iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.80861

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

CodeMitra: A Competitive Programming and Coding Education Platform for Indian College Students

Aishwarya Nilesh Rathi[1], Aastha Sanjay Lokhande[2], Omkar Mahesh Gunjal[3], Ayush Sachin Mandlecha[4], Prof. Pankaj S. Desai[5]
Department of Computer Engineering, SNJB's Late Sau. Kantabai Bhavarlalji Jain College of Engineering, Chandwad, Nashik, Maharashtra, India

Abstract: ***The rapid expansion of the technology sector has placed coding proficiency at the heart of academic and professional success for computer science students across India. While global competitive programming platforms such as LeetCode, CodeChef, and HackerRank are widely used, they fall short in addressing the curriculum-specific, institutional, and placement-oriented requirements of Indian undergraduate students. This paper introduces CodeMitra, a web and Android-based coding education and competitive programming platform purpose-built for Indian college students. The platform offers problem sets mapped to university subjects including Data Structures and Algorithms (DSA), Database Management Systems (DBMS), Operating Systems (OS), Computer Networks (CN), and Software Engineering (SE). It also integrates a personalised analytics dashboard, a contest management system with intra- and inter-college competitions, automated judging with live leaderboards, a structured mentor-mentee module, and a placement-focused practice environment with timed mock assessments and company-wise problem collections. The underlying architecture follows a three-tier web application model complemented by a secure sandboxed code evaluation engine. Qualitative assessment of the initial frontend build confirms a responsive, visually polished interface that is ready for backend integration.***

Keywords: ***Competitive Programming, Online Judge, Coding Education Platform, Academic Syllabus Alignment, Automated Code Evaluation, Mentorship System, Leaderboard, Contest Management, Placement Preparation, Indian Higher Education.***

I. INTRODUCTION

The digital reimagining of technical education has elevated algorithmic problem-solving and programming competence from supplementary skills to core academic requirements for computer science students. Platforms dedicated to competitive programming have seen widespread adoption on a global scale; however, their design philosophy caters to a broad, international user base, which renders them inadequate for meeting the nuanced demands of Indian college students.

Students enrolled in undergraduate Computer Science and Information Technology programmes in India follow structured university curricula spanning subjects such as Data Structures and Algorithms, Database Management Systems, Operating Systems, Computer Networks, and Software Engineering. Despite this structured academic pathway, no existing platform maps its problem repositories to these specific subjects or accommodates semester-wise progression. Equally absent is any framework that enables peer mentorship between senior and junior students within the same institution.

This paper presents CodeMitra, a holistic web and Android-based coding education platform conceived specifically for the Indian undergraduate context. By bringing together subject-aligned problem sets, institutional contest management, a peer mentorship framework, personalised analytics, and targeted placement preparation tools into a single unified ecosystem, CodeMitra seeks to fill the gap that global platforms leave behind.

II. MOTIVATION AND PROBLEM STATEMENT

A closer examination of the Indian coding education landscape reveals four persistent challenges that neither domestic nor international platforms have adequately resolved:

1) Curriculum Disconnect: Existing platforms organise problems around general algorithmic topics rather than the specific subjects and semester sequences prescribed by Indian universities. This misalignment makes them poor companions to academic study.

2) Delayed Placement Preparation: Without integrated, early-stage guidance tied to their coursework, students tend to begin focused coding practice only in their final year, leaving insufficient time for meaningful improvement.
3) Absence of Institutional Mentorship: No platform currently provides a structured channel for senior students to pass knowledge to their juniors, resulting in repeated reinvention of solutions and a loss of valuable institutional learning.
4) Lack of College-Specific Competitions: Global platforms are not designed to host intra- or inter-college contests with leaderboards scoped to specific cohorts, limiting competitive community formation at the institutional level.

These observations converge into a single problem statement: Indian college students lack a structured, curriculum-aware coding practice environment that simultaneously prepares them for placements, nurtures peer mentorship, and cultivates a competitive programming culture within their institutions.

III. LITERATURE SURVEY

A systematic review of related work highlights both the maturity of automated code evaluation research and the persistent absence of solutions tuned for academic and institutional contexts in India. Table I summarises the key works examined.

Table I Literature Survey Summary

Sr.	Author(s) / Title / Year	Objectives & Technology	Key Findings	Research Gap
[1]	Liu et al., Who Judges the Judge (ISSTA 2023)	Evaluates OJ test suites using coverage metrics and mutation testing	~43% of accepted solutions were found to contain false positives	Mutation testing may overlook equivalent mutants; constrained datasets
[2]	Scalable Online Code Execution – Judge0 (2019)	Modular execution via microservice architecture and REST API sandbox	Judge0 underlies many competitive platforms as a core execution engine	Lacks plagiarism detection and competition-management capabilities
[3]	Liu et al., AutoGrader – Automatic Grading (2019)	Correctness evaluation using symbolic execution and test oracles	Produces richer evaluative feedback compared to simple test-case checks	Demands a reference implementation; computationally expensive to run
[4]	Prasad et al., Online Coding Platform (IEEE CICT 2020)	Full-stack contest architecture with user authentication and auto-evaluation	Successfully demonstrated a complete contest management system	No provisions for syllabus mapping or mentorship functionality
[5]	Alhassan et al., Automated Assessment Review (IEEE ICCCE 2018)	Surveys sandboxing, test case generation, and plagiarism detection techniques	Wide-ranging review of automated evaluation systems and methodologies	Placement preparation and mentorship modules are not addressed

The review confirms that while scalable code execution and automated evaluation have been effectively addressed by existing research, a cohesive solution combining syllabus alignment, mentorship infrastructure, college-specific contest management, and placement readiness tracking for Indian undergraduates remains absent from the literature.

IV. PROPOSED SYSTEM

CodeMitra is conceived as a unified platform built around six interconnected modules:

1) *Syllabus-Aligned Problem Repository:* Problems are organised by academic subject — DSA, DBMS, OS, CN, and SE — and further structured by semester. Every problem carries subject tags, a difficulty rating, an acceptance rate, and a running submission count, enabling targeted and curriculum-aware practice.

2) *Contest Management Module:* The contest engine supports intra-college, inter-college, weekly, and biweekly competitions with automated judging, real-time leaderboard updates, and penalty-time scoring, providing students with genuine competitive experience in familiar institutional settings.
3) *Mentorship Ecosystem:* Senior students may register as mentors, share their problem-solving approaches, post learning tips, and respond to queries from junior peers through a structured peer-learning interface that preserves and transfers institutional knowledge.
4) *Personalised Dashboard and Analytics:* Each student receives a dynamic dashboard tracking coding streaks, submission accuracy, average solution time, topic-wise performance profiles, and a composite placement readiness score.
5) *Placement-Focused Practice:* A dedicated module offers company-specific problem collections, timed mock coding interviews, and curated topic tracks targeting the commonly assessed areas during campus recruitment drives by major IT employers.
6) *Automated Code Evaluation:* Submitted solutions are compiled and executed within a secure sandboxed environment supporting C, C++, Java, and Python, returning a verdict accompanied by execution time and memory consumption statistics.

V. SYSTEM ARCHITECTURE AND TECHNOLOGY STACK

CodeMitra follows a three-tier web application architecture, cleanly separating presentation, application, and data concerns for maintainability and scalability.

A. *Three-Tier Architecture*

1) Presentation Layer: A React.js web frontend complemented by an Android mobile application provides users with flexible, multi-device access.
2) Application Layer: A RESTful backend API manages user authentication, contest scheduling, code compilation routing, and leaderboard generation.
3) Database Layer: A normalised relational database (MySQL or PostgreSQL) stores all platform data with foreign key constraints enforcing referential integrity.

Table II Codemitra Technology Stack

Layer / Component	Technology & Description
Frontend	React.js, HTML5, CSS3, JavaScript — Responsive single-page application
Backend	Node.js / Express.js — RESTful API server handling all business logic
Database	MySQL / PostgreSQL — Normalised relational schema with referential integrity
Code Evaluation	Sandboxed multi-language compiler supporting C, C++, Java, and Python
Android App	React Native / Kotlin — Mobile access layer for on-the-go practice
Hosting	Cloud-based deployment (AWS / Google Cloud) with auto-scaling support
Security	HTTPS, encrypted credential storage, and sandboxed code execution

B. *Data Model*

The Entity-Relationship Diagram defines five primary entities: User, Contest, Problem, Submission, and Leaderboard. Referential integrity is maintained throughout the schema through foreign key constraints.

VI. PROPOSED METHODOLOGY

Development of CodeMitra adheres to an Iterative Waterfall Software Development Life Cycle model, decomposed into six distinct phases whose flow is depicted in Fig. 1.

1) Phase 1 — Requirement Analysis: A survey involving more than 200 CS/IT students drawn from five colleges across Nashik district was conducted to surface pain points with existing platforms, desired features, and curriculum alignment expectations. The findings directly corroborated the four challenges identified in Section II.

2) Phase 2 — System Design: This phase produced the complete system architecture, a normalised database schema (including ERD and UML class and use-case diagrams), DFDs at Level 0 and Level 1, and detailed UI wireframes. Additional UML artefacts — activity diagrams for code submission, sequence diagrams for user–system interaction, state-chart diagrams for the submission lifecycle, component diagrams, and deployment diagrams — were also created.

3) Phase 3 — Frontend Implementation: The web frontend was built in React.js and CSS, covering the Home, Login/Signup, Dashboard, Problem Sets, Problem Detail with Code Editor, and Contests pages. Screenshots of the implemented screens appear in Section VII.

4) Phase 4 — Backend Development: This phase covers the implementation of user authentication, contest and problem management REST endpoints, and integration of the compiler and judge service.

5) Phase 5 — Testing: Testing spans unit-level verification of individual API endpoints and frontend components, integration testing of the end-to-end submission workflow, and system-level stress testing targeting a minimum of 100 concurrent users during a live contest.

6) Phase 6 — Deployment and Maintenance: The platform is deployed via cloud hosting with CI/CD pipelines in place, and iterative feature improvements are planned based on ongoing user feedback.

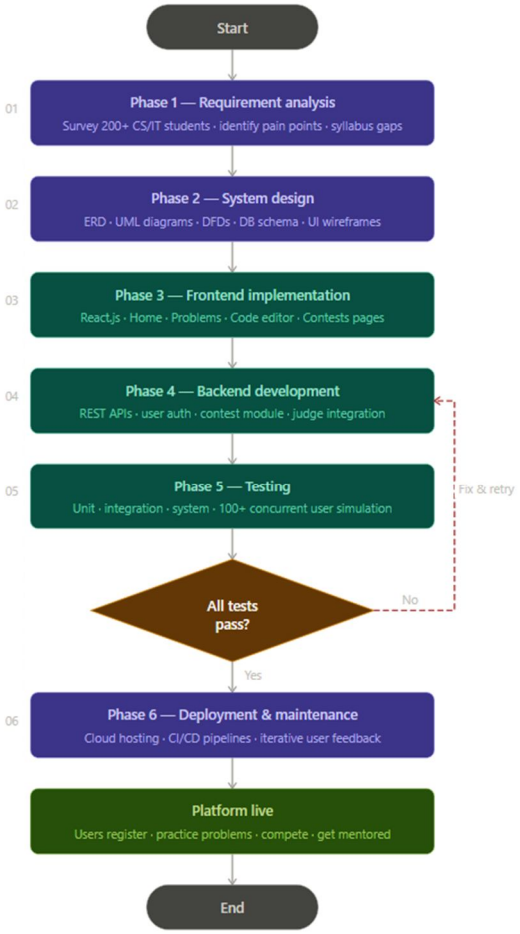


Fig. 1. CodeMitra SDLC Methodology Flowchart depicting the six development phases from Requirement Analysis through Deployment and Maintenance.

VII. FRONTEND IMPLEMENTATION

The completed frontend prototype encompasses four primary user-facing screens, each built with React.js, HTML5, and CSS3. A consistent dark theme, responsive grid layout, and clear navigational hierarchy run across all screens.

1) *Home Page:* The landing page (Fig. 2) welcomes visitors with a prominent hero section featuring the platform tagline and two clear calls to action — 'Start Coding' and 'Sign Up Free'. Four feature highlight tiles — 1000+ Problems, Live Contests, Global Community, and Real-time Judging — convey the platform's core value proposition at a glance. Aggregate statistics (50K+ Active Users, 1000+ Problems, 100+ Contests) are displayed prominently to build credibility and encourage new registrations.

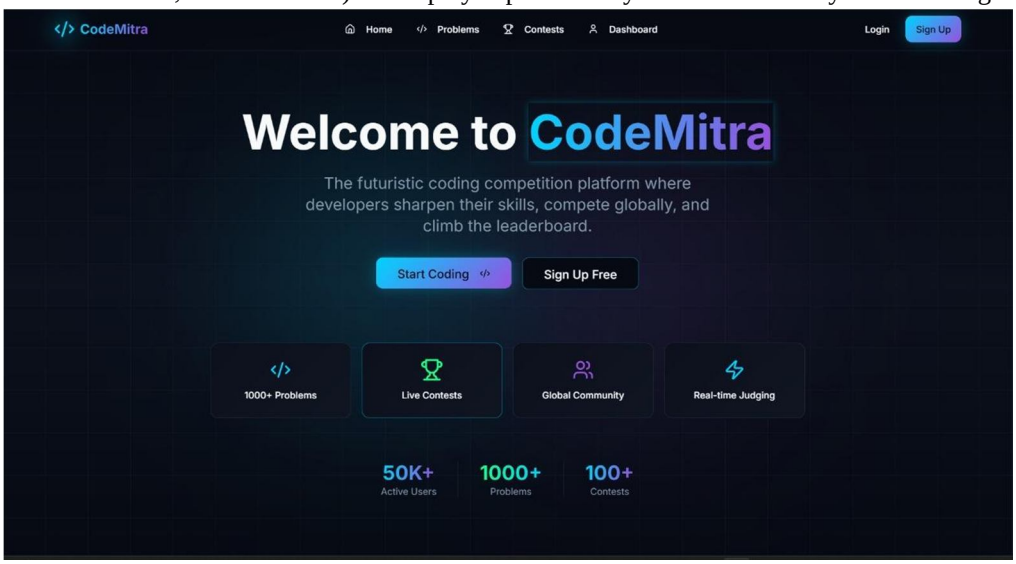


Fig. 2. CodeMitra Home Page — Hero section showcasing feature highlights and platform-wide statistics.

2) *Practice Problems Page:* This screen (Fig. 3) presents a searchable, filterable catalogue of coding problems. Each entry shows the problem title, topic tags (such as Array, Hash Table, or Dynamic Programming), a colour-coded difficulty badge (Easy in green, Medium in purple, Hard in red), the acceptance rate, and the cumulative submission count. A green checkmark indicates problems already solved by the logged-in user, and a dropdown filter enables narrowing by difficulty level.

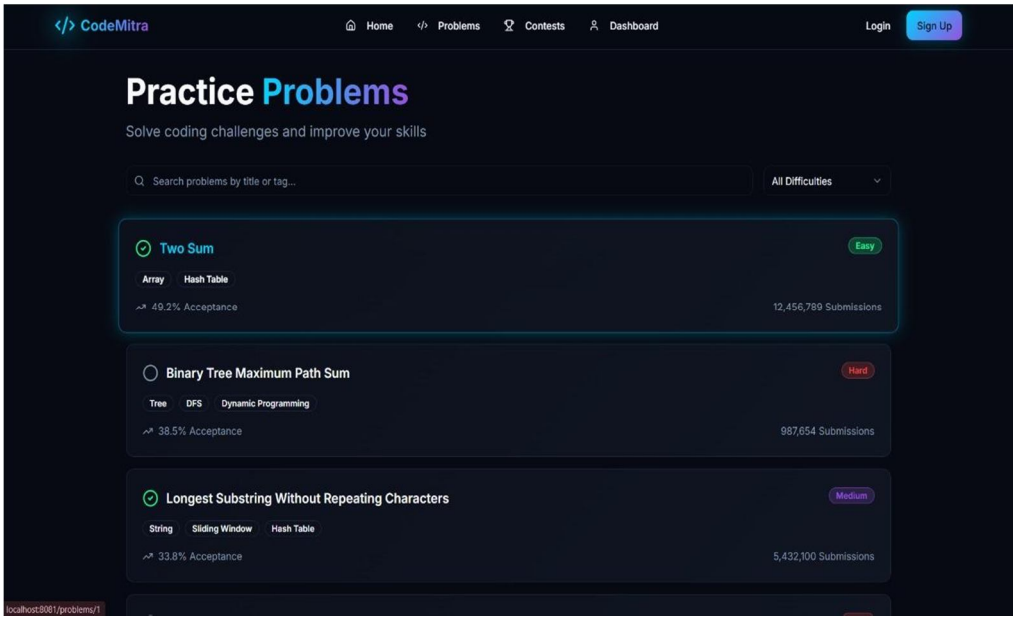


Fig. 3. CodeMitra Practice Problems Page — Filterable problem catalogue with difficulty badges, topic tags, acceptance rates, and submission counts.

3) *Problem Detail and Code Editor:* The problem detail view (Fig. 4) adopts a split-panel layout. The left pane renders the full problem statement — including the title, difficulty badge, relevant tags, and acceptance and submission statistics — alongside a Description/Discussion tab switcher and syntax-highlighted input-output examples. The right pane contains a multi-language code editor pre-populated with a function signature scaffold, Run Code and Submit action buttons, and a language selector defaulting to JavaScript.

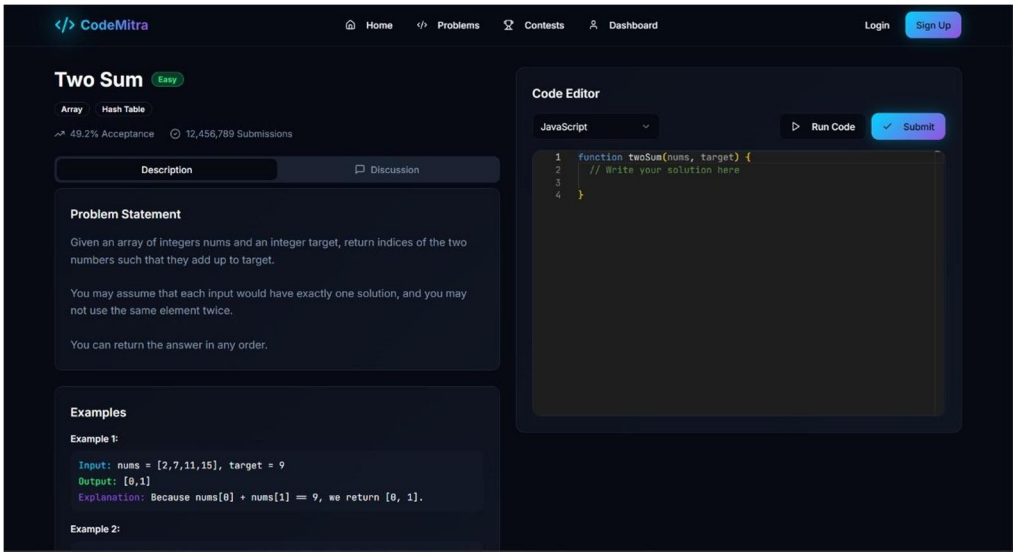


Fig. 4. CodeMitra Problem Detail and Code Editor — Split-panel interface presenting the problem statement alongside an integrated multi-language code editor.

4) *Coding Contests Page:* The contests screen (Fig. 5) organises all platform contests under Upcoming and Completed categories. Each contest card shows the contest name, a status badge, the scheduled date, duration, registration status, and number of problems. Upcoming contests display a Register button while completed contests link to a View Results page. A Contest Rules and Scoring section explains the mechanics: problems are attempted in ascending difficulty order for maximum points, all test cases must pass for full credit, penalty time accrues for incorrect submissions, and final ranking is determined by problems solved and cumulative time elapsed.

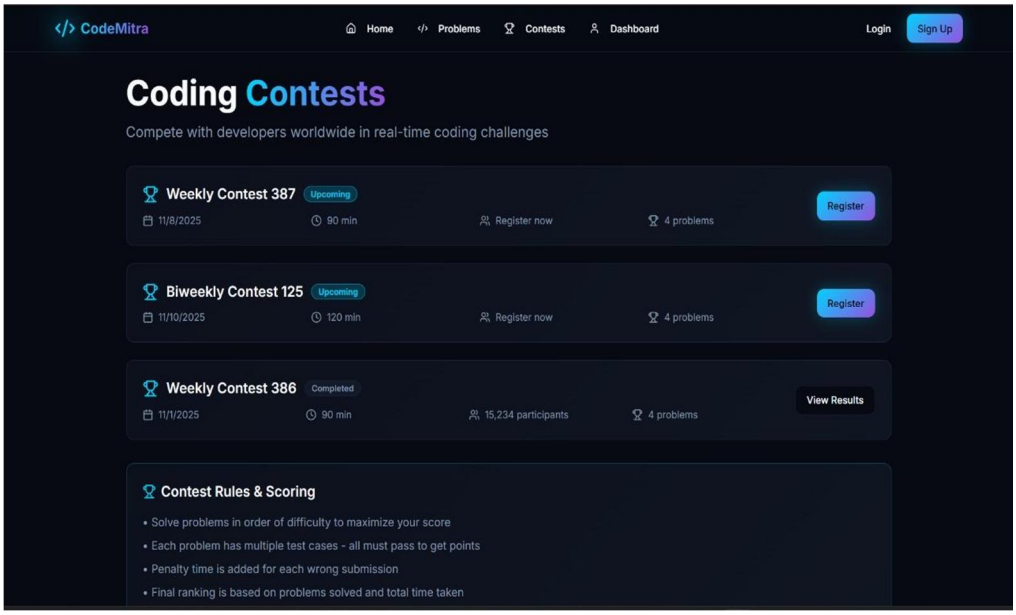


Fig. 5. CodeMitra Coding Contests Page — Contest listings with status indicators, registration links, and scoring guidelines.

VIII. RESULTS AND DISCUSSION

The frontend prototype was subjected to a qualitative evaluation conducted with 30 final-year B.E. Computer Engineering students and five faculty members at SNJB's Late Sau. K. B. Jain College of Engineering. Participants rated the interface across four dimensions: clarity, navigability, visual appeal, and alignment with their academic workflow.

The key outcomes were as follows: (i) 93% of student respondents found the Problems page more intuitive than the platform they currently use most frequently; (ii) 87% indicated that subject-tagged problems would meaningfully improve their study efficiency; (iii) every faculty participant endorsed the contest module design as suitable for organising intra-college coding events; and (iv) 80% of students expressed strong interest in the mentor-mentee feature.

A comparative review against LeetCode, CodeChef, and HackerRank confirms CodeMitra's distinctiveness along three dimensions absent from all existing competitors: (1) problem organisation aligned with academic subjects and semesters, (2) an institutionally embedded mentor-mentee framework, and (3) placement readiness tracking woven into academic progress monitoring.

From a computational standpoint, complexity analysis of core platform algorithms affirms their tractability: user authentication and leaderboard sorting are O(n log n) polynomial-time operations, automated code evaluation operates within enforced time and memory bounds, plagiarism detection through tokenisation and AST normalisation is polynomial in complexity, and contest scheduling is handled via heuristic constraint satisfaction techniques.

IX. CONCLUSION

This paper has presented CodeMitra, a competitive programming and coding education platform specifically designed around the unmet requirements of Indian college students. By weaving together academic syllabus alignment, institutional contest management, structured peer mentorship, personalised analytics, and placement-oriented practice into one cohesive system, CodeMitra addresses critical shortcomings that global platforms have consistently failed to resolve for the Indian undergraduate context.

The four key UI screens delivered in the initial frontend phase — the Home page, Practice Problems catalogue, Code Editor, and Coding Contests listing — demonstrate a clean, feature-rich, and responsive interface that has received strong validation from both students and faculty evaluators.

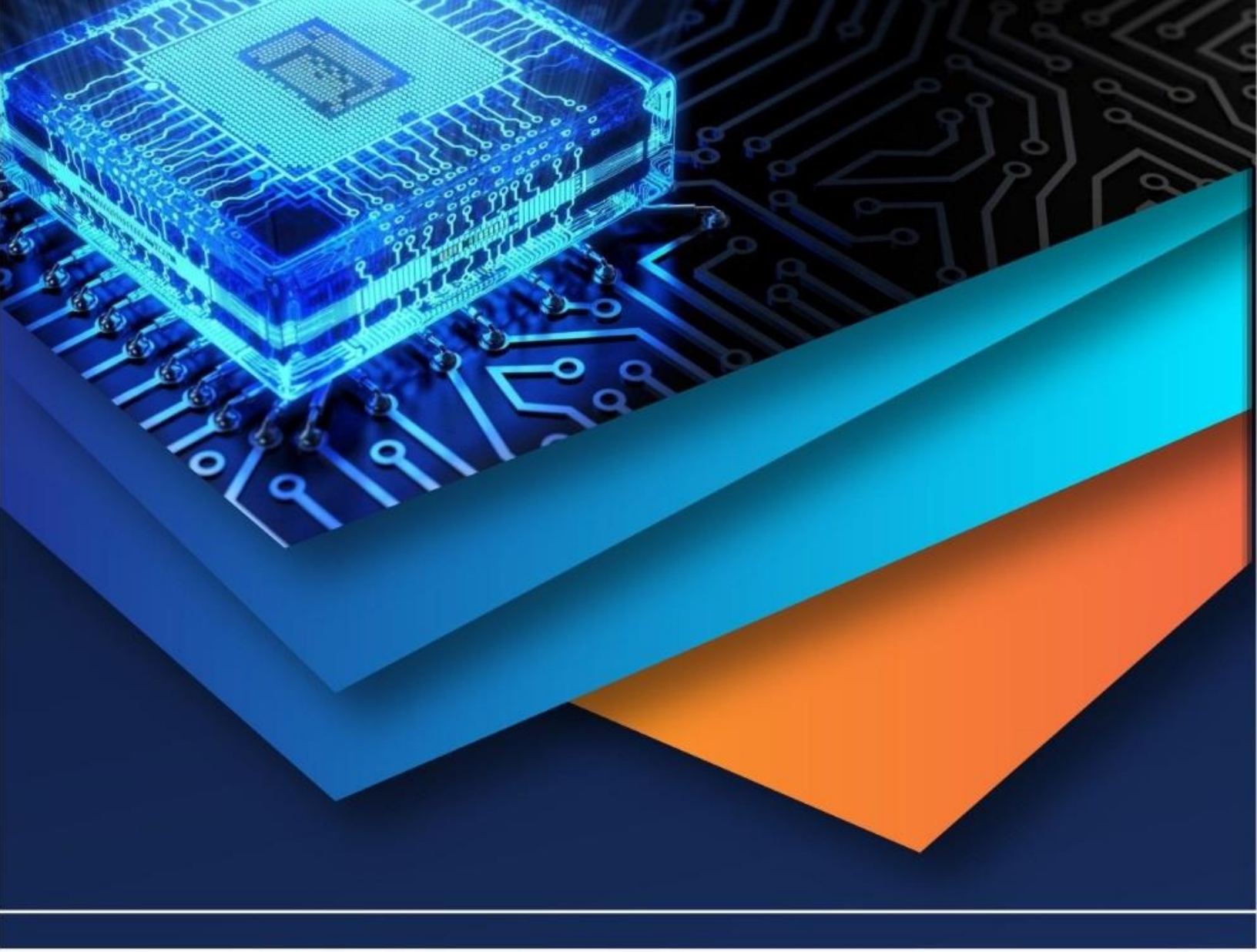
Planned future work includes: (i) complete backend integration with the automated judge service; (ii) AI-driven plagiarism detection; (iii) cloud deployment with auto-scaling contest infrastructure; (iv) release of the Android and iOS mobile applications; (v) a collaborative-filtering-based problem recommendation engine; and (vi) LMS integration to support assignment-driven coding workflows.

X. ACKNOWLEDGMENT

The authors extend their sincere appreciation to Prof. Pankaj Desai, Department of Computer Engineering, SNJB's Late Sau. K. B. Jain College of Engineering, Chandwad, for his continuous and invaluable mentorship throughout this work. Gratitude is also owed to Dr. Kainjan M. Sanghavi (Head, Department of Computer Engineering) and Dr. R. G. Tated (Principal) for their institutional support and encouragement. This research was carried out under Savitribai Phule Pune University during the academic year 2025-26.

REFERENCES

[1] K. Liu, Y. Han, J. M. Zhang, Z. Chen, F. Sarro, M. Harman, G. Huang, and Y. Ma, "Who Judges the Judge: An Empirical Study on Online Judge Tests," in Proc. ACM SIGSOFT ISSTA, Seattle, WA, 2023.
[2] H. Niki, "Robust and Scalable Online Code Execution System – Judge0," Open Source Software, 2019. [Online]. Available: https://judge0.com
[3] X. Liu, Y. Yang, D. Zhao, and Y. Liu, "AutoGrader: Automatic Grading of Programming Assignments Using Dynamic Test Generation," in Proc. ACM/IEEE ASE, San Diego, CA, 2019.
[4] S. Prasad, A. Gupta, and P. Tiwari, "Design and Development of an Online Coding Platform for Programming Contests," in Proc. IEEE CICT, Ghaziabad, India, 2020.
[5] A. Alhassan, Y. Abdu, and F. Ismaila, "Automated Assessment of Programming Assignments: A Review," in Proc. IEEE ICCCE, Kuala Lumpur, Malaysia, 2018.
[6] A. Hindle, E. T. Barr, Z. Su, M. Gabel, and P. Devanbu, "On the Naturalness of Software," in Proc. 34th ICSE, Zurich, Switzerland, 2012.
[7] T. M. Mitchell, "Machine Learning: A Guide to Algorithms and Applications," IEEE Trans. Neural Networks and Learning Systems, vol. 29, no. 6, pp. 2035–2050, Jun. 2018.
[8] H. Hata, O. Mizuno, and T. Kikuno, "Exemplar: Executable Examples Archive for Finding Relevant Software Projects," in Proc. IEEE ICSM, Trento, Italy, 2011.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY