Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 2, March-April 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i02.1246

CRITIC: SMART WEB-BASED REAL-TIME CODE EDITOR

Arya Gupta[1], Harsh Khare[2], Riya Parihar[3], Vineeta Barasker[4], Dr. Pankaj Singh Sisodiya[5]

Abstract:
A high-performance real-time collaborative code editor designed for today's software teams that work geographically apart has been developed as a project. This editor has been built with the capability of developing in multiple programming languages, such as JavaScript, Python, Java, C++, among others; it includes a secure and robust version control system, providing a professional development environment for all users. The technology behind this platform is based on the combination of WebSocket and an Operational Transformation (OT) algorithm. WebSocket allow for continual bidirectional communication over low latency and very low lag. The OT algorithm allows for data from multiple users to be synchronised using intelligent logic to prevent the generation of conflicting documents from many users making concurrent edits simultaneously.
Additionally, several advanced data handling techniques are used to enhance the overall user experience (i.e., edit batching, redundancy minimisation, etc.). These techniques can eliminate a significant amount of lag and server overhead, resulting in faster and more efficient use of resources. The security of all users' credentials and project integrity is accomplished using a sophisticated authentication mechanism and state-of-the-art hashing techniques. This platform offers an extremely responsive and scalable solution for various uses, including remote teamwork, real-time technical interviews, and live interaction within a classroom environment. This research provides an efficient system for multiple users to work with and high-fidelity synchronisation, and therefore an optimised and reliable framework for the next generation of web-based development tools.

Keywords: HTML, CSS, JavaScript, React, WebSocket, Operational Transformation.

1. INTRODUCTION

Developers have switched from developing large desktop applications with large IDEs to creating web apps that are developed in collaborative agile environments. In recent years, with the introduction of cloud technologies and WebSockets, indicates that editors that run on a browser (web-based editors) can now perform similarly to desktop-based editors. However, even with these newfound capabilities, there is still a lot of work to do before we will see an integrated solution that provides role-based permissioning, multi-language execution, and built-in community-based discussion (like the way Microsoft Teams provides collaborative tools).

This project aims to fill those gaps by providing a collaborative editing environment like the Visual Studio Code (VS Code) environment by integrating the Monaco Editor to provide a similar interface and providing a highly robust synchronisation model (utilising WebSockets and Operational Transformation (OT) to support low-latency bidirectional communication and managing concurrent edits) The OT-based model we selected has been enhanced to provide optimal edit batching and thus provide superior performance and stability for coding in high-intensity sessions. Each programming language will have

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 2, March-April 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i02.1246

isolated child processes and utilise sandboxes for creating temporary files to ensure there is no potential for a malicious user to disrupt your work while you are executing code in a multi-language environment (e.g., executing Python, JavaScript, Java through C++).
In addition to providing a high-performing technical synchronisation environment, we will create a built-in community forum so that users can learn from each other and get real-time assistance with their coding issues. By providing unified code editing, version control and secure code execution capabilities through a single browser-based system, our platform modernises the development process. This research ultimately bridges the gap between local development power and the accessibility of cloud-based collaboration.

2. LITERATURE REVIEW

Roberts (2025) – Trends in Cloud-Based Software Development Tools This paper provides insights into the convergence of collaboration, cloud execution, and AI features in developer tools. It supports the vision behind this project—a unified browser-based IDE supporting real-time development, code quality checks, and integrated community communication. [1]

Nguyen et al. (2024) – Collaborative Learning through Integrated Forums Researchers found that integrating discussion forums directly into coding platforms promotes peer learning and faster problem-solving. This influenced the inclusion of a built-in community forum for developers to exchange ideas, share snippets, and upvote solutions. [2]

Das & Mehta (2024) – Secure Execution Environments for Code Editors This paper discusses sandboxing and temporary file handling for running untrusted code securely. The backend uses similar techniques with spawned processes, ensuring each user's code executes in isolation, minimizing security risks. [3]

Lee et al. (2024) – Progressive Web Applications: Bridging Web and Native Experience This work analyzed PWA architecture that combines offline access and native-like interfaces. The editor uses this to enable developers to install the app on desktops or mobile devices, maintaining offline functionality for local edits and auto-syncing once reconnected. [4]

Wilson & Huang (2023) – Version Control Systems in Multi-Language Environments The study compared Git-based and semantic versioning systems, emphasizing the importance of version tracking for maintaining compatibility. This inspired the integration of version control alerts that notify users when their language version becomes outdated. [5]

Patel & Singh (2023) – AI-Assisted Linting and Error Detection in IDEs The authors analyzed the impact of linting tools integrated with AI models for detecting syntax and logical errors. This implementation adapts the concept using lightweight linting modules that analyze code in real time, offering immediate feedback without server overload. [6]

Kumar et al. (2022) – WebSocket Communication for Real-Time Applications This research explained WebSocket's bi-directional nature, ideal for continuous data exchange. The paper highlighted how real-time chat and collaborative applications benefit from WebSocket over traditional HTTP polling, influencing the choice to use Web Sockets for efficient synchronisation. [7]

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 2, March-April 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i02.1246

Jackson & Lewis (2021) – Browser-Based IDEs: The Evolution of Web Development Tools This paper reviewed modern in-browser IDEs like StackBlitz and CodeSandbox, identifying advantages like zero-setup coding and limitations like reduced debugging depth. These insights guided the focus on a lightweight architecture combined with deeper language support. [8]

Sun & Ellis (2020) – Conflict-Free Replicated Data Types (CRDTs) for Distributed Systems CRDTs were introduced as a mathematically sound way to merge concurrent operations without conflict. While this approach influences many open-source projects, our editor implements OT over Web Sockets to achieve lower latency and simpler server management. [9]

Zhang et al. (2019) – Real-Time Collaborative Editing Systems This study explored Operational Transformation (OT) algorithms for real-time editing platforms like Google Docs. The authors analyzed synchronization mechanisms for text consistency, ensuring that simultaneous user actions maintain data integrity and document order. [10]

3. METHODOLOGY

The methodology for the CRITIC code editor is built upon the following algorithms..

Operational Transformation (OT)

This core synchronization algorithm resolves conflicts during concurrent multi-user editing by transforming local operations to ensure all participants maintain a consistent document state.. It works by breaking down every text change—such as inserting or deleting a character—into a specific "operation" that includes the type of action and its exact position (index) in the document. When multiple users edit the same line simultaneously, the OT engine "transforms" these operations so that the final state of the document remains identical for every participant, effectively preventing text from becoming jumbled or overwritten.

In our CRITIC project, OT acts as the "referee" for the Monaco Editor to ensure seamless teamwork. As you type, your changes are sent as operations via Socket.io to the central server, which maintains a history of all edits. If the server detects that another collaborator's edit arrived a millisecond earlier and shifted the text, it automatically adjusts the index of your edit before broadcasting it to the rest of the room. This process allows CRITIC to support high-concurrency coding sessions without the need for file locking, ensuring that every user sees the exact same code in real-time.

Role-Based Access Control (RBAC)

This algorithm manages project security by validating user requests against specific assigned roles, such as Owner or Viewer, to ensure only authorized individuals can modify the codebase. Role-Based Access Control (RBAC) is a security method that restricts system access to authorized users based on their assigned roles within an organization. Instead of giving every user individual permissions, RBAC groups permissions into "roles," making it much easier to manage security as a project grows. It acts as a digital guard that ensures people can only see or change the parts of the system that are necessary for their specific job.

In our CRITIC project, RBAC is used to manage teamwork and prevent accidental code deletion. When a user joins a room, the system identifies them as an Admin (who can manage users), a Member (who can write and run code), or a Viewer (who can only watch the screen). Before any action is taken—like clicking the "Run" button—the system checks the user's "Identity Pass" (stored in a JWT) to confirm they have the right permission for that specific task.

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 2, March-April 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i02.1246

Publish-Subscribe (Pub/Sub) Model

The Publish-Subscribe (Pub/Sub) pattern is a messaging architecture where the senders of messages (publishers) do not send them directly to specific receivers (subscribers). Instead, messages are categorized into "topics" or "channels," and subscribers express interest in one or more of these topics to receive only the relevant data. This creates a "decoupled" system where the publisher doesn't need to know who or how many people are listening, allowing the application to scale efficiently as more users join.

In our CRITIC project, the Pub/Sub model is the engine behind the Discussion Forum and real-time room updates. When you send a chat message, your client acts as a Publisher and sends the data to a specific Room ID on the server. The server then identifies every other user who has "Subscribed" to that particular Room ID via Socket.io and pushes the message to their screens instantly. By using this pattern, CRITIC ensures that conversations stay private to their respective rooms while maintaining the high speed required for live technical collaboration.

Language Server Protocol (LSP)

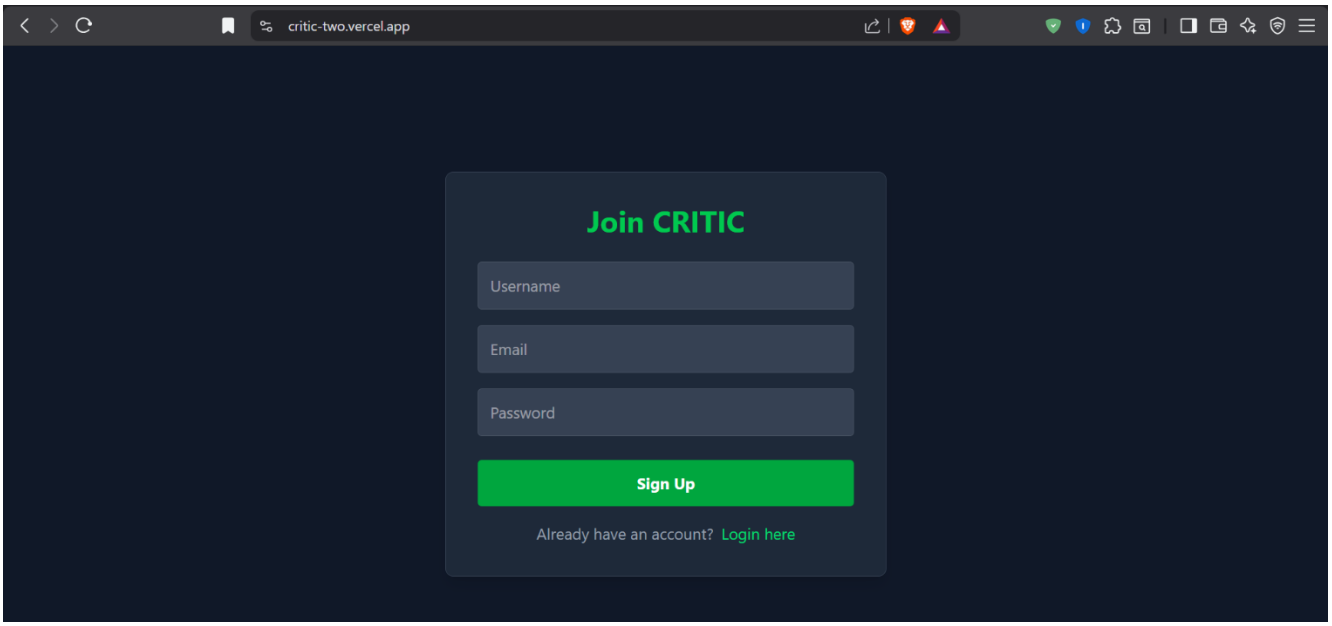
This protocol enables standardized communication between the editor and language-specific backends to provide “smart” features like syntax highlighting and IntelliSense

WebSocket

Web Sockets maintain an open and persistent connection to your client and server, enabling real-time, two-way communication between your client and your server, as opposed to traditional methodologies such as using HTML requests. This ability allows for instant updating the data in your web applications, helping to create features, such as live chat, notifications, or real-time dashboards.

4. RESULTS

4.1 Registration/Login page

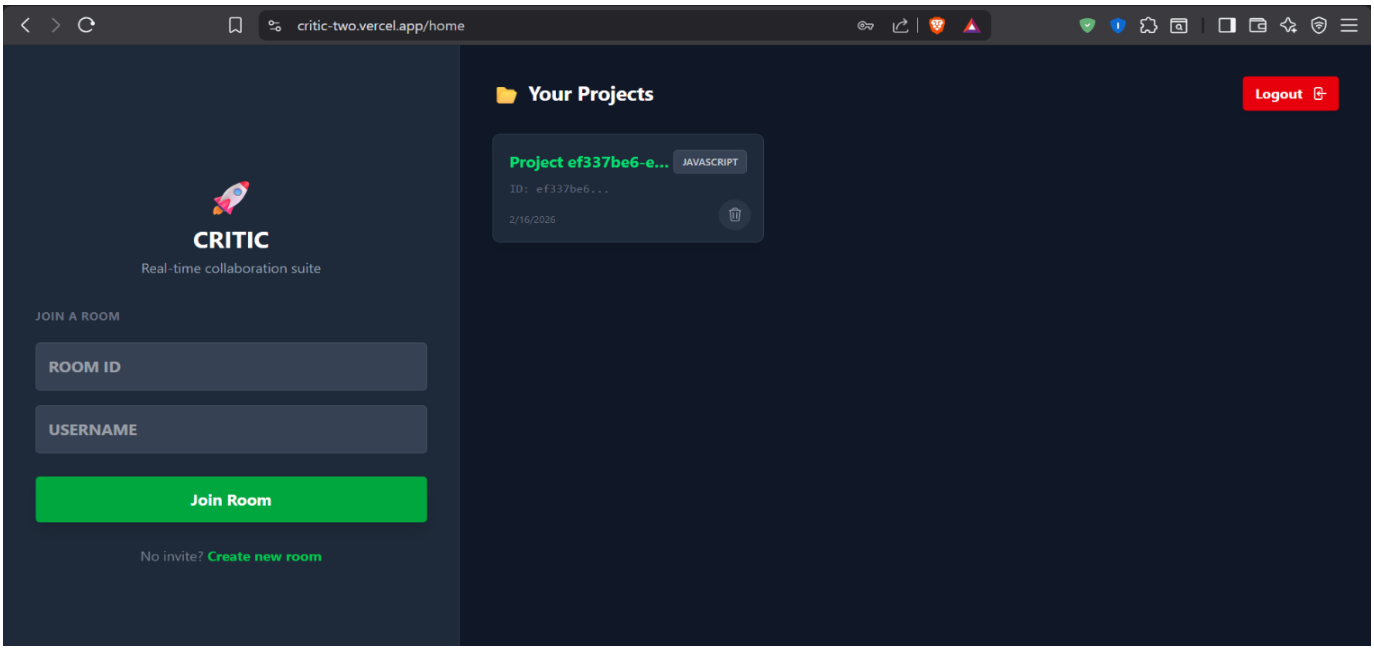


Screen 4.1: Registration/Login Page

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 2, March-April 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i02.1246

The login screen serves as the secure entry point for the CRITIC platform. It features a clean, dark-themed UI consistent with modern developer tools. The implementation includes input validation for the email and password fields to ensure data integrity before sending requests to the backend. Below the primary "Login" button, a "Create one" link facilitates new user registration. This screen is critical for identifying users so that their collaborative actions and saved projects can be correctly attributed to their accounts in the database.

4.2 Dashboard

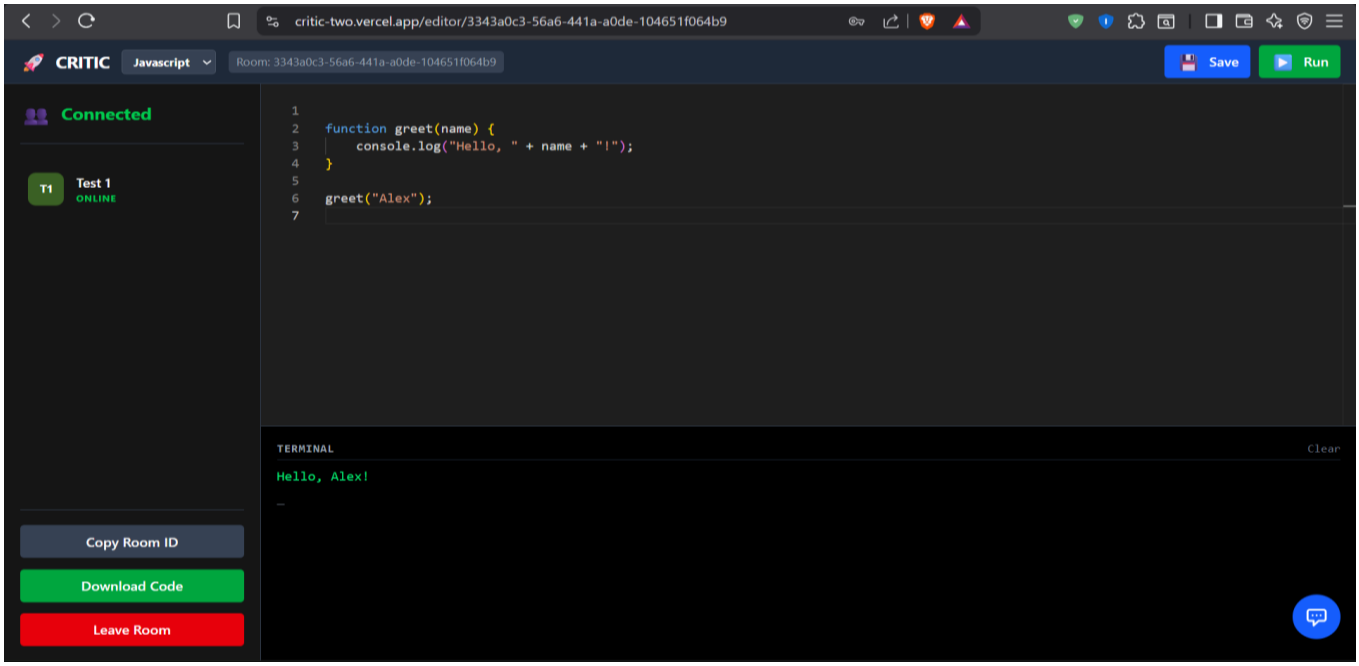


Screen 4.2: Dashboard

After logging in, users are directed to this dual-purpose dashboard. On the left, the "Join a Room" interface allows users to enter a specific Room ID and choose a display name to start collaborating immediately. On the right, the "Your Projects" section displays a grid of saved work. Each project card shows the project name, the primary language (e.g., JAVASCRIPT), a unique ID, and the last modified date. This screen demonstrates the system's ability to manage persistent storage alongside real-time room-based interactions.

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 2, March-April 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i02.1246

4.3 Editor Window

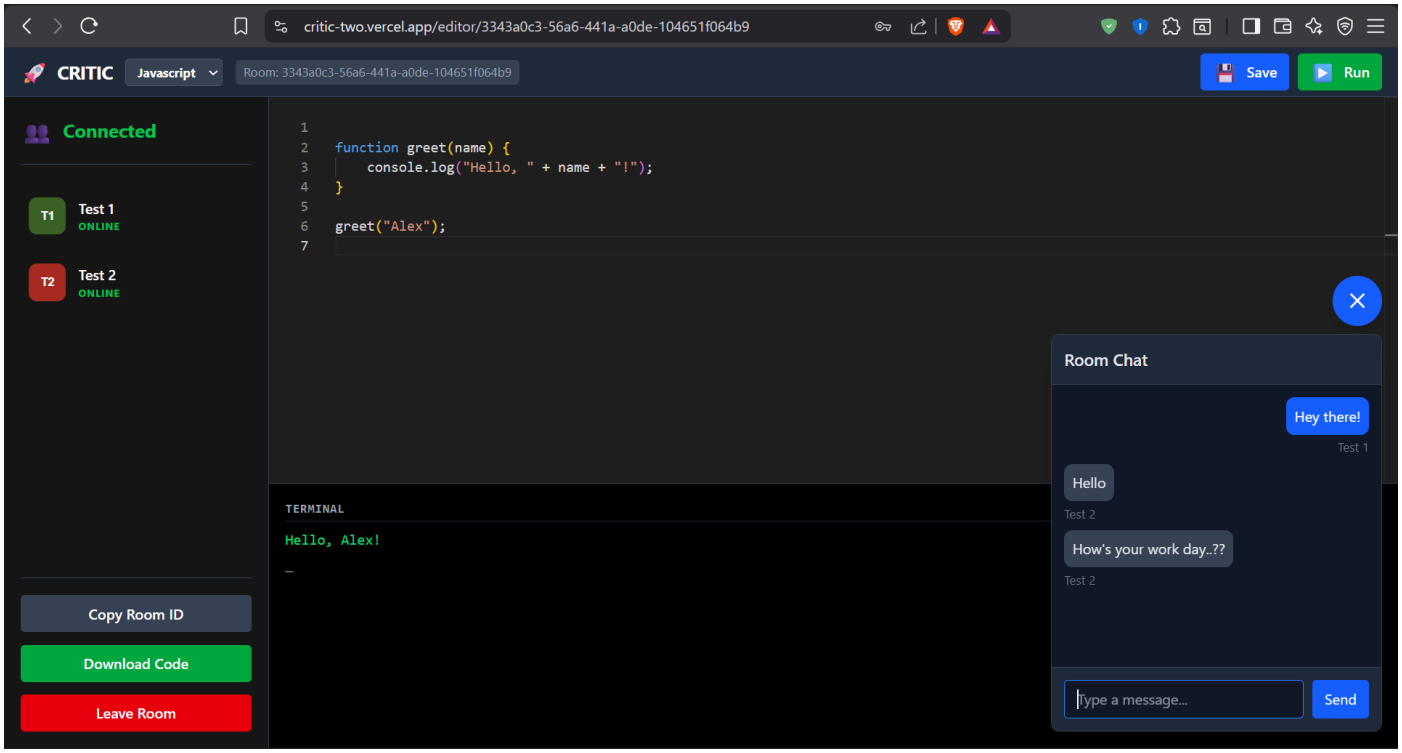


Screen 4.3: Code Editor Window

This is the primary interface where the "greet" function logic is being implemented. The screenshot displays the Monaco-powered editor with syntax highlighting for a JavaScript function. Below the editor, the integrated "Terminal" successfully displays the execution output: "Hello, Alex!". This confirms that the code execution module is functioning correctly. On the left, the "Connected" panel shows real-time presence (e.g., users "test 1" and "Vineeta"), while the bottom buttons provide essential actions like "Copy Room ID," "Download Code," and "Leave Room."

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 2, March-April 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i02.1246

4.4 Chat room



Screen 4.4: Real-Time Chat Room

The Room Chat overlay, triggered by the message icon, facilitates instant communication without leaving the coding environment. The screenshot shows a threaded conversation between "test 1" and "test 2." This module is essential for pair programming, allowing developers to discuss logic changes or debug issues in tandem with the code visible on the screen. The UI distinguishes between sent and received messages using color-coded bubbles, ensuring a clear and intuitive communication flow during high-intensity development sessions.

5. SECURITY AND ENCRYPTION

- **Authentication:** The system uses secure password hashing and JSON Web Tokens (JWT) for session management and edit validation.
- **Role-Based Access Control (RBAC):** Users are assigned specific roles—Owner, Editor, Reviewer, or Viewer—to prevent unauthorised code overwriting and manage workflow permissions.
- **Sandboxing:** To prevent malicious code execution from affecting the server, the platform uses sandboxed environments (temporary files and child processes) for isolated code execution.

6. CONCLUSION

The web application Critic - a smart real-time code editor - was analysed as part of this review paper to illustrate the efforts towards a unified and frictionless experience for distributed software teams. Through the delivery of a web-accessible Integrated Development Environment (IDE), Critic removes the barriers

Advanced International Journal of Multidisciplinary Research
E-ISSN: 2584-0487
editor@aijmr.com
Volume 4, Issue 2, March-April 2026
CrossRef DOI: 10.62127/aijmr.2026.v04i02.1246

associated with installation and dependency management that have historically plagued coding environments.

The application of distributed computing principles has been vital in the development of Critic. Specifically, the use of Operational Transformation (OT) in Critic provides for consistency of concurrent edits made by multiple team members, ensuring that users are able to edit concurrently with intentional preservation of user intentions. Additionally, Critic's implementation of Role Based Access Control (RBAC) provides for a secure codebase and formalizes the collaborative workflow.
In addition to Critic providing multi-language support and high levels of code quality via built-in linting features, it also demonstrates the viability of the next generation of cloud-native development tools, by creating a scalable, secure, and uniquely equipped platform that meets the modern collaborative coding environment.

In conclusion, the development of Critic represents a major movement towards a unified cloud-based development environment that offers users the flexibility of using web tools while maintaining the power and richness of traditional IDEs.

REFERENCES:

1. Roberts, Henry. "*Trends in Cloud-Based Software Development Tools.*" 2025.
2. Nguyen, Hoang, et al." *Collaborative Learning through Integrated Forums.*" 2024.
3. Das, Arjun, and Priya Mehta. "*Secure Execution Environments for Code Editors.*" 2024.
4. Lee, Eun, et al." *Progressive Web Applications: Bridging Web and Native Experience.*" 2024.
5. Wilson, James, and Min Huang. "*Version Control Systems in Multi-Language Environments.*" 2023.
6. Patel, Ankit, and Rohan Singh. "*AI-Assisted Linting and Error Detection in IDEs.*" 2023.
7. Kumar, Rajesh, et al. "*WebSocket Communication for Real-Time Applications.*" 2022.
8. Jackson, David, and Robert Lewis. "*Browser-Based IDEs: The Evolution of Web Development Tools.*" 2021.
9. Sun, Cheng, and Clarence Ellis. "*Conflict-Free Replicated Data Types for Distributed Systems.*" 2020.
10. Zhang, Wei, et al. "*Real-Time Collaborative Editing Systems.*" 2019.