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 6, June 2025

DOI: 10.17148/IJARCCE.2025.14629

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

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

Assistant Professor, DEPT of CSE, KSIT, Karnataka, India[1]

Student, DEPT of CSE, KSIT, Karnataka, India[2]

Student, DEPT of CSE, KSIT, Karnataka, India3

Student, DEPT of CSE, KSIT, Karnataka, India[4]

Student, DEPT of CSE, KSIT, Karnataka, India[5]

Abstract: Coding interviews, hackathons held over the internet, and international software collaborations all necessitate platforms 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, Docker, Web Sockets, Collaborative development, Code execution, Multi-anguage support, OAuth 2.0, Git integration

I. INTRODUCTION

In the modern, more interconnected world, software programming is no longer confined to one-device systems or local teams. Today's programmer typically works with people from various cities, zones, and even continents. This has brought forth a pressing demand for platforms that facilitate real-time, hassle-free collaboration—especially in software development, where code sharing and team programming are common. Programmers now demand the capability to cooperate as if they are sitting side by side, irrespective of physical distance.

Most solutions available offer incomplete support to such requirements. Platforms such as Google Collab or VS Code with Live Share offer partial real-time editing capabilities but usually do not integrate fully. These solutions might limit language choices, have local setup requirements, or be missing voice communication and secure execution environments. Due to this, teams have to balance several tools and services in order to get simple collaboration, which slows down productivity and user experience. The Real-Time Code Collaborator (RTCC) was created to solve these issues by providing an integrated, browser-based coding interface.

RTCC enables users to write, edit, and execute code collaboratively in real time. The software accommodates several programming languages and offers a completely interactive experience by means of live cursor tracking, real-time updates, and communication features within the editor such as chat and voice. These enable breaking down of the barriers that typically hinder seamless collaboration in conventional settings. One of RTCC's strengths is its technical backbone. The system is constructed on a contemporary, cloud-native stack with technologies such as React.js, Next.js, Node.js, Web Sockets, Docker, Firebase, and OAuth 2.0.

Each of these technologies was selected to optimize performance, scalability, and security with the guarantee that users be able to use the platform without installations or intricate configurations. This architecture enables RTCC to provide a uniform and responsive experience on devices. Finally, RTCC seeks to emulate the dynamics of a physical coding session in virtual form.

II. LITERATURE SURVEY

Over the last ten years, real-time coding tools have become more important because of remote work, online learning, and global teams. Traditional IDEs like VS Code or JetBrains support collaboration, but they require local setup and powerful devices. Tools like Live Share in VS Code help with collaboration, but they aren't browser-based and need installations.

Google Colab allows real-time Python coding but only supports one language. It also lacks features like voice chat, version control, or secure execution through Docker. Replit supports many languages in the browser, but it doesn’t provide strong container security, voice chat, or advanced versioning needed for professional use.

The Real-Time Code Collaborator (RTCC) addresses these issues. It supports multiple languages and ensures secure code execution using Docker. It also offers real-time syncing, built-in voice and text chat, all within a browser. RTCC uses OAuth and JWT for secure login, along with tools like Firebase and PostgreSQL for syncing and data storage. RTCC brings together the best features into one simple, powerful platform for students, developers, and interview practice.

III. OBJECTIVES

The Real-Time Code Collaborator (RTCC) project will revolutionize collaborative software development via a cloud-based, multi-user, real-time coding platform. The system is meant to address the changing needs of developers, educators, and interviewers who need secure, scalable, and interactive programming environments. The following objectives of RTCC include:

1) ***Improve Collaborative Coding Experience***: Design an engaging and effective environment for real-time multi-user coding. RTCC enables users to collaborate effortlessly from anywhere, allowing for synchronous code writing, editing, and debugging to improve productivity in distributed teams.

2) ***Facilitate Multi-Language Execution with Security***: Enable secure and isolated execution environments for code using Docker containers that allow different programming languages such as Python, Java, and C++. This allows users to test and execute code in a secure and consistent environment without posing threats of system-level disruption or cross-user effects.

3) ***Incorporate Real-Time Communication***: Incorporated voice and chat capabilities within the platform through WebRTC and WebSockets. This allows users to communicate and resolve issues together in the same environment, eliminating the third-party communications tools.

4) ***Scale Cloud-Based Infrastructure:*** Use cloud platforms (AWS, Firebase, Google Cloud) to deploy and scale the application based on user traffic. This will keep RTCC responsive and operational for both small coding groups and large classrooms or hackathons.

5) ***Provide Secure and Easy User Access:*** Use OAuth 2.0 for easy and secure authentication through Google or GitHub, and secure sessions with JWT. These features streamline user login while providing strong data protection.

IV. METHODOLOGY

The Real-Time Code Collaborator (RTCC) is a cloud-based platform that lets users code together in real time. It features a user-friendly interface, secure code execution, live chat, and cloud scalability using modern full-stack and cloud technologies.

- ***System Flow***: When a user types code, it syncs instantly with others through WebSockets. Code runs in a secure Docker container, and the output is shared live. Users can also talk or chat during sessions.

- ***Frontend***: Built with React and Next.js, the interface includes a smart code editor (Monaco or Code Mirror) and video/audio chat using WebRTC. WebSockets manage live updates and interactions.

- ***Backend***: It uses Node.js and Express.js for APIs, WebSockets, and Docker control. Each code run starts a secure container, which shuts down after use to ensure safety.

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 6, June 2025
DOI: 10.17148/IJARCCE.2025.14629

- ***Authentication***: Users can log in through Google or GitHub using OAuth 2.0. JWT tokens handle user access and roles, such as admin or guest, which determine who can edit or run code.

- ***Code Execution***: Code runs in isolated Docker containers with limits on CPU, memory, and time. This keeps the system safe from harmful scripts.

- System Workflow Summary:

1. User Login: Authentication through OAuth 2.0 with Google/GitHub.
2. Editor Load: Frontend loads and connects to WebSocket server.
3. Real-Time Editing: Code updates and cursor locations are communicated through Socket.io.
4. Voice Session (Optional): WebRTC channel provides peer-to-peer voice session.
5. Code Execution: Code executed on backend API → forwarded to Docker container → run → result returned.
6. Sync & Display: Result shown on all logged-in users' screens.
7. Version Control: User can commit code through Git integration.
8. Logout/Exit: Session expires; changes saved to cloud database.

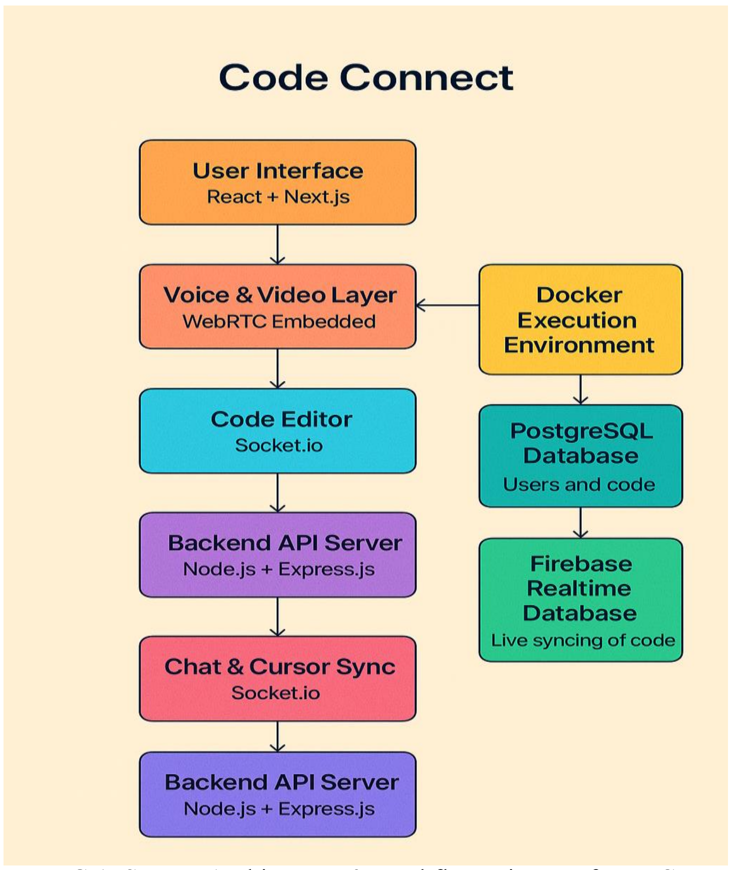

FIG 1: System Architecture & Workflow Diagram for RTC

V. KEY FEATURES

The Real-Time Code Collaborator platform provides several strong features that create a unified environment for collaborative coding. It is designed for students, interview training, coding bootcamps, and virtual development teams. Below, we highlight the key features that define its capabilities.

- ***Multi-Language Support with Compilation***: The platform supports various programming languages, including Python, C++, Java, and JavaScript. Users can choose the syntax they are using and compile their code in real time. The backend offers secure sandboxed environments to run user code, with output streamed back to all collaborators instantly.

- ***Role Assignment Interview Mode***: One notable feature is the "Interview Mode," where users can assign roles like interviewer and interviewee. In this mode, the interviewer can ask coding or system design questions, see how the candidate approaches them, and give feedback. This makes the platform perfect for mock interview practice.

- ***Smart Hints and Real-Time Feedback***: With GPT integration, the system can look at the code being written and offer intelligent, context-sensitive hints or suggestions. For example, if a user gets stuck in a loop or uses an inefficient sorting algorithm, the system might recommend alternatives like, "Use a hash map here to reduce time complexity." This feedback is optional and can be turned on or off. It is especially helpful for beginners needing guidance while learning problem-solving strategies during collaborative sessions.

- ***Session Recording and Replay***: Each session can be recorded and replayed, capturing both the code development process and the audio/video conversation, with user consent. This is helpful for reviewing what went well or poorly in a coding interview or group debugging session.

VI. APPLICATIONS

Interview Prep Platforms: These platforms are great for mock coding interviews on sites like LeetCode or Hacker Rank. They provide real-time code editing, video chat, and interview modes. Users can practice and receive feedback.

- ***Online Coding Bootcamps and EdTech***: Bootcamps and online courses can use these tools for live group coding, pair programming, and mentor sessions. Teachers can monitor and guide students in real time.

- ***Remote Software Development Teams***: This tool is useful for remote teams to code together live, chat, compile, and comment, all in one place. This is different from standard video meeting tools.

- ***Classroom and University Labs***: This helps students work together on assignments in real time. Teachers can track progress, give feedback, and use session data for grading.

- ***Career Mentorship and Coaching***: Mentors can guide junior developers live by watching them code, giving tips, and debugging together using voice and code sharing.

- ***Hackathons and Developer Events***: These tools are perfect for online hackathons. Teams can code, chat, and collaborate live without needing to set up local tools or share screens.

- ***Code Tutoring and Freelance Platforms***: Tutors and freelancers can utilize it for live lessons, debugging help, or project work. It's more effective than chat, offering real-time code, voice, and feedback.

VII. TECHNICAL REQUIREMENTS

To install and run the Real-Time Code Collaborator, developers need a system with at least an Intel i5 or Ryzen 5 processor and 8 to 16 GB of RAM. About 10 GB of storage is sufficient, unless you plan to keep session logs or history for a long time.
.

- ***Hardware Requirements***

The Real-Time Code Collaborator works well on most modern devices. Developers who run or build the system need a computer with at least an Intel i5 or AMD Ryzen 5 processor and 8 GB of RAM; 16 GB is better for smoother multitasking. It requires around 10 GB of storage unless you are storing logs or media. A GPU isn't necessary but could be useful for future features like screen sharing. A stable internet connection is vital for real-time collaboration. End-users don't need high-end devices; any modern laptop, tablet, or phone with a web browser will work, though using a keyboard and mouse is best for coding.

- ***Software Requirements***

The system uses modern, open-source software tools. The interface is built with HTML, CSS, and JavaScript. It employs the Monaco Editor for coding, and frameworks like Tailwind CSS or Bootstrap support a responsive design. Real-time editing is made possible with WebSockets and the Socket.IO library. The backend operates on Node.js (version 18 or newer) with Express.js for APIs and routing. Secure code execution is managed by APIs such as Judge0 or through Docker-based sandboxes. MongoDB stores session data, code history, and user logs, but Firebase or Firestore can also be used. For deployment, platforms like AWS, Google Cloud, Heroku, or Vercel are suitable. Docker is used to containerize services, and SSL ensures secure HTTPS or WSS connections. Optional file storage can be added using AWS S3 or Firebase Storage.

VIII. CONCLUSION

The Real-Time Code Collaborator provides a practical solution for the increasing demand for real-time, remote coding collaboration in education and software development. It combines tools like WebSockets, containerized code execution, and live databases to create a shared coding space. Users can write, view, and run code together while tracking changes and communication. This mimics in-person collaboration. Current tools do not offer this all-in-one experience, especially for learning environments. This platform addresses that need with a simple, browser-based interface that is even suitable for beginners. Initial feedback from students indicates higher engagement and quicker learning.

The system is useful not only for group work but also for studying independently with input from peers, tutors, or teachers. It is designed modularly, allowing components like the code engine, database, or interface to be updated separately. This keeps the project current with new web technologies, compilers, and AI tools. For instance, it could integrate an AI assistant like Codex or Tabnine to suggest and explain code in real time. This feature would help beginners identify and correct their mistakes. Future capabilities might also include smart commenting, voice chat, or live screen sharing during sessions.

In summary, the Real-Time Code Collaborator is a robust tool for coding together from anywhere. It supports remote learning, coding bootcamps, interviews, and team projects. With real-time feedback and access across platforms, it enhances both coding and teamwork skills. The system helps users improve through live interaction and instant code execution. As it develops, it could become a standard for learning and remote software development.

REFERENCES

[1]. GitHub, Inc. (2024). Operational Transforms and CRDTs for Real-Time Collaboration. Retrieved from https://github.com/
[2]. WebRTC Project. (2024). Real-Time Communication in Modern Browsers. Retrieved from https://webrtc.org/
[3]. Node.js Foundation. (2024). Node.js Documentation. Retrieved from https://nodejs.org/
[4]. Socket.IO. (2025). Bidirectional Real-Time Communication for Web Apps. Retrieved from https://socket.io/
[5]. MongoDB Inc. (2025). MongoDB for Real-Time Applications. Retrieved from https://www.mongodb.com/
[6]. Docker, Inc. (2024). Developing and Deploying Applications Using Docker Containers. Retrieved from https://www.docker.com/
[7]. Google Cloud Platform. (2025). Cloud Functions and Compute Engine for Serverless Backends. Retrieved from https://cloud.google.com/
[8]. Amazon Web Services. (2025). Scalable Application Deployment with AWS EC2 and Lambda. Retrieved from https://aws.amazon.com/