IJARSCT
ISSN: 2581-9429
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Volume 5, Issue 1, December 2025
Impact Factor: 7.67

Code Arena

Afshin R Jamadar, Ananya S, Anushree Jaiswal, Mrs Akshatha Preeth
Department of Information Science and Engineering
Global Academy of Technology, Bengaluru, India
afshinjamadar963@gmail.com, ananyas272005@gmail.com, anushree416jaiswal@gmail.com

Abstract: *The project aimed to create "CODEARENA – Code Together, Learn Together", a real-time collaborative coding platform for time-bound challenges. Participants join sessions and solve problems within 10–30 minutes. The system includes a collaborative code editor, timer-based challenge module, and automated test case validation. It ensures the live activity tracker updates within 2 seconds and requires 80–100% test case success for solution acceptance.*

Keywords: Collaborative coding, real-time editor, code challenge, time-bound tasks, live activity tracker, automated validation, test case evaluation, latency, teamwork, learning platform

I. INTRODUCTION

CODEARENA – Code Together, Learn Together In today's rapidly evolving technological landscape, students, aspiring developers, and coding enthusiasts often struggle to gain the kind of real-time, collaborative experience that mirrors modern software development environments. Traditional learning platforms frequently emphasize individual progress, leaving a gap between academic problem-solving and the teamwork-driven demands of industry. Together—also referred to as Code Arena—was designed to bridge this gap by providing a dynamic, real-time collaborative coding platform that immerses users in a setting similar to technical interviews, hackathons, and real-world engineering challenges.

1.1 The Gap in Collaborative Programming and Skill Development

In universities, coding bootcamps, and technical training programs, thousands of students practice programming every day. However, most of this learning occurs in isolated, individual environments where collaboration, code transparency, and real-time teamwork are limited. As a result, students graduate with strong theoretical knowledge but insufficient experience in collaborative coding—an essential component of modern software development workflows. Existing platforms such as learning management systems (LMS) and standalone coding portals primarily focus on single-user submissions, offering limited interaction between learners. This lack of real-time collaboration often leads to inefficient learning, reduced problem-solving exposure, and increased chances of plagiarism.

Current coding education tools face several foundational issues: restricted peer interaction, minimal transparency, no contribution tracking, and weak enforcement of academic integrity. Without structured collaboration, students are unable to practice team-based coding scenarios commonly used in interviews, hackathons, and industry-level development. Furthermore, plagiarism incidents continue to rise due to the absence of tools that highlight user contributions in shared problem-solving tasks.

These challenges emphasize the need for a platform that not only supports coding but also enables real-time teamwork, transparency, originality, and verifiable contribution tracking, ensuring that learning aligns closely with industry expectations.

1.2 Code Arena – A Real-Time Collaborative Coding Platform for Authentic Skill Building

To address these persistent limitations, this work introduces Code Arena, an interactive and real-time collaborative coding platform engineered to enhance skill development, teamwork, and authenticity in programming education. Code Arena provides a live shared environment where multiple users can simultaneously contribute to the same codebase, enabling transparent collaboration and reducing opportunities for academic dishonesty.

Unlike traditional coding tools, Code Arena integrates:

- A live collaborative editor,
- Automated test case validation,
- Real-time activity and contribution tracking,
- Time-based challenges,
- Integrated communication features, and
- Secure sandboxed execution environments.

These features mirror the engineering practices used in modern workplaces—pair programming, peer debugging, and collaborative problem-solving.

Code Arena is built not merely as a theoretical solution but as an operational, scalable platform that encourages teamwork, communication, originality, and high-pressure problem-solving.

By enabling transparent collaboration and real-time observation, Code Arena significantly mitigates plagiarism risks and establishes a culture of accountability. Every contributor's actions—typing, modifying logic, or executing solutions—are recorded and displayed, making plagiarism detectable and discouraging dishonest practices.

1.3 System Features, Educational Benefits, and Skill Impact

Code Arena is designed to operate as a comprehensive learning ecosystem that supports technical skill development, soft-skill enhancement, and academic integrity. The platform offers a suite of features that together create an environment suitable for classroom learning, competitive programming, coding clubs, and institutional assessments.

Key capabilities include:

- Real-time Collaborative Editing - Allows multiple users to write and debug code simultaneously with live visibility.
- Automated Evaluation-Test cases run instantly, providing immediate feedback and teaching logical correctness under pressure.
- Activity Tracking -Similar to verification in decentralized systems ,this ensures transparency and fairness by recording each user's contributions.
- Timed Challenges and Leaderboards-Encourage healthy competition and improve time-bound problem-solving skills.
- Secure Execution Environment-Isolated runtime prevents data leakage, unauthorized access, or code manipulation.

Educational Impact:

Enables peer learning and collective reasoning.

- Strengthens communication and team-based problem-solving.
- Mimics interview-style pair programming and hackathon dynamics.
- Reduces plagiarism through transparent contribution tracking.
- Enhances confidence through practice in realistic coding situations.

Aligned with SDG 4 (Quality Education), Code Arena democratizes access to high-quality programming practice and fosters a collaborative learning culture. In parallel, it contributes to SDG 9 (Industry, Innovation & Infrastructure) by training learners in modern development workflows and preparing them for technologically advanced careers.

1.4 Challenges Faced and Motivations for System Design

Building a real-time, synchronized, and secure collaborative coding platform presented several technical challenges—many analogous to those addressed in decentralized repository systems

Key challenges include:

- Ensuring real-time synchronization across multiple users without delays.

- Managing server load caused by concurrent code execution requests.
- Handling conflicts when multiple users edit the same section of code.
- Preventing plagiarism despite increased visibility of shared work.
- Supporting multiple programming languages with secure sandboxing.
- Maintaining low latency for users with varying network speeds.
- Designing engaging problem sets suitable for both beginners and advanced users.

These challenges informed the architectural decisions made during development, emphasizing responsiveness, transparency, fairness, and scalability.

1.5 Summary and Vision

Code Arena fills a critical gap in technical education by combining collaboration, analytics, evaluation, and integrity enforcement into one unified platform. It transforms coding practice from an isolated task into a dynamic, interactive, and teamwork-driven experience. By embracing real-time collaboration, transparent contribution tracking, and industry-aligned workflows, Code Arena supports the development of competent, ethical, and industry-ready developers.

Future enhancements—including AI-driven feedback, intelligent skill analytics, plagiarism detection algorithms, and cross-institution competitive modes—position Code Arena as a forward-looking platform capable of redefining modern programming education.

1.6 Overview

To establish the context for CODEARENA, an analysis of existing competitive programming and collaborative coding platforms was conducted. These platforms serve as benchmarks, highlighting both established features and areas for improvement.

1.LeetCode: A popular platform for technical interview preparation, LeetCode offers a vast library of coding problems, individual timed contests, and discussion forums. While it excels in providing a single-player practice environment, it lacks a native, real-time feature for a small group of friends to join a private, timed challenge session simultaneously and see each other's progress live. Collaboration is primarily asynchronous, happening through post-solution discussions.

2.HackerRank: Similar to LeetCode, HackerRank is widely used for skill assessment and competitive programming. It supports public competitions and allows companies to host coding challenges. However, its primary model is geared towards individual performance in large-scale events or asynchronous practice. It does not focus on providing an intimate, real-time collaborative environment for small peer groups to practice together under timed conditions.

3.CoderPad: CoderPad is a leading tool for live technical interviews, offering a shared editor where an interviewer and a candidate can code together in real-time. Its strength is its powerful collaborative editor and code execution environment. However, it is designed as an assessment tool, not a practice platform. It lacks features like automated problem serving, timed challenges for groups, live activity tracking among peers, and post-session ranking systems.

The survey revealed a gap in the market for a platform specifically designed for synchronous, peer-to-peer learning and competition. While existing platforms are excellent for individual practice or large-scale contests, they do not adequately cater to small groups wanting to simulate a live hackathon or group study session with real-time feedback and a competitive edge.

II. PROBLEM DESCRIPTION

Real-time collaborative programming has become a critical component of modern software development, technical interviews, hackathons, and team-based engineering workflows. However, most existing learning and coding environments continue to operate as isolated systems, where learners work individually with limited collaboration, minimal visibility into teammate contributions, and no mechanisms to prevent plagiarism or coordinate real-time coding activities.

These limitations hinder skill development, reduce authenticity in assessments, and fail to prepare students for workplace expectations that demand teamwork, communication, and rapid problem solving under pressure.
There is a pressing need for an integrated platform that enables real-time collaboration, transparent contribution tracking, automated evaluation, and intelligent coordination among multiple programmers working on the same problem. Such a system must ensure fairness, originality, and productive teamwork while preventing conflicts, code overwriting, or misuse of shared resources.

2.1 Existing Problem

Most existing coding platforms or LMS-based evaluation systems operate using single-user submissions, where collaboration is either not supported or happens informally through external tools. These systems fail to address several practical issues:

- Lack of real-time teamwork support-Learners cannot actively see what others are coding, making genuine collaboration difficult.
- Absence of contribution tracking-It is impossible to determine who wrote what, enabling plagiarism and unfair evaluation.
- No conflict resolution mechanism-When multiple users edit shared code, existing platforms cannot prevent overwriting or version conflicts.
- Minimal simulation of real-world environments-Traditional platforms lack timed contests, shared debugging, or group-based problem-solving features.
- No transparency in shared tasks-Students can copy code outside the system without detection.
- Limited interaction tools-Most systems require external applications for communication and coordination.
- No dynamic evaluation or feedback-Educators and teams cannot monitor progress in real time.

Because these systems treat each programmer as an isolated user rather than a collaborative participant, they fail to support team-based learning or maintain academic integrity. This lack of transparency leads to plagiarism, inefficient teamwork, and unrealistic learning experiences that do not mirror industry practices.

2.2 Proposed Technique

To resolve the limitations of existing platforms, Code Arena introduces a two-stage collaborative coding mechanism that combines real-time contribution tracking with dynamic automated evaluation.
This ensures that the coding process is transparent, fair, and synchronized across all team members.
The approach ensures that each member's contribution is verifiable and that conflicts in collaborative environments are automatically detected and resolved. The platform also ensures that the learning process remains plagiarism-free by monitoring activity logs and enforcing visibility of all coding actions.

STAGE 1: USER COLLABORATION & CONTRIBUTION MANAGEMENT

Code Arena begins by establishing a synchronized collaborative environment in which all interacting users have individual identities, roles, and contribution timelines.

A. User Roles and Permissions

Each participant is assigned a role within the collaborative session:
Driver:
The primary coder currently contributing to the code.
Navigator:
A supporting user who reviews logic, suggests ideas, and monitors code behavior.
Observers or Team Members:
Additional members who provide guidance or help debug.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 1, December 2025
Impact Factor: 7.67

B. System Assumptions

Each collaboration session contains:

- Multiple programmers
- Shared problem statements
- A central coordinator/instructor (optional)

All users have access to:

- A real-time shared code editor
- Version history
- Communication modules (chat or voice)
- Automated test validation

The system can record:

- Keystrokes
- Code changes
- Execution attempts
- Debugging steps
- Idle time and conflict points

C. Contribution Logging and Conflict Handling

a) Real-Time Activity Capture

As soon as a user starts typing, Code Arena logs:

- User Identity
- Timestamp
- Code location
- Version snapshot

This prevents copying and verifies originality.

b) Conflict Detection

If two users attempt to modify the same function or code block simultaneously:

- The system alerts both participants
- Suggests merge or lock mechanisms
- Assigns temporary ownership of the code block
- Updates version history with minimal disruption

This structured conflict management ensures that collaboration proceeds smoothly without overwriting or code loss.

STAGE 2: AUTOMATED EVALUATION & REAL-TIME FEEDBACK

Once collaboration begins and contributions are logged, Code Arena transitions into dynamic assessment and feedback generation.

The evaluation engine considers:

- Live code execution
- Test case results
- Time-based performance
- Individual contribution weightage
- Code quality indicators

When a user submits code for execution, the system automatically:

- Compiles and runs the program
- Tests it against predefined test cases

- Generates real-time feedback
- Updates leaderboard ranking
- Records execution logs

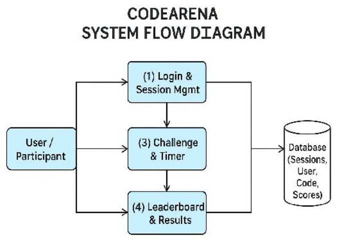
Handling Worst-Case Scenarios
During high-paced collaborative sessions, multiple worst-case scenarios can occur:

- Two or more users attempting major conflicting edits
- Code being overwritten accidentally
- Infinite loops or crashes affecting all participants
- Simultaneous requests for driver control
- Miscommunication due to rapid changes

To resolve this, Code Arena:
Prioritizes actions based on contributor activity
Users with more active engagement may receive temporary edit-lock privileges.
Implements temporary code-block ownership
Prevents overwriting and ensures safe merging.
Automatically freezes unstable code states
Protects the shared environment from being disrupted.
Issues real-time warnings and suggestions
Prevents deadlocks before they escalate.
Releases control once conflicts are resolved
Ensures smooth continuation of the collaborative task.
This ensures safe, efficient, and seamless team programming without interruptions or conflict-induced delays.

III. METHODOLOGY AND SYSTEM ARCHITECTURE

The objective of Code Arena is to create a real-time collaborative coding platform that enhances learning efficiency, prevents plagiarism, and simulates industry-like coding environments. Users log into the system to participate in collaborative sessions, coding challenges, or team-based problem-solving activities. Once a session is created, the Code Arena coordination engine synchronizes all participants, assigns roles, and ensures transparent contribution tracking. Users write, edit, and debug code in a shared workspace, while the platform continuously evaluates submissions using automated test cases.
All session data—including user edits, timestamps, code snapshots, and execution logs—is stored in a database for future analysis, performance evaluation, and academic integrity verification. The system integrates real-time synchronization protocols, live messaging, activity monitoring, and automated grading pipelines. Live notifications inform users of merge conflicts, execution results, errors, and updates to the collaborative workspace.
The following figure represents the system flow diagram of our project.



1. Technical Implementation and Deployment

The Code Arena platform is built using a multi-layered hybrid architecture optimized for real-time editing, low-latency operations, and secure collaborative environments. In contrast to traditional coding systems that focus on individual submissions, Code Arena integrates synchronized editing, automated evaluation, and behavioral tracking to create a holistic and interactive learning experience.

HARDWARE REQUIREMENTS

Backend Services:

1. Cloud-based hosting for a lightweight Node.js/Socket.IO server (for editor synchronization).
2. Supabase cloud services for the database, authentication, and real-time backend.
3. Client: A personal computer or laptop with a modern web browser (e.g., Chrome, Firefox) and an internet connection.

SOFTWARE REQUIREMENTS

Frontend: React (built with Vite)

1. UI/Styling: Tailwind CSS
2. Code Editor: Monaco Editor
3. Database :MySQL
4. Backend-as-a-Service (BaaS): Supabase (Used for its PostgreSQL database).
5. Real-Time Editor Sync: Socket.IO (Client & Server)
6. Code Execution Engine: Judge0 API
7. Build Tool: Vite

2. SYSTEM ARCHITECTURE

The architecture of CODEARENA is a modern, decoupled client-server model. It separates concerns by using dedicated services for the frontend, real-time editor sync, state management, and code execution. This "serverless" approach (using Supabase) reduces backend complexity.

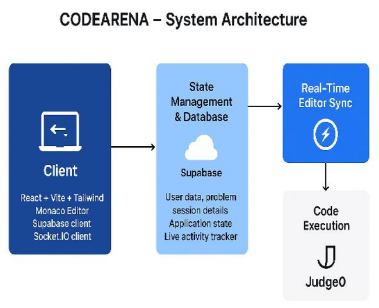


Figure System Architecture of CODEARENA

Description: The application flow is managed by three independent communication channels originating from the client:

1. Client (React + Vite + Tailwind): The user interacts with the application through a fast, responsive frontend built with React and Vite. The UI is styled with Tailwind CSS. The client integrates the Monaco Editor component, the Supabase client library, and the Socket.IO client.
2. State Management & Database (Supabase): The Supabase client connects directly to the Supabase backend. This channel is used for all primary data and application state:

i. Storing and retrieving user data, problem statements, and session details.

ii .Managing application state (like timers).

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 1, December 2025
Impact Factor: 7.67

iii .Broadcasting live activity tracker updates (e.g., "completed", "passed 2/5

3. Real-Time Editor Sync (Socket.IO): For the high-frequency, low-latency updates required by a collaborative editor (e.g., live typing, cursor movements), the client opens a separate connection to a lightweight, dedicated Socket.IO server. This server's only job is to receive editor changes from one client and instantly broadcast them to all other clients in the same coding room.

4. Code Execution (Judge0 API): When a user clicks "Run" or "Submit," the client makes a direct, asynchronous API call to the external Judge0 API. It sends the source code and test cases. Judge0 executes the code in a secure sandbox and returns the results (e.t., Pass, Fail, Runtime Error). The client then displays this result to the user and updates the user status . This decoupled architecture makes the system scalable, as each component (editor sync, database, code execution) can be managed and scaled independently.

5. Collaboration Request Processing and Execution Pipeline
Every collaborative coding session in Code Arena undergoes a multi-stage execution pipeline designed to coordinate users, maintain fairness, and optimize team performance. The pipeline integrates real-time synchronization with automated evaluation to create an accurate and engaging coding environment.

1) Session Creation and Metadata Capture

When a user initiates a coding challenge or joins a collaborative room, the system captures:

- User identity
- Role (Driver, Navigator, Observer)
- Problem ID and difficulty
- Session timestamp
- Active collaborators
- Permissions allowed

Metadata is structured and stored for analysis, assessment, and replay.

2) Structured Metadata Storage

All session metadata is indexed within MongoDB based on:

- User ID
- Session ID
- Edit timestamps
- Contribution quantity and quality
- Test-case pass/fail logs

This ensures fast, filtered retrieval during performance evaluation or academic auditing.

3) User Synchronization & Contribution Tracking

The backend engine assigns real-time synchronization channels to each participant. Every code edit or cursor movement is:

- Captured as a micro-event
- Mapped to a unique user
- Time-stamped
- Added to the session event log

This provides complete transparency and prevents plagiarism or hidden contributions.

4) Dynamic Execution and Automated Testing

A dedicated execution module compiles and runs the code inside a secure sandbox. The system:

- Evaluates logic using predefined test cases

- Measures time and memory usage
- Returns immediate feedback to all users
- Stores execution results for analysis

If a user introduces errors or crashes, the system logs the state and notifies the team.

4. Contribution

Research Contribution

Code Arena provides several innovations that expand the capabilities of traditional coding environments and contribute to modern educational technology.

1) Integrated Real-Time Collaboration Framework

The platform unifies synchronized editing, communication, and automated evaluation into a single workspace—something rarely achieved by conventional learning tools.

2) Intelligent Contribution Tracking

Fine-grained logging ensures academic integrity by identifying the exact contribution of each participant.

3) Automated Evaluation and Feedback Loop

The system supplies instant, continuous feedback that accelerates learning and reduces debugging time.

4) Conflict-Aware Multi-User Coding Engine

The OT/CRDT-based synchronization engine is specifically optimized for educational multi-user scenarios, reducing conflict and ensuring stability.

5) Scalable Architecture for Academic and Industrial Use

The platform can support:

- Pair programming
- Group assignments
- Hackathons
- Interview coding rounds
- Classroom collaborative tasks

6) Foundation for Future AI-Based Enhancements

The system's architecture supports future extensions such as:

- AI tutoring
- Code similarity detection
- Skill analytics
- Intelligent task recommendations

Code Arena thus serves as a comprehensive solution to enhance teamwork, transparency, integrity, and learning outcomes .

5. Security, Safety, and Data Handling

Just like EVDS required rigorous safety, Code Arena requires strict integrity, privacy, and reliability mechanisms. Because the platform records fine-grained user activity, it must secure all internal processes.

1) Authentication and Access Control

- Mandatory login using secure credentials

- Role-based access for drivers, observers, instructors, and administrators
- Real-time authorization for high-impact actions (session reset, lock override)

2) Secure Data Handling

- Encryption of all user activity logs and code snapshots
- Local processing of editor events for faster decision-making
- Integrity checks for all session events to prevent corruption
- User-controlled deletion of session data when allowed

IV. LITERATURE REVIEW

[1] H. Li and E. Torres (2022), in "Collaborative Learning Environment for Programming Education" (ACM Transactions on Computing Education), highlighted that team-based learning enhances problem-solving and peer interaction. Their findings support CodeArena's objective of fostering teamwork, real-time feedback, and collaborative coding in a shared online space.

[2] A. Rahman and L. Chen (2022), in "Web-Based IDEs for Collaborative Programming" (International Journal of Computer Applications), proposed a web-based IDE using Web Sockets for real-time interaction. This serves as a technical foundation for Code Arena's live code synchronization and communication using Socket.IO.

[3] M. Patel and J. Kumar (2023), in "Gamification in Coding Education" (Education and Information Technologies, Springer), demonstrated that gamification—through leaderboards, timers, and rewards—boosts engagement and motivation. Code Arena integrates these concepts to make collaborative coding more interactive and competitive.

[4] S. Gupta and R. Singh (2023), in "Real-Time Collaborative Coding Platforms: Enhancing Peer Learning" (IEEE Access), showed that synchronous code editing improves understanding and teamwork. This directly aligns with Code Arena's goal of enabling users to code, debug, and learn together in real time.

V. ACKNOWLEDGMENT

We would like to express our sincere gratitude to our guide Mrs. Akshatha Preeth P, Assistant Professor, Department of Information Science & Engineering, GAT For their valuable guidance support an encouragement throughout the completion of this project.

REFERENCES

1] "Monaco Editor Documentation," Microsoft. [Online]. Available: https://microsoft.github.io/monaco-editor

[2] "W3Schools Tutorials – Easy Web Development References," W3Schools. [Online]. Available: https://www.w3schools.com

[3] "Real-Time Collaborative Systems," IEEE Xplore. [Online]. Available: https://ieeexplore.ieee.org

[4] "Collaborative Coding in Education," ResearchGate. [Online]. Available: https://www.researchgate.net

[5] S. Gupta and R. Singh, "Real-Time Collaborative Coding Platforms: Enhancing Peer Learning," IEEE Access, vol. 11, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10234156

[6] H. Li and E. Torres, "Collaborative Learning Environments for Programming Education," ACM Trans. Computer. Educ., 2022. [Online]. Available: https://dl.acm.org/journal/tecs