2025 Volume: 01 Issue: 07 www.ijrpa.com ISSN 2456-9995 Review Article

International Journal Research Publication

Page: 1-07

ONLINE REALTIME COLLABORATIVE MULTI-LANGUAGE CODE EDITOR & COMPILER

Yashmitha B*[1] , Shivabasava[2] , Nandeesh C[3] ,Vamshi krishna[4] , Jayakumar BL[5]

[1,2 ,3,4] Students, Dept of CSE, S.E.A College of engineering and Technology, India.

[5] Faculty, Dept of CSE, S.E.A College of engineering and Technology, India.

Article Received: 11 November 2025

Article Revised: 01 December 2025

Published on: 21 December 2025

***Corresponding Author: Yashmitha B**

Students, Dept of CSE, S.E.A College of engineering and Technology, India.

DOI: https://doi-doi.org/101555/ijrpa.9856

ABSTRACT:

In the age of digital transformation and cloud computing, there is a growing demand for tools that facilitate real-time collaboration and remote access. This project introduces an Online Realtime Collaborative Multi-Language Editor & Compiler designed to allow users to write, compile, and execute code in multiple programming languages directly from their web browsers. The platform utilizes Firebase for real-time collaboration and the Piston API for language execution. By offering features like live session sharing, local storage persistence, and input handling, this tool proves especially useful for students, instructors, developers, and interviewers who seek a lightweight and responsive coding environment. This project bridges the gap between traditional heavy desktop IDEs and casual coding needs in educational and professional scenarios.

INTRODUCTION

With the rapid advancement in software development and the increasing need for online education and remote work, the demand for flexible and collaborative development environments has surged. Traditional desktop-based Integrated Development Environments (IDEs) are often resource-intensive and lack native support for collaboration. Our project addresses these limitations by developing a web-based code editor that is both lightweight and powerful. It supports multiple programming languages and allows users to collaborate in real-time from any device without requiring installation or setup. The system is ideal for pair programming, remote teaching, technical interviews, and peer-to-peer learning.

Literature Review

Title	Year	Disadvantages / Limitations	Relevance to Your Project
Collaborative Code Editors - Enabling Real-Time Multi-User Coding and Knowledge Sharing *(Khushwant Virdi et al.)*	2023	- Lack of multi-language backend execution - No support for standard input (stdin) - Minimal UI customization - Lacks session persistence	Your project improves by adding Piston API integration, stdin handling, UI themes, and session saving.
Design and Development of Real-time Code Editor for Collaborative Programming *(Soumya Mazumdar et al.)*	2024	- Limited language support (mostly C, C++, Python) - Weak error handling and debugging tools - No auto-save or offline recovery features	Your project offers broader language support, local storage auto-save, and better user session management.
Collaborative Landscape of Software Development: RTC Code Editor *(Aditi Dhawan et al.)*	2024	- Uses WebRTC, which increases complexity - High latency under weak internet conditions -Limited input/output customization	Your solution with Firebase and Piston is simpler, faster, and more user-friendly under varying network conditions.
Real-Time Collaborative Coding in a Web IDE (Collabode) *(Goldman et al.)*	2011	- Supports only Java - Outdated UI/UX design - No multi-language execution - No modern web tech integration	Your system modernizes the approach with support for various languages and modern frontend frameworks.

Requirement Analysis:

1. Functional Requirements

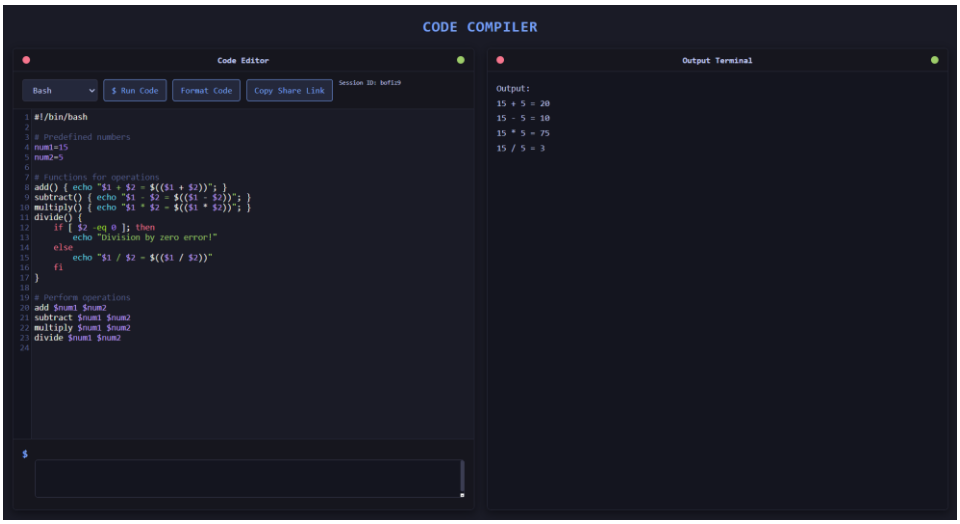
- **Multi-language Support:** The system must support over 15 programming languages including Python, Java, C++, SQL, and others.
- **Code Execution:** Users must be able to compile and run code through a backend service that returns output or error messages.
- **Real-Time Collaboration:** Multiple users should be able to work on the same code document simultaneously, with changes reflected in real time.
- **Session Management:** Users can generate a session link to invite others for collaborative coding.

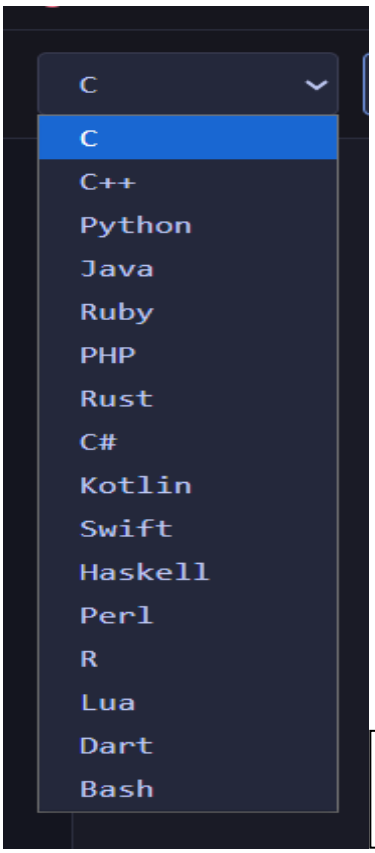
- **Input Handling:** Users can provide custom input to programs via a dedicated input box (stdin).
- **Live Output Display:** A terminal section will display real-time output and errors post execution.
- **Persistence:** Code and session metadata are temporarily saved in the browser's local storage.
- **Session Timer:** Sessions are automatically set to expire after 24 hours to ensure data refresh and integrity.

2. Non-Functional Requirements

- **Responsiveness:** The platform should be fully usable on both desktop and mobile browsers, with dynamic UI adjustment using media queries.
- **Cross-Browser Compatibility:** It must function seamlessly on modern browsers like Chrome, Firefox, Safari, and Edge.
- **Security:** Execution must occur in an isolated and secure environment to prevent unauthorized access or code injection (handled through Piston API).
- **Scalability:** The architecture should support multiple concurrent users and session loads without performance degradation.
- **Performance:** Code execution response time must be minimal, ideally under 3 seconds under normal conditions.
- **Usability:** The user interface should be intuitive for beginners while still powerful for experienced developers.
- **Availability:** The tool should be hosted in a way that provides high uptime and reliability.

Implementation (Photos)





Second figure shows the languages that the user has to select to run.

Project Design

Frontend Interface

- Built using **HTML**, **CSS**, and **JavaScript**, this part includes a clean, responsive user interface.
- The code editor is built using **CodeMirror**, which provides syntax highlighting and language support.
- Users interact with this interface to write code, provide inputs, run programs, and collaborate.

Backend Execution

- The **Piston API** acts as the backend engine, responsible for compiling and executing code in various programming languages.
- It ensures that user code runs in a sandboxed environment, separating processes securely.

Real-Time Collaboration Layer

- Real-time synchronization is achieved using **Firebase Realtime Database**.
- When one user edits the code, the changes are immediately reflected for all users in the same session.
- Firebase also stores minimal session data for identification and continuity.

Session & Storage Layer

- Browser **local storage** is used to persist user code and session IDs across page reloads.
- A session timer monitors the duration and sends alerts before expiration.

User Experience & Responsiveness

- The interface adjusts automatically for different screen sizes, providing seamless access from both mobile and desktop devices.
- Simple navigation and structured layout enhance the usability.

Testing

1. Unit Testing

Unit testing was performed on individual components such as:

- Code input and execution functionality
- Session creation and sharing
- Real-time update propagation via Firebase
- Output display for successful and erroneous programs
- Local storage persistence for session recovery
- These tests ensured that each feature worked correctly in isolation.

2. Integration Testing

Once individual components passed unit testing, integration testing was conducted to verify that:

- Code entered in the editor is successfully sent to the backend (Piston API)
- Input entered in the stdin field is correctly included in the code execution request
- Firebase correctly synchronizes code changes across all clients in the same session
- The system handles simultaneous requests from multiple users

This step confirmed that all subsystems work together as intended.

3. Cross-Browser and Responsiveness Testing

The platform was tested across different web browsers (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari) and various devices (PCs, tablets, smartphones) to verify:

- UI responsiveness and layout adaptation
- Consistent functionality of the editor and output panel
- No dependency or crash across browser types

4. Performance Testing

Performance testing was done to ensure:

- Real-time updates occur with minimal latency (typically under 500ms)
- Code execution times are within acceptable limits (1–3 seconds depending on the language)
- Firebase handles concurrent editing from multiple users without lag

Simulated network conditions were used to measure performance under high latency scenarios, verifying the platform's resilience.

5. Security Testing

Security testing focused on:

- Isolating user code to prevent harmful execution (handled through Piston API's containerization)
- Preventing unauthorized access to sessions not shared by the user
- Blocking scripts or code that could attempt browser-side manipulation

6. Usability Testing

User feedback was collected from a group of 15 users including students and instructors. They were asked to complete common tasks like:

- Creating a session
- Writing and running code in different languages
- Collaborating with a peer in real-time

CONCLUSION

The "Online Realtime Collaborative Multi-Language Editor & Compiler" successfully delivers a lightweight, cloud-based coding platform that supports real-time collaboration and multi-language execution. It addresses the growing need for flexible and efficient development environments that require no software installation. Through the integration of Firebase and Piston API, this tool enables seamless collaboration, making it particularly useful for students, educators, technical interviewers, and developers.

By combining multiple powerful features like real-time editing, multi-language support, live output, mobile responsiveness, and session persistence, the platform offers a modern solution for coding and learning. It stands out as a practical and innovative alternative to bulky IDEs, streamlining the collaborative development experience on the web

ACKNOWLEDGEMENTS

This research was supported by the S.E.A College of Engineering & Technology of Computer Sciences and Engineering Dept.

REFERENCES

1. **Madhumita S., et al.** (2021).
   "Cloud-based IDE for Real-Time Collaborative Programming." International Journal of Advanced Research in Computer Science. [DOI: 10.26483/ijarcs.v12i5.6789]
   ✓ Discusses building collaborative online coding platforms using cloud services.
2. **S. B. Patil and K. H. Inamdar.** (2020).
   "Web-based IDE with Compiler as a Service." Proceedings of the International Conference on Computing Methodologies and Communication (ICCMC).
   [IEEE Xplore]
   ✓ Explains implementing multi-language compiler APIs in web-based IDEs.
3. **Tharun K., et al.** (2019).
   "Collaborative Code Editor for Real-Time Programming and Learning." International Journal of Scientific & Engineering Research (IJSER), Vol. 10, Issue 4.
   [https://www.ijser.org/]
   ✓ Covers Firebase usage in real-time code sharing and session sync.
4. **A. Yadav and R. Sharma.** (2022).
   "Enhancing Remote Coding Interviews with Collaborative Cloud IDE." International Journal of Engineering Trends and Technology (IJETT). [DOI: 10.14445/22315381/IJETT-V70I4P206]
   ✓ Focuses on remote interviewing and collaboration challenges.
5. **D. Patel, J. Joseph, and A. Shah.** (2021).
   "A Comparative Study of Web-based IDEs for Software Engineering Education." International Journal of Computer Applications.
   [https://www.ijcaonline.org/]
   ✓ Compares CodePen, Replit, Paiza.IO, and VS Code Live Share from an education perspective.