IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.141276

Real Time Code Collaborator: A Cloud-Based Platform for Seamless Multi-User Programming

Ms. Shruthi T S[1], Sagar M[2], Sourav G[3], Srujan G[4], Yashaswini S L[5]

Assistant Professor, CSE, KSIT, Bangalore, India[1]

Student, CSE, KSIT, Bangalore, India[2]

Student, CSE, KSIT, Bangalore, India[3]

Student, CSE, KSIT, Bangalore, India[4]

Student, CSE, KSIT, Bangalore, India[5]

Abstract: Coding interviews, hackathons held over the inter net, and international software collaborations all necessitate plat forms that provide real-time, interactive coding environments. But most of the available tool lack in providing smooth safe, and multi-language support for effective collaboration. In this paper, we introduce the Real-Time Code Collaborator (RTCC), a browser-based, full-stack system that allows multiple users to code, execute, and debug programs in real time collaboratively. RTCC integrates the strength of Web Sockets for live synchrony, Docker containers for secure running, and a React/Next.js frontend with voice and chat support. The platform not only increases productivity for remote teams but also enables greater accessibility in technical education and hiring. We cover the design principles, technologies, and architecture employed to build RTCC and contrast it with conventional tools such as Google Collab and VS Code Live Share.

Keywords: Real-time coding, Cloud IDE, Web Sockets, Collaborative development, Code execution, Multi Language support, OAuth 2.0, Git integration

I. INTRODUCTION

In today's interconnected world, software programming is no longer limited to single-device systems or local teams. Modern developers often collaborate with people across different cities, time zones, and countries. This shift has created a strong need for real-time collaborative coding plat forms—tools that allow programmers to work together as if they are in the same room, even when separated by distance. However, most existing platforms provide only partial solutions. Tools like Google Collab support collaboration but restrict users to a single language. Editors like VS Code with Live Share offer strong real-time editing but require installation, setup, and a powerful device. Due to these limitations Identify applicable funding agency here. teams often need multiple tools just to achieve smooth collaboration, which reduces efficiency and user experience. The Real-Time Code Collaborator (RTCC) was developed to address these challenges by offering an integrated, browser-based coding interface. RTCC enables users to write, edit, and execute code collaboratively in real time. It supports multiple programming languages, provides live cursor tracking, real time updates, and includes a built-in text chat system for communication inside the editor. These features help remove common barriers in remote coding environments. Technically, RTCC is built using a modern web stack consisting of React.js, Node.js, WebSockets, and Firebase (if applicable in your project). The architecture allows users to collaborate directly in their browser without installing any software or configuring their local environment. The backend supports code execution for languages such as C, C++, Java, Python, and JavaScript, giving users a powerful and flexible environment for learning, practicing, and teamwork. Ultimately, RTCC aims to replicate the feel of a real, in person coding session—allowing multiple users to work on the same codebase simultaneously, discuss ideas through chat, and see changes instantly.

II. SYSTEM ARCHITECTURE

RTCC follows a modular, layered architecture designed to provide seamless real-time code collaboration, secure edu cation, and persistent data storage. The system is divided into four primary components: Frontend, Backend, Execution Layer, and Database. Each layer is optimized to handle specific responsibilities while ensuring smooth integration with other layers.

A. ***Frontend:*** The frontend of RTCC is responsible for providing an interactive and responsive user interface where users can write, run, and collaborate on code in real time. The main features and technologies used include:

• React.js:
Used to build a modern, component-based interface. It allows fast UI rendering, smooth user interactions, and an efficient structure for managing editor, chat, and room-based collaboration features.

• Monaco Editor:
Powers the in-browser code editor. It provides essential developer features such as: syntax highlighting, auto completion, error detection support for multiple programming languages,This allows users to write code comfortably within a professional editor environment.

• WebSockets:
Handles real-time, bidirectional communication between all connected users. WebSockets enable: live code synchronization, real-time cursor tracking, instant message updates in the chat consistent multi-user collaboration This ensures that all participants see updates immediately, similar to real-time pair programming.

• Room-Based Collaboration: Users can create or join rooms to collaborate on a shared code file. Each room maintains its own code state, active users list, and chat messages.

B. ***Backend:***
The backend layer forms the communication backbone of the Real-Time Code Collaborator (RTCC), enabling live collaboration and multi-language code execution. It is implemented using Node.js and Express.js, which provide a lightweight and scalable server environment for handling HTTP requests, routing, and API management. The backend also manages room creation, code execution workflows, and overall session coordination.

A key component of this layer is WebSocket management, which enables persistent, bidirectional communication between clients and the server. WebSockets allow the system to broadcast real-time updates such as code edits, cursor movements, and chat messages to all connected users with minimal latency. This ensures a seamless collaborative experience similar to in-person pair programming.

C. **Execution Layer:**
The execution layer ensures efficient and isolated execution of user-submitted code across multiple programming languages. This layer operates closely with the backend to manage compilation, execution, and error reporting. It provides multi-language support, enabling users to run programs in Python, JavaScript, Java, C, and C++. Each execution request is processed independently, preventing interference between users working in different rooms or running multiple code snippets concurrently. The layer also features robust error handling and output streaming, capturing syntax errors, runtime exceptions, com piler messages, and standard output. This real-time feedback allows users to debug and test their code instantly. Temporary file isolation ensures that each execution is contained within its own directory, improving both reliability and safety.

D. **Database Layer:**
The database layer is responsible for storing and synchronizing key application data required for real-time collaboration. RTCC utilizes Firestore and Supabase depending on deployment needs.Firestore, a real-time NoSQL database, is used for:

- maintaining active room states
- synchronizing chat messages
- tracking user presence
- optionally backing up editor content

Its low-latency synchronization model ensures that collaborative data is updated instantly across all connected clients. Supabase serves as an optional structured storage layer for:

- saving user code snippets
- managing room metadata
- storing project-specific files

Supabase provides PostgreSQL-backed storage with fast querying and secure data handling, making it suitable for persistent project data. Lightweight user data and session information may also be stored in these services to support room participation and activity tracking. Full authentication and version control features are not yet implemented but can be integrated in future enhancements using Firebase Auth or Supabase Auth.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁐ Peer-reviewed & Refereed journal ⁐ Vol. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.141272

III. METHODOLOGY

The Real-Time Code Collaborator (RTCC) is a cloud-based collaborative coding platform that enables multiple users to write, edit, and run code together in real time. It offers a simple, browser-based interface integrated with real-time editing, live chat, and multi-language code execution using a modern full-stack architecture.

- ***System Flow:***
  When a user types code, the changes are synchronized instantly for all participants through WebSockets. Users in the same room can see each other's code updates, cursor movements, and chat messages without delay. Code execution requests are sent to the backend, where the program is com piled or interpreted, and the output is returned and displayed to all users in the session.

- ***Frontend:***
  The frontend is built using React.js, providing a fast, component-based UI. It integrates the Monaco Editor to deliver features such as syntax highlighting, autocompletion, multilanguage support, and error detection. WebSockets handle live collaborative editing, cursor synchronization, and messaging. The interface is fully browser-based, requiring no installation or setup.

- ***Backend:***
  The backend is implemented using Node.js and Express.js, which manage API endpoints, WebSocket connections, and room coordination. A custom code execution handler com piles and runs programs using Node's child process module, supporting languages such as C, C++, Java, Python, and JavaScript. Each execution uses a temporary directory to maintain safe and isolated processing.

- ***Real-Time Collaboration:***
  WebSockets provide continuous, low-latency connections between users and the server. This enables real-time updates for:
  - code changes
  - cursor movements
  - active user presence
  - chat messages

- **Code Execution:**

The backend receives code, creates a temporary file, runs it through the appropriate compiler or interpreter, and returns the output or error logs. This ensures users receive instant feedback during collaborative sessions.

➢ ***System Workflow Summary***

1) **Room Join:**
A user enters a room by entering a room ID or creating a new one.

1) Editor Load:
The frontend loads the Monaco editor and establishes a WebSocket connection with the server.

2) Real-Time Editing:
Code updates, cursor movements, and chat messages are exchanged through WebSockets for instant collaboration.

3) Code Execution:
User submits code → backend API receives the program → backend compiles or runs the code → output/errors are returned.

4) Live Sync:
The execution result is displayed to all users in the room in real time.

5) Session Exit:
When a user leaves, the room updates the active users list. (Optional future work: store code to database.) illustrates the overall methodology of the Real-Time Code Collaborator (RTCC). The top section represents the user workflow, beginning from user login and room selection, followed by real-time collaborative editing, code compilation, and execution, and finally session exit. The lower section shows the system architecture that supports this workflow. The frontend, built with React.js and Monaco Editor, captures user inputs and editor events, which are synchronized through Fig. 1. System Methodology Flow of RTCC WebSockets. The backend, implemented using Node.js and Express.js, processes code execution requests and manages real-time communication. Firestore or Supabase serves as the cloud storage layer for maintaining room data, messages, and optional code backups. Together, these components enable seamless, low-latency, browser-based collaborative coding.

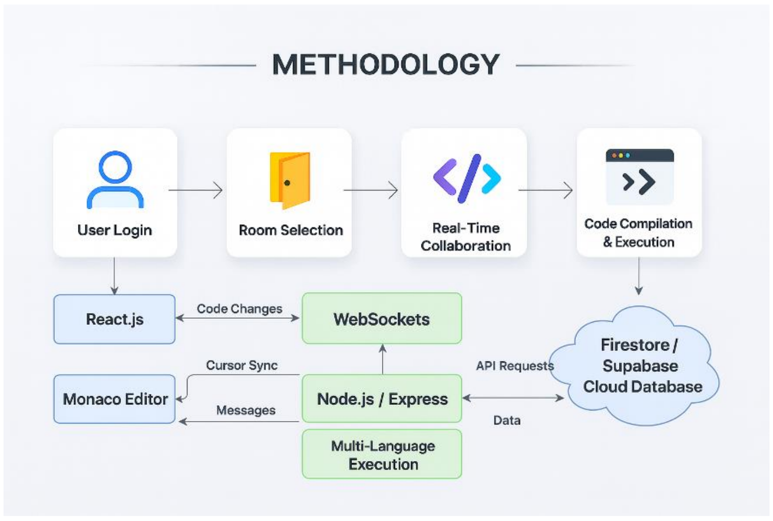

Fig 1: System Methodology Flow of RTCC

IV. IMPLEMENTATION

The Real-Time Code Collaborator (RTCC) is designed to deliver a smooth, interactive, and fully browser-based collab orative coding experience. The workflow ensures that multiple users can edit code simultaneously, execute programs, and exchange messages in real time. The implemented workflow consists of the following stages:

A. *Room Creation and User Entry*

- Users begin by either creating a new room or joining an existing one using a room ID.
- No external authentication is required; users can enter directly with a chosen display name.
- Once inside the room, the system initializes all collabo rative features for that session

B. *Editor Initialization and WebSocket Connection*

After entering a room, the Monaco Editor is loaded on the frontend, providing syntax highlighting, autocompletion, and multi-language support. A persistent WebSocket connection is established between the client and the Node.js backend. This connection enables:

- real-time code synchronization
- live cursor updates
- instant chat message delivery
- broadcast of execution results

Any edit made by one participant is immediately reflected for all others, ensuring a synchronized collaborative environment

C. Multi-Language Code Editing and Execution

RTCC supports multiple programming languages including Python, JavaScript, Java, C, and C++.Users can switch languages from the editor without refreshing the page. When the user runs code:

- The code is sent to the backend using an API request.
- The backend creates a temporary file.
- The program is compiled or executed using the appropriate compiler/interpreter via Node.js child process.
- The resulting output or error logs are returned to all users in the room.

This enables real-time collaborative debugging and program testing.

D. Real-Time Text Chat

- A built-in text chat system enables users to exchange messages instantly while coding.
- Chat messages use the same WebSocket channel as code updates, ensuring minimal delay.

E. Database Synchronization (Firestore / Supabase)

Depending on deployment, RTCC integrates Firestore or Supabase for storing essential session data such as:

- chat history
- room metadata

- optional backup of code snippets The database helps preserve consistency even if users re fresh or rejoin the session.

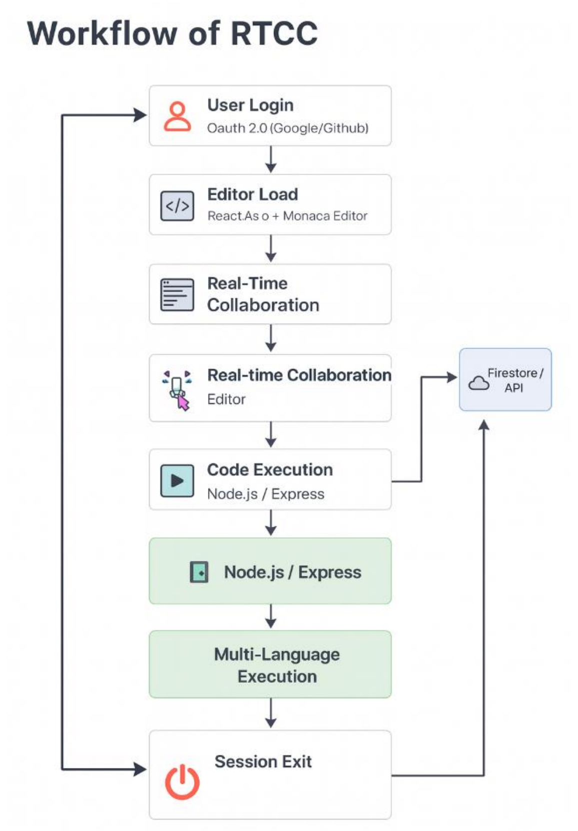

Fig 2: Workflow of RTCC illustrating room entry, editor initialization, real time collaboration via Web Sockets, multi-language execution, and synchronized output display.

F. Security and Communication Handling

- All HTTP requests are transmitted through HTTPS, and all WebSocket communication takes place over WSS, ensuring encrypted data transfer.
- Temporary code execution files are isolated to prevent conflicts between different user sessions.
- No sensitive authentication credentials are stored since the current system operates without OAuth or password based login.

V. APPLICATIONS

The Real-Time Code Collaborator (RTCC) is a versatile platform with numerous practical applications across education, software development, and distributed teamwork. Its real time collaboration features, browser-based accessibility, and multi-language execution make it an effective tool in various domains.

A. *Mock Coding Interviews and Recruitment Assessments*

- RTCC can be used by companies, training institutes, and educational organizations to conduct mock coding interviews in an online environment.
- Interviewers can share a coding problem, observe real time code edits, and evaluate a candidate’s thought pro cess through the integrated text chat.

B. *Hackathons and Developer Events*

- RTCC is suitable for online hackathons and coding com petitions where participants must collaborate efficiently under time constraints.
- Teams can work together on the same code file, share ideas through chat, and see updates instantly, avoiding issues related to local environments or version conflicts.

C. *Remote Pair Programming and Distributed Team Collaboration*

- Software development teams operating remotely can use RTCC for pair programming sessions, debugging, and collaborative problem solving.
- •Developers can work on the same piece of code simultaneously, observe each other's changes, and communicate through the built-in chat system.

VI. CONCLUSION AND FUTURE WORK

A. *Conclusion*

The Real-Time Code Collaborator (RTCC) successfully provides a fully browser-based environment for multi-user collaborative programming. By integrating React.js, Monaco Editor, WebSockets, and a Node.js backend with multi language execution, the platform eliminates the need for local development setup and enables seamless real-time coding interactions. Users can write, edit, and execute code together in multiple languages while communicating through an integrated chat system.

The system's architecture ensures responsiveness, scalability, and ease of use, making it suitable for educational institutions, online coding practice, distributed development teams, and collaborative debugging scenarios. RTCC demonstrates that real-time programming collaboration can be achieved with lightweight, modern web technologies without relying on heavy installations or dedicated IDEs.

B. *Future Work*

Although RTCC provides a strong foundation for real time collaborative coding, several enhancements can further improve its performance, usability, and security:

- Voice/Video Communication Integration
- Containerized Secure Execution
- AI Assistance and Code Suggestions
- Multi-File Workspace Support
- Classroom and Admin Dashboard

REFERENCES

[1]. A. Mesbah and A. van Deursen, "An Architectural Style for Ajax Web Applications," IEEE Software, vol. 29, no. 6, pp. 30–37, 2012.

[2]. G. C. Hunt and M. L. Scott, "The Coign Automatic Distributed Partitioning System," Operating Systems Review, vol. 33, no. 5, pp. 187–200, 1999.

[3]. M. Greiler, J. van der Hoek, and A. van Deursen, "Code Synchronization in Collaborative Software Development," in Proc. ACM/IEEE International Conference on Software Engineering, 2012, pp. 578–588.

[4]. Monaco Editor Documentation, Microsoft. Accessed: 2025. [Online]. Available: https://microsoft.github.io/monaco-editor

[5]. Supabase Documentation: Realtime Database Cloud Storage. Accessed: 2025. [Online]. Available: https://supabase.com/docs

[6]. S. Luo, X. Zhu, and Q. Zhou, "Operational Transformation in Real-Time Collaborative Editing," International Journal of Software Engineering and Applications, vol. 10, no. 3, pp. 45–56, 2022.

[7]. K. Thirunavukarasu et al., "Real-Time Programming Education Tools Using Web Technologies," International Journal of Advanced Trends in Computer Science and Engineering, vol. 10, no. 6, pp. 58–63, 2021