International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 05 | May – 2025 DOI: 10.55041/ISJEM03394
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Design and Implementation of a Real-Time Feedback System for Enhancing College-Level Coding Proficiency

Babasaheb Satpute
Dept of CSE
FET, JAIN (Deemed to be University)
Bangalore, India
babasaheb.s@jainuniversity.ac.in

Abu Bakar Shibli
Dept of CSE
FET, JAIN (Deemed to be University)
Bangalore, India
21btrcs107@jainuniversity.ac.in

Samiksha Shahi Dept of CSE
FET, JAIN (Deemed to be University)
Bangalore, India
21btrcs204@jainuniversity.ac.in

S.S.S. Dhyuthidhar
Dept of CSE
FET, JAIN (Deemed to be University)
Bangalore, India
21btrcs207@jainuniversity.ac.in

Shabbir Poonawala
Dept of CSE
FET, JAIN (Deemed to be University)
Bangalore, India
21btrcs070@jainuniversity.ac.in

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 05 | May – 2025 DOI: 10.55041/ISJEM03394
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

***Abstract*— This paper presents the design and implementation of a real-time feedback system for enhancing coding proficiency among college students. The Code Problem Platform (CPP) is a full-stack application that integrates immediate error detection, detailed corrective insights, and persistent progress tracking to create an interactive learning environment for programming education. The system architecture follows a phased development approach: Phase 1 establishes a robust, multi-language code editor built on Monaco Editor with Judge0 API integration for code execution, while Phase 2 implements advanced features including syntax and logical error detection, local Storage-based progress persistence, and role-based access controls for academic integration. Technical challenges addressed include feedback generation for multiple programming languages, efficient state management across user sessions, and integration of real-time evaluation systems. The platform's architecture is designed to bridge the gap between academic programming courses and industry requirements through a comprehensive system that simulates real-world coding scenarios while providing contextual guidance. Future work includes empirical validation of the system's educational impact through controlled studies with student participants.**

Keywords— Real-time feedback systems, Programming education platform, Full-stack application design, Monaco Editor integration, Judge0 API implementation, Client-side state persistence, Educational technology architecture, Curriculum integration systems, Role-based access control, Programming feedback algorithms

I. INTRODUCTION

In today's technology-driven world, coding proficiency has become a cornerstone of education, especially for college students pursuing degrees in computer science and engineering. Despite the availability of numerous online coding platforms, a persistent challenge remains: the provision of timely and effective feedback. Real-time feedback is critical as it enables students to immediately identify and correct mistakes, fostering a cycle of continuous improvement and self-regulated learning (Aleven et al., 2016)[1]. This approach not only enhances understanding of complex programming concepts but also supports the development of practical problem-solving skills that are essential in both academic and professional contexts.

This paper presents the design of a system that aims to address this challenge by integrating immediate error detection and personalized feedback into its learning environment. The Code Problem Platform's (CPP) phased development—starting with the establishment of a versatile code editor and moving towards the incorporation of real-time feedback mechanisms— mirrors the evolving needs of learners. The primary objectives of this paper are to present a system architecture for delivering real- time feedback to enhance coding proficiency, describe the technical implementation of problem-solving support features, and outline how the platform's design bridges the gap between classroom instruction and industry demands. By focusing on these areas, this work contributes to a growing body of literature that underscores the importance of immediate, constructive feedback in enhancing educational outcomes (Nicol & Macfarlane-Dick, 2006[2]; Hattie & Timperley, 2007[3]; Deterding et al., 2011[4]).

II. LITERATURE SURVEY

Educational coding platforms like LeetCode, HackerRank, and Codeforces have become go-to resources for aspiring developers looking to sharpen their technical skills. As [5] points out, these platforms offer a well-structured learning space where users can tackle a wide range of coding problems while receiving instant feedback to improve their approach.
[6] further emphasize how these platforms do more than just teach coding—they recreate real-world problem-solving scenarios and foster a sense of friendly competition. This combination of hands-on practice and community-driven learning makes them an invaluable tool for anyone preparing for technical interviews or looking to level up their coding skills.

Gamification has transformed the way people learn by making education more interactive and enjoyable. In essence, it involves incorporating game-like features—such as leaderboards, badges, and point systems—to enhance learner engagement and motivation. Deterding, Dixon, Khaled, and
[4] explain that these features help create an immersive experience, turning learning into something students actually look forward to. [7] back this up with research showing that gamification doesn't just make learning fun—it also boosts academic performance by encouraging consistent participation and reinforcing positive study habits.

Real-time feedback is a game-changer in coding education, helping learners catch mistakes right away and build better programming habits. [3] highlight that getting instant, detailed feedback speeds up the learning process by allowing students to tweak their approach as they go. [2] add that this kind of quick, guiding feedback is especially important in fast- moving fields like computer science, where constant learning and self-improvement are key.

Every student learns differently, and personalized learning helps cater to those unique needs. [8] stress how adaptive educational systems can shape content based on each learner's strengths and progress. In coding education, this means adjusting problem difficulty, offering helpful hints, and aligning challenges with specific coursework. By making sure students tackle problems that are both relevant and suited to their skill level, personalized learning keeps them engaged and helps them grow at their own pace.

There's often a noticeable gap between what students learn in school and what employers actually expect in the workplace.
[9] highlight how important it is to bring real-world problem-solving into academic programs to better prepare students for their careers. By incorporating industry-relevant challenges and focusing on hands-on experience, education can align more closely with industry needs—helping graduates step into jobs with the right skills and confidence.

Lastly, mastering algorithms and data structures is a game-changer when it comes to cracking technical interviews and excelling in software development roles. [10] have long been the go-to experts in this field, providing the essential theory and

problem- solving techniques that every programmer needs. [11] further emphasizes that a strong grasp of these concepts isn't just helpful—it's often the deciding factor in whether a candidate succeeds in high-pressure coding interviews. To excel in the tech industry, it is highly recommended to build a strong foundation in algorithms and data structures.

Development Phases

The system development was structured in two distinct phases to ensure pedagogical principles were integrated with technical implementation

Phase 1: Core Platform Development

- Implementation of the Monaco Editor with language- specific configurations
- Development of the application state management system
- Integration of Judge0 API for secure code execution

III. METHODOLOGY

A. Technical Architecture and Development Approach

This paper presents the design and implementation of the Code Problem Platform (CPP), a full-stack system developed to provide real-time feedback for programming education. The architecture follows a modern web application approach with clear separation of concerns between frontend and backend components.

1) Technical Stack Selection

The platform leverages the following technologies, selected for their suitability in educational applications:

- **Frontend Framework**: Next.js with React for server-side rendering capabilities and improved performance
- **Code Editor Integration**: Monaco Editor, the same technology powering VS Code, for a professional development experience
- **State Management**: React Context API (via AppContext) for application-wide state management combined with localStorage for persistent user progress
- **Backend Framework**: Node.js with Express.js providing RESTful API endpoints.
- **Database Architecture**: PostgreSQL with Sequelize ORM for structured data storage
- **Code Execution Engine**: Judge0 API for secure, sandboxed code execution across multiple programming languages
- Basic user authentication and session management
- Creation of the question repository and navigation system

Phase 2: Feedback Systems and Academic Integration

- Development of real-time feedback mechanisms based on established educational theories
- Implementation of role-based access with authentication.
- Teacher-specific management tools for curriculum alignment.
- Student progress tracking and analytics for learning assessment

B. Key System Components

The CPP was implemented as a full-stack web application designed to provide real-time feedback, track progress, and simulate placement-level coding challenges.

1) Question Management System

The questions-page component implements a comprehensive problem browsing and filtering system designed to present coding challenges in an accessible format. Features include difficulty-based filtering, search functionality, and visual completion indicators that provide students with clear progression paths through the curriculum.

2) Code Editor Implementation

The code-editor component serves as the core coding interface, providing syntax highlighting for multiple programming languages, auto-completion, and error detection. The editor is configured specifically for educational purposes, balancing professional features with an approachable interface for students at various skill levels.

3) Real-Time Feedback System

The question-detail-page component implements the feedback system based on principles of immediate evaluation and targeted guidance. The system provides detailed output formatting for error messages, test case visualization, and implements a points system for gamification—a technique shown to increase engagement in educational contexts.

4) User Authentication and Role Management

The authentication system differentiates between student and teacher roles, providing tailored experiences for each. Teachers gain access to problem creation tools and analytics, while students receive a focused problem-solving environment with appropriate scaffolding.

5) Application State Management

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 05 | May – 2025 DOI: 10.55041/ISJEM03394
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

The app-context implementation provides comprehensive state management that maintains consistency across the application while ensuring user progress persists between sessions—a critical feature for maintaining continuity in the learning experience.

C. System Integration and Data

The architecture follows a clear flow of data as illustrated in Fig.1, from user authentication through problem selection, code submission, execution, feedback generation, and progress tracking. This flow is designed to minimize friction in the learning process, allowing students to focus on coding concepts rather than system navigation.

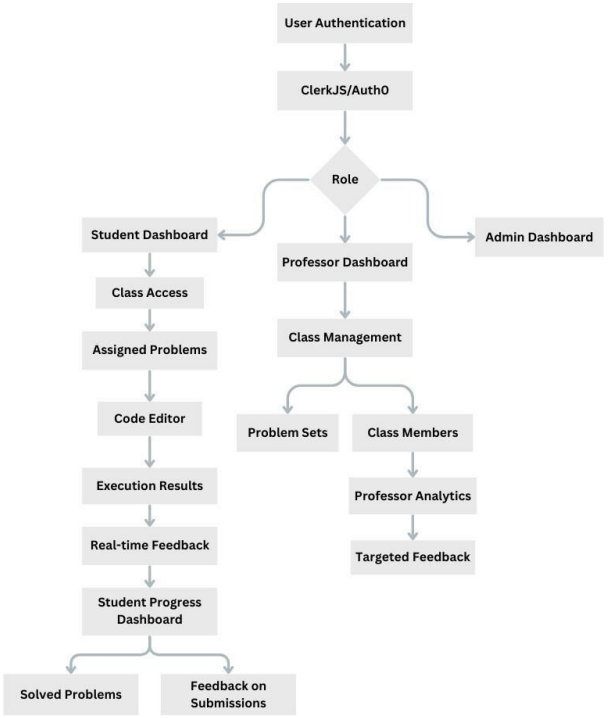

Fig. 1. Architecture Flow Chart for the CPP

D. Technical Challenges and Educational Solutions

1) Challenge 1: Offline Resilience and Learning Continuity

A critical challenge was ensuring the platform remained functional even when backend services were unavailable, thereby maintaining educational continuity.

Solution: Implementation of a comprehensive fallback system that detects API unavailability and switches to localStorage-based operation with client-side evaluation where possible, ensuring students can continue learning regardless of connectivity issues.

2) Challenge 2: Code Execution Security with Educational Freedom

Balancing the need for secure code execution with providing students freedom to experiment presented significant challenges.

Solution: Integration with Judge0 API with appropriate resource limitations, combined with a rate limiting system that prevents abuse while still allowing students to explore different approaches to problem-solving.

3) Challenge 3: Progress Persistence for Learning Assessment

4)

5) Maintaining consistent tracking of student progress was essential for both motivation and assessment purposes.

Solution: A hybrid approach using localStorage for client-side persistence combined with server synchronization when available, enabling students to track their own progress while providing instructors with assessment data.

6) Challenge 4: Multi-Language Support for Diverse Learning Paths

Supporting multiple programming languages while providing consistent feedback required careful implementation to accommodate diverse learning preferences.

Solution: Development of language-specific templates and feedback formatting, ensuring a consistent educational experience regardless of a student's chosen programming language.

This architecture creates an educational environment that provides immediate, actionable feedback while maintaining robustness even in challenging network conditions—bridging the gap between academic learning and industry-standard development environments.

IV. SYSTEM CAPABILITIES AND FUTURE EVALUATION

A. System Capabilities Assessment

The Code Problem Platform (CPP) implementation has resulted in several key capabilities that address common challenges in programming education:

1) Real-Time Feedback Mechanisms

The platform successfully implements multi-layered feedback through:

1. **Syntax Error Detection**: Immediate identification of coding errors through Monaco Editor's integration with language services.
2. **Logical Error Analysis**: Execution against test cases to identify algorithmic and logical mistakes.
3. **Performance Assessment**: Analysis of execution time and memory usage when available.

These feedback mechanisms have been technically validated through system testing, though formal user

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 05 | May – 2025 DOI: 10.55041/ISJEM03394
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

studies remain a future objective.

2) Cross-Platform Persistence

The hybrid approach to state management enables several important capabilities:

1. **Offline Functionality**: Students can continue working even when disconnected from the server.
2. **Progress Resumption**: Work is preserved between sessions through localStorage integration.
3. **Multi-Device Consistency**: Server synchronization ensures progress remains consistent across devices.

3) Educational Integration Features

The platform includes several features designed specifically for academic integration:

1. **Role-Based Access**: Differentiation between student and educator interfaces.
2. **Custom Problem Sets**: Educators can create and assign curriculum-aligned problem sets.
3. **Performance Analytics**: Basic metrics on problem completion and attempts are tracked.

B. Technical Limitations

Despite the successful implementation, several technical limitations have been identified:

1) **Language Support Variability**: The feedback quality varies across programming languages, with JavaScript and Python receiving more comprehensive support than C++ or Java due to differences in language service capabilities.

2) **Execution Limitations**: The reliance on Judge0 API introduces potential bottlenecks during high traffic periods, requiring implementation of rate limiting and fallback mechanisms.

3) **Feedback Precision Challenges**: Providing meaningful feedback for complex algorithmic problems remains challenging, particularly for open-ended assignments without clearly defined test cases.

4) **Client-Side Performance**: Real-time feedback processing can impact performance on lower-end devices, requiring careful optimization of the feedback generation pipeline.

.

C. Comparative Feature Analysis

When compared to conventional coding platforms without curriculum alignment capabilities, CPP offers several distinctive features:

TABLE I. COMPARATIVE FEATURE ANALYSIS BETWEEN CPP AND TRADITIONAL PLATFORMS

Feature	Code Problem Platform	Traditional Platforms
Curriculum-specific problems	Supported through professor customization	Limited or None
Error detection approach	Comprehensive (syntax and logic)	Basic (primarily syntax)
Feedback response design	Immediate (2-3 seconds target)	Variable or delayed
Performance analytics	Detailed progress tracking	Limited or general
Academic integration	Direct curriculum alignment	Minimal or none

V. CONCLUSION

This paper presents the design and implementation of the Code Problem Platform (CPP), a full-stack application aimed at enhancing coding proficiency through real-time feedback mechanisms. The platform integrates several innovative components:.

A. **Multi-language Code Editor:** Built on Monaco Editor with syntax highlighting and auto-completion for multiple programming languages, providing a professional-grade coding environment for students.

B. **Real-time Feedback System:** Designed to deliver immediate insights on syntax errors, logical mistakes, and optimization opportunities through a combination of static analysis and runtime evaluation.

C. **Progress Tracking Architecture:** Implementing a hybrid approach with both server-side database storage and client- side localStorage persistence to maintain seamless user progress across sessions.

D. **Navigation System**: Utilizing React Router to create an intuitive flow between different platform components (problem browser, code editor, performance dashboard), enhancing user experience.

E. **Multi-level Access Control**: Supporting differentiated roles for students and professors, enabling curriculum alignment and classroom integration through custom problem sets.

The technical architecture of CPP demonstrates how modern web technologies can be leveraged to create educational platforms that bridge the gap between academic learning and industry requirements. By focusing

on immediate feedback and contextual learning, the platform's design addresses the limitations of traditional programming education approaches.

F. Future Work

1) Empirical Evaluation

Future work will focus on conducting a thorough empirical evaluation of the platform using the methodology outlined in the previous section. This will include:

- Controlled experiments with student participants to measure the impact of real-time feedback on coding proficiency
- Comparative analysis between students using the platform and traditional learning methods
- Longitudinal studies to assess long- term skill retention and development

2) Technical Enhancements

Several technical improvements are planned for future development:

- **Advanced Error Detection**: Implementing machine learning algorithms to identify complex logical errors and provide more targeted feedback based on common mistake patterns.
- **Collaborative Features**: Adding real- time collaboration capabilities to allow peer programming and code review activities among students.
- **Enhanced Analytics**: Developing comprehensive learning analytics to provide instructors with insights into student performance, common misconceptions, and learning obstacles.
- **Offline Functionality**: Expanding the platform's capabilities to function effectively in environments with limited connectivity through enhanced client-side processing.
- **Mobile Support**: Optimizing the interface for mobile devices to enable learning on the go.

3) Educational Integration

To maximize educational impact, future work will also include:

- **Curriculum Mapping**: Creating a framework that allows educators to map platform problems directly to course learning objectives and assessment criteria.
- **Industry Problem Sets**: Collaborating with industry partners to develop problem sets that reflect real-world coding challenges faced in technical interviews and workplace scenarios.
- **Automated Assessment Integration**: Developing APIs to integrate with educational learning management systems for seamless assignment creation and grading.

The CPP platform represents an important step toward creating more effective programming education tools. With continued development and validation, it has the potential to significantly contribute to computer science education by providing students with the immediate, contextual feedback that research has shown to be critical for skill development in complex domains.

REFERENCES

[1] V. Aleven, E. A. McLaughlin, R. A. Glenn, and K. R. Koedinger, "Instruction based on adaptive learning technologies," in *Handbook of Research on Learning and Instruction*, 2nd ed., R. E. Mayer and P. A. Alexander, Eds. New York, NY, USA: Routledge, 2016, pp. 123–145.

[2] D. Nicol and D. Macfarlane-Dick, "Formative assessment and self-regulated learning: A model and seven principles of good feedback practice," *Stud. High. Educ.*, vol. 31, no. 2, pp. 199–218, Apr. 2006.

[3] J. Hattie and H. Timperley, "The power of feedback," *Rev. Educ. Res.*, vol. 77, no. 1, pp. 81–112, Mar. 2007.

[4] S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gamefulness: Defining 'gamification'," in *Proc. 15th Int. Academic MindTrek Conf.*, Tampere, Finland, 2011, pp. 9–15.

[5] L. Liu, "The role of online coding platforms in computer science education," *J. Comput. Sci. Educ.*, vol. 27, no. 3, pp. 182–195, 2019.

[6] M. Ramírez and C. Peña, "Enhancing learning through online coding competitions: A case study," *Int. J. Comput. Sci. Educ.*, vol. 23, no. 4, pp. 289–305, 2018.

[7] J. Hamari, J. Koivisto, and H. Sarsa, "Does gamification work? — A literature review of empirical studies on gamification," in *Proc. 47th Hawaii Int. Conf. Syst. Sci. (HICSS)*, Waikoloa, HI, USA, 2014, pp. 3025–3034.

[8] P. Brusilovsky and E. Millán, "User models for adaptive hypermedia and adaptive educational systems," in *The Adaptive Web: Methods and Strategies of Web Personalization*, P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds. Berlin, Germany: Springer, 2007, pp. 3–53.

[9] L. Johnson, S. Adams Becker, V. Estrada, and A. Freeman, *The NMC Horizon Report: 2015 Higher Education Edition*. Austin, TX, USA: The New Media Consortium, 2015.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.

[11] G. L. McDowell, *Cracking the Coding Interview: 189 Programming Questions and Solutions*, 6th ed. Palo Alto, CA, USA: CareerCup, 2015.