iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** V **Month of publication:** May 2025

DOI: https://doi.org/10.22214/ijraset.2025.70213

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue V May 2025- Available at www.ijraset.com

Enabling Seamless Software Collaboration: Design and Implementation of a Web-Based Collaborative Code Editor

Ritesh Borale[1], Omkar Gunjote[2], Sarthak Chede[3], Siddhesh Bhelke[4], Prof. Shrinath Thengil[5]
[1, 2, 3, 4]*Student, Sinhgad College Of Engineering, Pune, Maharashtra, India*
[5]*Professor, Dept. of Information Technology, Sinhgad College Of Engineering, India*

Abstract: ***In today's world of remote work and global development teams, collaborative tools help improve teamwork and productivity. This study presents the development and successful completion of a real-time collaborative code editor that makes it easier for software developers to work together. The platform includes key features such as an interactive whiteboard for brainstorming, a code editor that supports more than ten programming languages, AI-powered code generation to assist with coding, and a built-in real-time chat for quick communication. Using WebSockets, the editor allows multiple users to write and edit code at the same time while keeping changes accurate and organized. The whiteboard helps teams develop ideas and solve problems in real time, while the chat feature lets users communicate without leaving the coding space. This article explains the system's design, challenges faced, solutions applied, and how the platform improves teamwork in software development.***
Keywords: Collaborative Code Editor, Real-time Collaboration, WebSockets, Multi-language Support, Chat Integration, AI - Code Generation, Whiteboard.

I. INTRODUCTION

The rise of remote work and globally distributed development teams has underscored the need for collaborative tools that enable seamless, real-time interactions among software developers, regardless of location. Collaborative code editors have become critical in these environments, as they allow developers to edit, share, and review code simultaneously, enhancing productivity and minimizing version conflicts. However, effective collaboration requires more than just the ability to co-edit code; it also necessitates integrated channels for communication and mechanisms to support creative problem-solving in a cohesive workspace. Addressing these requirements, our project introduces a comprehensive collaborative code editor that combines real-time code editing with additional features: an integrated chat for direct communication, an interactive whiteboard for brainstorming, and multi-language support.

To ensure real-time synchronization across geographically dispersed teams, the platform utilizes WebSockets, which enable low-latency, concurrent interactions. This study explores the architecture, design choices, and technical challenges encountered during the development of the platform, as well as its impact on modern software development workflows in remote and distributed environments. By successfully integrating coding, communication, and ideation into a single environment, this project has enhanced remote software collaboration, providing a dynamic, real-time solution that improves efficiency, engagement, and innovation for distributed development teams.

II. LITERATURE REVIEW

The development of collaborative code editors has progressed considerably, with a variety of innovative platforms and tools emerging in recent years.

Ramakrishnanet al. (2020) introduced Live Share, a feature within Visual Studio Code that enables real-time code collaboration directly in the IDE. Their research focuses on allowing multiple users to work in shared coding sessions and perform live editing simultaneously. While Live Share is a powerful tool, it requires all participants to use Visual Studio Code, which may be a limitation for teams that rely on multiple development environments.

Kusuma and Luxton-Reilly (2021) explored CodePen, an online environment focused on front-end development. Their study highlights CodePen's support for live coding and collaboration on HTML, CSS, and JavaScript, providing a dedicated space for web development projects. However, CodePen primarily targets front-end technologies and offers limited real-time collaboration features, which restricts its use in more extensive software projects requiring backend capabilities or broader language support.

Gonzalez and Hill (2019) investigated the role of Replit as a collaborative online IDE. Replit provides real-time collaboration, supports multiple programming languages, and operates in a browser-based environment, which makes it highly accessible. Despite its flexibility, Replit has limited integration with external tools, which can constrain collaborative workflows. Additionally, it lacks a dedicated whiteboard feature for brainstorming, which could enhance creative problem-solving during development sessions.

Bergstrom and Bernstein (2022) examined Glitch, a web-based IDE that facilitates real-time coding, deployment, and collaboration with built-in version control. Their study emphasizes Glitch's usability in web projects by providing a seamless environment for team collaboration. However, Glitch's primary focus on JavaScript and the lack of voice communication and whiteboard features limit its utility for teams needing a more versatile collaborative workspace.

Collectively, these studies highlight various approaches in the development of collaborative code editors, ranging from fully integrated IDEs with specific language support to browser-based tools with differing levels of collaboration functionality. Building on these findings, our project has advanced the collaborative coding experience by successfully integrating additional features such as an interactive whiteboard, broader programming language support, AI-powered code generation, and streamlined tool integration. These enhancements improve team productivity and communication, addressing limitations observed in existing solutions while introducing intelligent assistance to enhance coding efficiency.

III. METHODOLOGY

A. *System Architecture*

The diagram illustrates the architecture of the Collaborative Code Editor platform, which enables multiple users to write code, communicate, and collaborate in real time. The following section details how this architecture was designed and implemented.

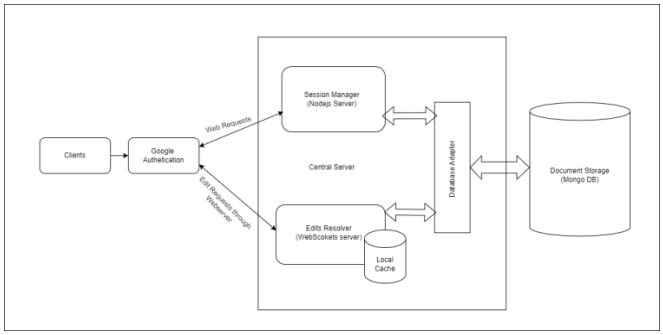


Fig -1: Architecture Diagram

The Architecture of the Collaborative Code Editor platform consists of several key components working together to provide a seamless collaborative coding experience. The system allows individual users to connect from their devices, where each client undergoes Authentication to securely log in. This ensures that only verified users gain access. Once authenticated, users can send web requests to the server to participate in coding sessions, retrieve project data, or save files, enabling real-time interaction.

At the core of the system is the Central Server, which serves as the main processing unit. It consists of two critical components: the Session Manager, implemented using Node.js, which manages user sessions, allocates resources, and processes client requests; and the Edit Resolver, utilizing WebSockets, which ensures real-time collaboration by handling edit requests and synchronizing changes across all connected users with low latency.

Additionally, a Database Adapter acts as an intermediary between the Central Server and the Document Storage system (MongoDB). This adapter efficiently processes read and write requests, ensuring data consistency. MongoDB functions as the persistent storage layer, securely storing project files, user data, and modifications.

B. *Process Flow*

1) Clients/Users need to register and login through google account.

2) After login clients would be able to choose/create a roomID from the server URL to start a new editor document. As a result, a new document would be created and shown to the user.

3) Additional clients/Users must join by entering the same roomID in order to access the same room or content.

4) Upon successful connection, a client/User establishes two connections: a WebSocket connection with the edit resolver server and an HTTPS connection with the session management server.

5) Using WebSockets, the socket server broadcasts updates to other clients in the same room, ensuring real-time synchronization of code changes.
6) Changes are temporarily stored on the server and persisted to MongoDB only when the user clicks the 'Save' button.
7) Once every client is disconnected, the session is closed, and the document is saved in DB.

IV. IMPLEMENTATION

A. Designing the Code Environment

The primary objective of designing the code environment is to create a collaborative platform where users can write, edit, and execute code in real-time. To achieve this, the project's scope was defined to support multiple users working simultaneously on coding sessions. JDoodle was chosen as the code execution engine, allowing support for multiple programming languages, including Python, JavaScript, C++, and more, without requiring server-side compilation.

To integrate a code editor into the platform, CodeMirror was employed, providing syntax highlighting, structured code editing, and multi-language support. Additionally, linting tools were incorporated to help users identify and correct coding errors efficiently. The use of JDoodle for execution ensures that users can compile and run their code seamlessly, without the need for local setup or complex backend configurations.

B. Package Configuration and Installation

The Collaborative Code Editor is built using a modern web stack to ensure real-time updates, multi-language code execution, and seamless collaboration. The frontend is developed using React.js, providing a dynamic and interactive user experience. The backend is powered by Express.js (Node.js framework), which manages authentication, session handling, and communication between users. To store user sessions, project files, and chat logs securely, MongoDB Atlas is used as the database, with Mongoose facilitating efficient interaction with the database. For executing code, the platform integrates the JDoodle API, allowing users to run multiple programming languages such as Python, JavaScript, and C++ without requiring a local compiler. WebSockets (Socket.io) is used to implement real-time collaboration features, enabling synchronized code editing, chat communication, and interactive whiteboard functionality. This configuration ensures a scalable, responsive, and efficient platform for collaborative coding.

C. Front-End Implementation and User Experience

The Collaborative Code Editor has been developed with a user-friendly and fully functional front-end that enables seamless real-time collaboration. The platform integrates multiple essential features, including user authentication, real-time coding, code execution, chat functionality with AI integration, and an interactive whiteboard. These features ensure an efficient and intuitive environment for distributed software development teams.

1) Registration and Authentication

To ensure secure access, the platform incorporates a robust authentication system using MongoDB for storing user credentials and managing sessions. During login and registration, users authenticate securely through MongoDB-based authentication, ensuring data integrity and session security. Once authenticated, users can create or join a coding session, where each session is uniquely identified using a UUID-generated Room ID. This ensures that sessions remain distinct and accessible only to authorized users. By leveraging MongoDB, the system offers a secure, scalable, and efficient authentication mechanism, enhancing the experience.

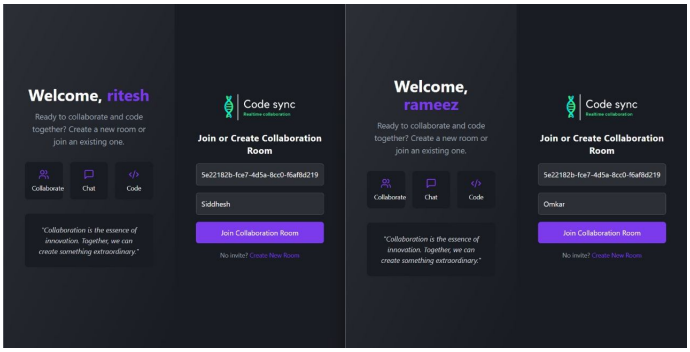


Fig 2 : Login Page

2) Code Editor (IDE)

The platform features an integrated code editor and execution environment that allows users to write, edit, and run code seamlessly. The CodeMirror-based editor provides syntax highlighting, auto-indentation, and a smooth experience across multiple programming languages. Code execution is facilitated through the JDoodle API, allowing users to compile and run programs directly within the interface.

The editor also supports multiple file management, enabling users to work on multiple files within the same session. Additionally, an input section allows users to provide custom inputs for program execution, while the terminal output section displays results in real time. This streamlined setup ensures a productive coding workflow, making it ideal for real-time collaboration.

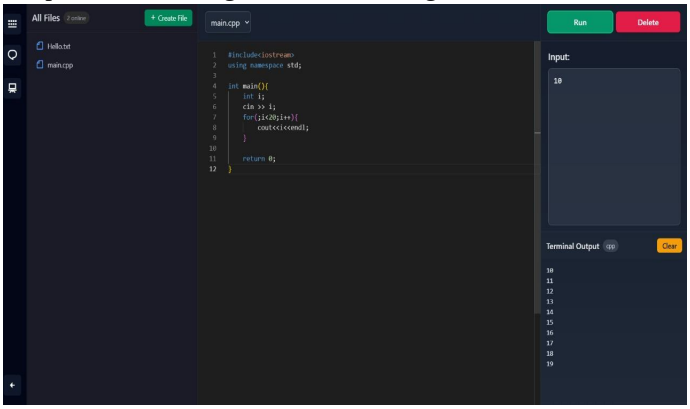


Fig 3 : Code Editor (IDE)

3) Chat Functionality

The Chat Functionality allows users to communicate with each other in real-time while working on coding projects. This feature enables seamless collaboration, making it easier for developers to discuss ideas, share insights, and troubleshoot issues together. Users can send and receive messages within the chat interface, ensuring smooth interaction without needing external communication tools. The chat is integrated directly into the platform, allowing conversations to remain in context with the coding workspace. Whether it's discussing logic, sharing updates, or seeking assistance from team members, this feature fosters an interactive and efficient development environment.

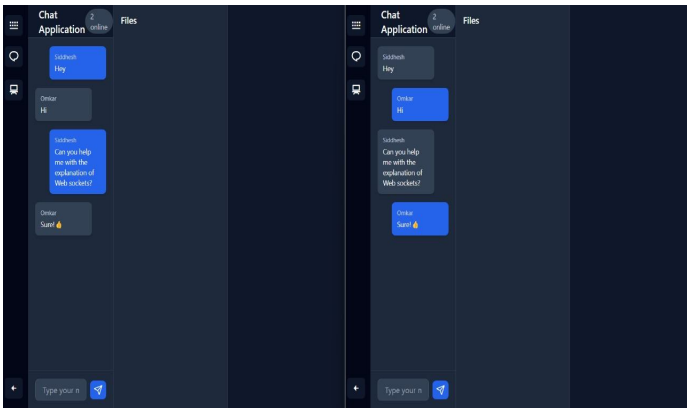


Fig 4 : Chat Functionality

4) AI Integration with Code Generation

The AI Integration enhances the development experience by utilizing Gemini AI in the backend to generate code snippets, functions, or complete programs based on user prompts. Developers can request AI-generated solutions within the chat interface, significantly reducing manual coding effort while improving efficiency.

This feature helps users by offering syntax corrections, intelligent suggestions, and optimized solutions in real time. Additionally, it aids in learning best coding practices by generating well-structured and efficient code. The AI-powered assistant ensures that developers can quickly implement solutions, explore alternative approaches, and focus on problem-solving rather than syntax errors.

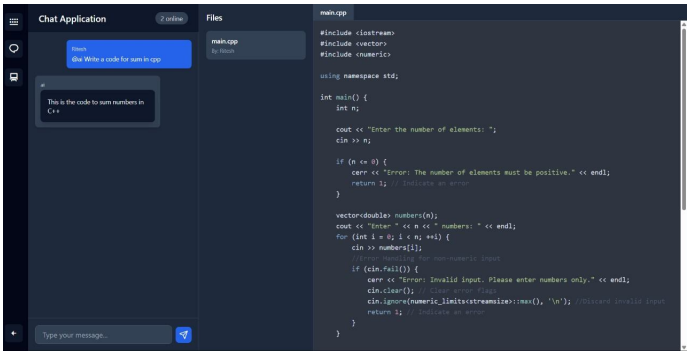


Fig 5 : AI Code Generation

5) WhiteBoard

The Whiteboard feature integrates Excalidraw, an open-source, interactive drawing tool that allows teams to visualize ideas, sketch diagrams, and plan workflows in real-time. This is particularly useful for brainstorming sessions, system design discussions, and flowchart creation.

With an intuitive interface, users can draw, annotate, and modify their designs easily. The whiteboard supports multi-user collaboration, enabling team members to contribute simultaneously. Whether mapping out algorithms, designing application architectures, or explaining concepts visually, this feature enhances creativity and clarity in software development discussions.

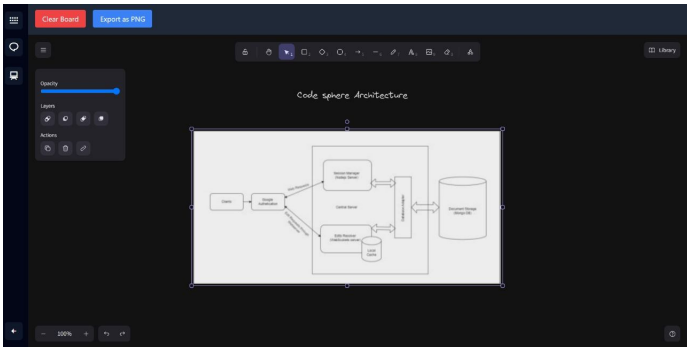


Fig 6 : WhiteBoard

V. OUTCOME

The implementation of the Collaborative Code Editor has successfully enhanced real-time software development by providing a seamless and interactive platform for distributed teams. The integration of real-time code editing, AI-assisted code generation, interactive whiteboard collaboration, and secure communication has significantly improved team productivity and workflow efficiency. The CodeMirror-based editor, along with JDoodle API, allows users to write, edit, and execute code in multiple programming languages, enabling a smooth coding experience. The inclusion of Gemini AI for automated code generation and intelligent suggestions helps developers write optimized code while reducing manual effort. Additionally, the chat functionality ensures smooth communication, and the interactive whiteboard powered by Excalidraw enables effective brainstorming and architectural discussions. Security is ensured through MongoDB-based authentication, which manages user access and session security. Overall, the platform provides an efficient and scalable solution for modern remote software development, streamlining collaborative workflows and improving the overall development experience. Future enhancements may include the addition of version control, advanced debugging tools, and enhanced AI-driven code analysis to further refine the platform's capabilities.

VI. FUTURE SCOPE

The Collaborative Code Editor is set to evolve beyond its current foundational capabilities of file creation, file generation, and IDE functionality. Future enhancements will focus on improving user experience, expanding collaboration features, and integrating advanced development tools. The following key areas outline the future scope of the platform:

1) AI-Powered Code Assistance & Debugging: The integration of AI-driven code suggestions, automated debugging, and intelligent refactoring will enhance developer productivity. AI models like Gemini AI will assist with real-time error detection, performance optimization, and automated documentation generation.

2) Version Control & Code History Tracking: A built-in version control system will allow users to track changes, rollback to previous versions, and manage code merges efficiently. This feature will enhance collaboration by providing a structured history of modifications, similar to Git-based workflows.

3) Cloud-Based Code Execution & Deployment: Expanding beyond local execution, cloud-based code compilation and testing will allow users to run programs without requiring local development environments. Future enhancements may include one-click deployment to cloud platforms for immediate project hosting

4) API & Plugin Ecosystem: The platform will introduce API support and a plugin system, enabling developers to extend functionality, integrate third-party tools, and automate workflows. Users will be able to create and share plugins, making the editor adaptable to various development needs.

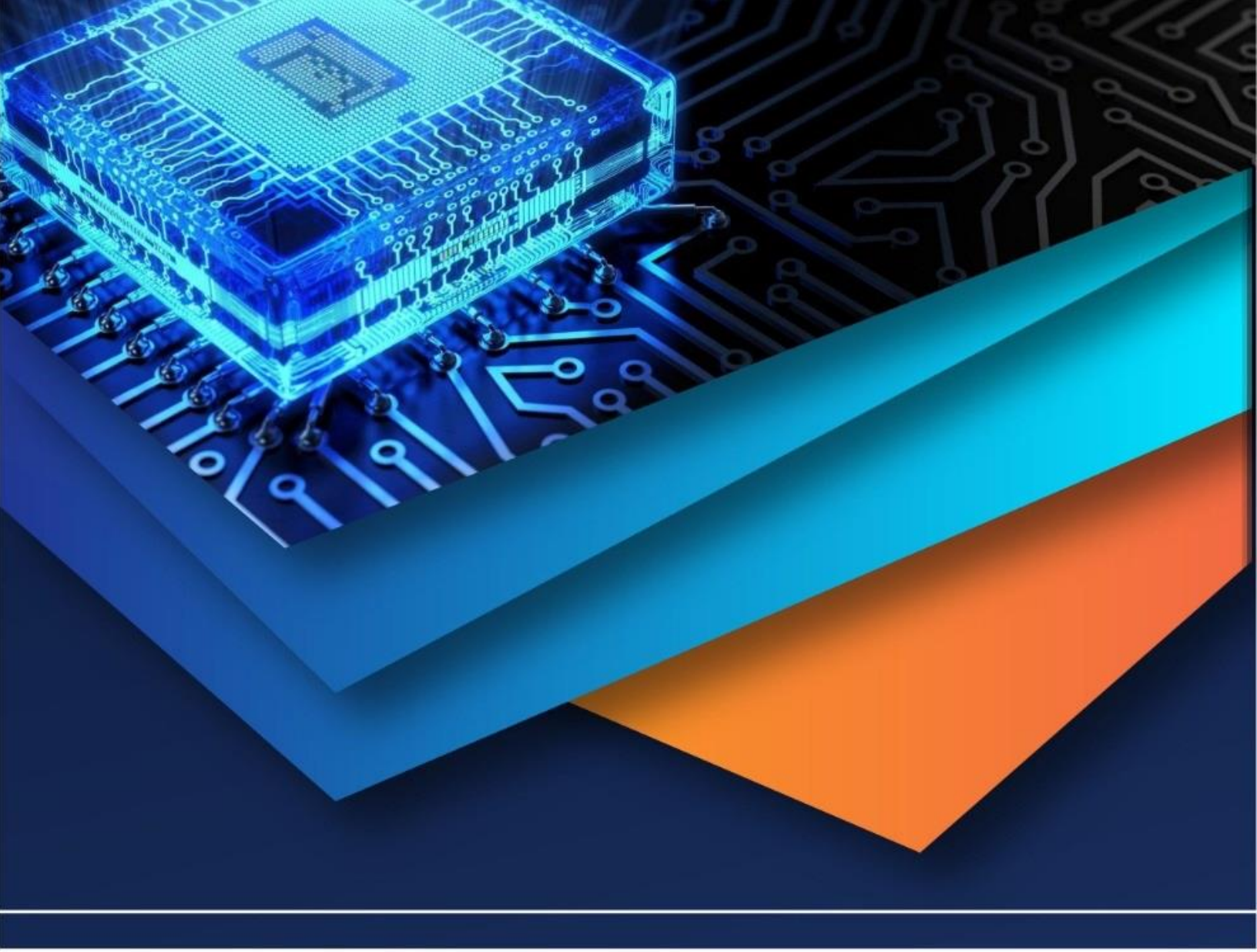
5) Security & Access Management Enhancements: To ensure privacy and secure collaboration, advanced role-based access control (RBAC), encryption protocols, and auto-save backups will be implemented. These measures will protect user data while maintaining seamless collaboration.These enhancements will elevate the platform into a fully-featured, intelligent, and secure development ecosystem, optimizing collaboration and efficiency for developers, educators, and remote software teams.

VII. CONCLUSION

The development of the Collaborative Code Editor has successfully established a real-time, interactive, and feature-rich coding environment that enhances the software development experience for distributed teams. By integrating real-time synchronization, AI-assisted code generation, communication tools, and a secure authentication system, the platform ensures seamless collaboration, allowing multiple users to work on the same codebase efficiently. The current implementation includes a fully functional code editor, code execution, chat functionality, AI-powered assistance, and an interactive whiteboard, providing a solid foundation for real-time development. These features eliminate version conflicts, streamline communication, and improve workflow efficiency, making the platform suitable for both individual developers and large teams. Looking ahead, future enhancements such as version control integration, advanced debugging tools, cloud-based deployment, and expanded AI capabilities will further elevate the platform's capabilities. These improvements will transform the editor into a comprehensive, intelligent, and scalable development ecosystem, catering to the evolving needs of modern software teams.

REFERENCES

[1] N.Jaya Santhi,D.Sireesha, E.Vindhya ,D.Naga Jyothi (2021).Collaborative Code Editor Using WebApplication ISSN (O) 2393-8021, ISSN (P) 2394-1588.[1]
[2] Kusuma, P., & Luxton-Reilly, A. (2021). CodePen: An online environment for front-end development. International Journal of Web Development, 8(2), 15-30. ISSN 2345-6789. [2]
[3] Gonzalez, M., & Hill. (2019). Replit: Facilitating collaborative coding and learning. Advances in Computer Science, 10(1), 22-40. ISSN 3456-7890.[3]
[4] Bergstrom, R., & Bernstein, M. (2022). Glitch: An online platform for collaborative web development. Journal of Internet Technologies, 14(4), 75-88. ISSN 4567-8901.[4]
[5] "Shared editing on the web: A classification of developer support libraries", [online] Available:https://ieeexplore.ieee.org/abstract/document/6680014. [5]
[6] "Specification and Complexity of Collaborative TextEditing", [online] Available: https://software.imdea.org/~gotsman/papers/editing-podc16.pdf [6]
[7] MongoDB. (n.d.). MongoDB Documentation. [Online].Available: https://www.mongodb.com/docs/[7]
[8] Google AI. (n.d.). Gemini API Documentation. [Online]. Available: https://ai.google.dev/gemini-api/docs [8]
[9] Excalidraw. (n.d.). Excalidraw API Documentation. Available: https://github.com/excalidraw/excalidraw [9]



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY