement Analysis

In the initial phase, the functional and non-functional requirements of the system were identified. Functional requirements included user registration and authentication, problem management, code submission, automated evaluation, leaderboard generation, and faculty monitoring features. Non-functional requirements included system scalability, secure data handling, low response time, and real-time feedback capability.

A detailed study of existing coding platforms was conducted to identify their strengths and limitations. Based on this analysis, requirements specific to an institution-oriented coding environment were defined, emphasizing faculty involvement, structured assignments, and centralized monitoring.

B. System Design Approach

A modular and service-oriented architecture was adopted to ensure separation of concerns. The system was divided into independent modules including:

- User Authentication Module
- Problem Management Module
- Code Execution and Evaluation Module
- Gamification and Leaderboard Module
- User Activity Tracking Module

The frontend and backend components were designed as decoupled systems communicating via RESTful APIs. This architectural decision improved maintainability and allowed independent scaling of services.

C. Technology Stack Selection

The selection of technologies was based on performance, scalability, and ease of development. The frontend was implemented using React along with Tailwind CSS to create

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 04, ICTEM 2.0 (2026)

a responsive and interactive user interface. The backend was developed using Node.js and Express.js to handle API requests and business logic efficiently. MongoDB was selected as the database due to its flexibility in handling dynamic data structures such as user submissions and activity logs.

For secure and isolated code execution, the Judge0 API was integrated. This API allowed submitted programs to be compiled and executed in a sandboxed environment, thereby preventing malicious execution and ensuring reliable evaluation.

D. Code Execution Workflow

The methodology for code evaluation followed these steps:

1) The student selected a problem and wrote the solution in the online editor.
2) Upon submission, the student's code was combined with predefined driver code.
3) The combined code was sent to the Judge0 API through the backend server.
4) The API compiled and executed the code against predefined visible and hidden test cases.
5) Execution results including status ID, compilation errors, runtime errors, and output were returned.
6) The system compared the output with expected results and generated feedback.
7) Scores and leaderboard rankings were updated accordingly.

This structured workflow ensured fairness, accuracy, and real-time response in evaluation.

E. Gamification Strategy

To maintain student engagement, gamification principles were incorporated into the methodology. Leaderboards were dynamically updated based on scores and streaks. Badges and achievement milestones were awarded based on consistent participation and performance. This approach was implemented to promote continuous learning behavior rather than short-term participation.

F. Testing and Validation

The system was tested using unit testing and integration testing approaches. Each module was tested independently before integration. Multiple edge cases were used to verify code execution reliability. Stress testing was conducted to evaluate system performance under simultaneous submissions.

User testing was also performed within a controlled academic environment to ensure usability and effectiveness. Feedback from students and faculty members was collected and incorporated to refine system features.

The adopted methodology ensured that the final system was scalable, secure, interactive, and aligned with institutional requirements.

The overall workflow is illustrated in Fig. 1 and Fig. 2.

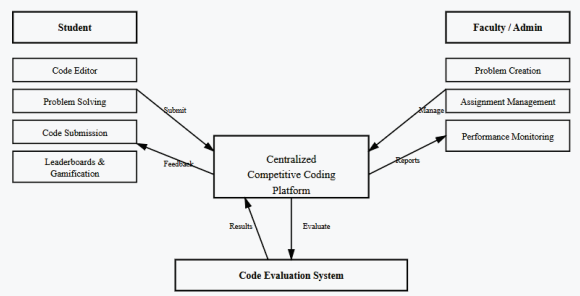

Fig. 1. Conceptual overview of the centralized competitive coding platform

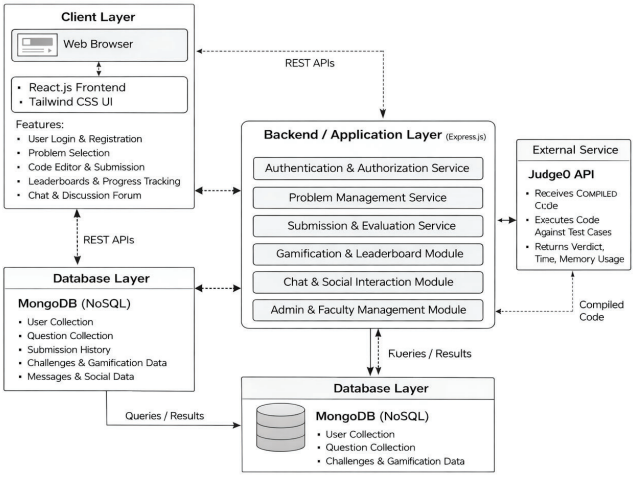

Fig. 2. System Architecture

IV. Implementation and Result Analysis

The implementation of the proposed centralized competitive coding platform was carried out using a full-stack web development approach. The system architecture was divided into frontend, backend, database, and external API integration layers to ensure modularity, scalability, and maintainability. After development, the system was evaluated to analyze its functional performance, user engagement, and overall effectiveness.

A. System Implementation

The frontend of the platform was developed using React.js along with Tailwind CSS to provide a responsive and interactive user interface. The frontend handled user authentication interfaces, problem browsing, online code editing, submission tracking, and leaderboard visualization. RESTful APIs were used to enable communication between the frontend and backend components.

The backend was implemented using Node.js with the Express.js framework. It managed business logic, authentication, role-based access control, problem management, submission processing, and leaderboard updates. Secure password hashing and JWT-based authentication were implemented to ensure secure access control for students, faculty, and administrators.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 04, ICTEM 2.0 (2026)

MongoDB was used as the database for storing user credentials, coding problems, test cases, submission history, and leaderboard records. The NoSQL structure allowed flexible handling of submission logs and activity tracking data. Indexing was applied on frequently queried fields such as user ID and problem ID to improve performance.

For secure code execution, the Judge0 API was integrated into the backend. When a student submitted a solution, the code was combined with predefined driver code and sent to the Judge0 API for compilation and execution in a sandboxed environment. The API returned execution results, including compilation status, runtime errors, and output. The system then compared the output against predefined expected results and generated an appropriate verdict such as Accepted, Wrong Answer, or Compilation Error. Scores and rankings were updated dynamically.

Gamification elements such as leaderboards, streak tracking, and achievement badges were implemented within backend logic and reflected in the frontend dashboard. These features were designed to encourage consistent participation and healthy competition among students.

B. Result Analysis

After implementation, the system was tested in a controlled academic environment to evaluate its functionality and effectiveness.

From a functional perspective, the platform successfully executed and evaluated code across multiple programming languages including C, C++, Java, and Python. The integration with Judge0 ensured reliable compilation and execution with accurate feedback. The average response time for submission evaluation remained within acceptable limits under normal usage conditions.

Stress testing was conducted by simulating multiple simultaneous submissions. The modular backend design ensured that external API delays or failures were handled gracefully without affecting overall system stability.

User engagement was analyzed through observation of submission frequency and leaderboard participation. The introduction of gamification features such as streaks and ranking systems resulted in increased consistency in coding practice among students. Leaderboards encouraged peer comparison, while badges provided milestone-based motivation.

Faculty members reported improved monitoring capability due to centralized tracking of student performance and submission history. The structured assignment feature ensured alignment between academic objectives and coding practice.

Overall, the results demonstrated that the proposed platform effectively addressed the identified skill gap by providing structured learning, real-time evaluation, competitive engagement, and institutional oversight within a single integrated system.

V. Conclusion and Future Scope

This study presented the design and development of a centralized competitive coding platform aimed at improving

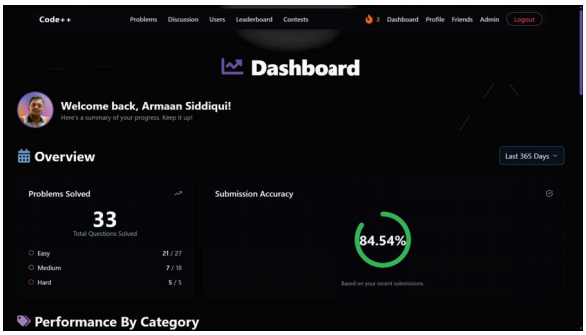

Fig. 3. Dashboard interface displaying user progress, problem statistics, and coding performance overview

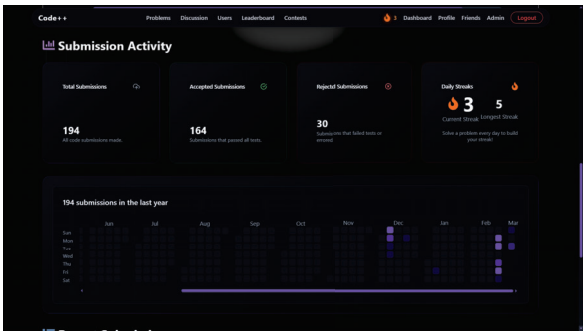

Fig. 4. Submission activity dashboard showing total submissions, accepted solutions, and coding streak statistics

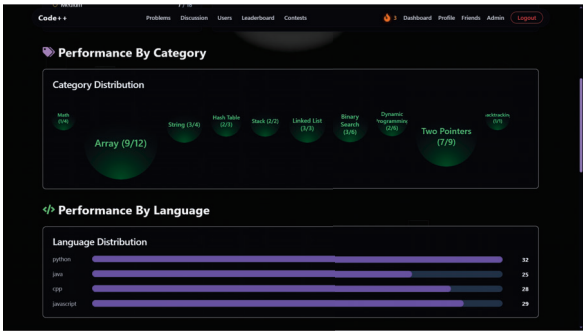

Fig. 5. Performance analysis showing problem-solving distribution across categories and programming languages

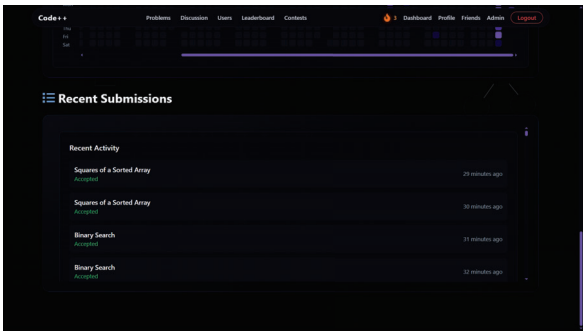

Fig. 6. Recent submissions panel displaying latest solved problems and user activity

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 04, ICTEM 2.0 (2026)

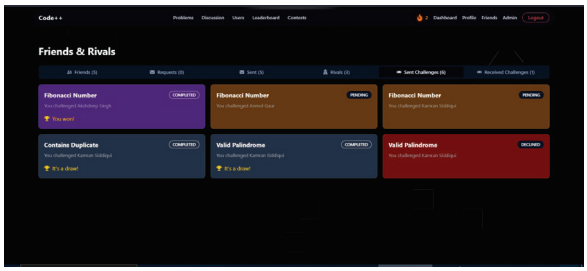

Fig. 7. Friends and rivals challenge system enabling users to compete with peers

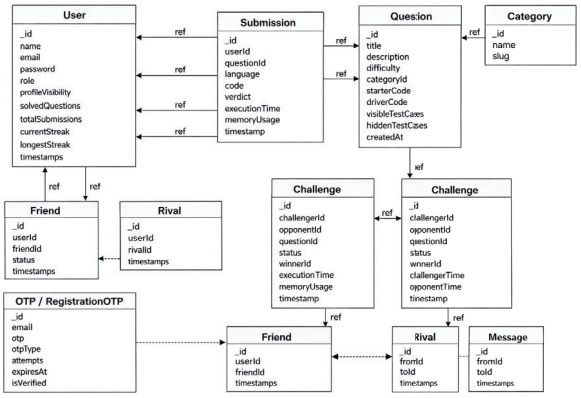

Fig. 8. Database design of the proposed competitive coding platform using MongoDB

programming skills among college students. While many online coding platforms exist, this system stands out because it focuses on institutions, has support from faculty, and includes gamification features designed for academic settings. The platform combines structured problem-solving practice, automated code evaluation using the Judge0 API, execution tracking, and competitive learning elements into one unified system. By giving real-time feedback on code submissions, the platform helps reduce the fear often linked to learning programming and allows students to find and fix mistakes quickly. Moreover, administrative and faculty control helps with organized task assignment, structured monitoring, and connection to academic goals of the institution.

The implementation and evaluation results showed that the system effectively increased student engagement, coding consistency, and logical problem-solving skills. The centralized design cuts down on dependence on scattered external resources and encourages a uniform coding culture at the institution. Gamification features like leaderboards, streaks, and badges fostered healthy competition and ongoing participation among students.

Although it achieved its main goals, the platform still has significant room for improvement. Future enhancements could involve adding AI-based analytics to provide personalized problem recommendations, including plagiarism detection tools to uphold academic integrity, adding timed mock placement tests, and creating advanced performance dashboards with data visualization tools for faculty. Expanding the system to support inter-college competitions and developing a mobile application could further enhance accessibility and scalability. With ongoing improvements and wider use across institutions, this platform has the potential to develop into a complete academic coding ecosystem that effectively prepares students for technical careers in the fast-changing digital world.

REFERENCES

[1] J. Hausladen et al., “A cloud-based integrated development environment,” Proc. IEEE MESA, 2014.
[2] S.-C. Su et al., “Web-based programming learning platform,” Proc. iFuzzy, 2016.
[3] A. Kanade et al., “Normalization in MongoDB,” ICCCI, 2014.
[4] F. Thung et al., “Social coding networks in GitHub,” CSMR, 2013.
[5] N. Zhang et al., “Software course group platform,” CSSS, 2011.
[6] A. J. Poulter et al., “RESTful services using MEAN stack,” IoT Cloud, 2015.
[7] I. K. Chaniotis et al., “Real-time web application with Node.js,” MobiWIS, 2013.
[8] S. Frerich et al., “Virtual labs for programming,” EDUCON, 2014.
[9] G. Shivacheva et al., “Virtual laboratory for programming,” CompSysTech, 2017.
[10] K. Davis and E. Roberts, “Collaborative code systems,” IEEE Software, 2017.
[11] K. Werbach and D. Hunter, “Gamification in education,” 2012.
[12] L. Hakulinen et al., “Gamification in programming courses,” ITiCSE, 2015.
[13] M. Ibanez et al., “Gamification for engagement,” Computers & Education, 2014.
[14] J. Bishop and M. Verleger, “Flipped classroom review,” ASEE, 2013.
[15] D. Boud and N. Falchikov, “Assessment and feedback,” Higher Education, 2006.
[16] P. Brusilovsky and E. Millan, “User modeling in adaptive systems,” 2007.
[17] R. Ihantola et al., “Automatic assessment tools,” Koli Calling, 2010.
[18] M. Joy et al., “Plagiarism detection systems,” IEEE Trans. Educ., 2005.
[19] A. Kapp, “The gamification of learning,” 2012.
[20] T. Clear et al., “Institutional programming environments,” ACM Inroads, 2011.