iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** XI **Month of publication:** November 2025

DOI: https://doi.org/10.22214/ijraset.2025.75646

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 XI Nov 2025- Available at www.ijraset.com

CollabCode: A Real-Time Code Sharing Space

Medha Wyawahare[1], Milind Rane[2], Chandrashekhar Chandekar[3], Soham Chavan[4], Atharva Chaudhari[5]
Department of Electonics and Telecommunication Engineering, VIT, Pune, India

Abstract: ***This paper introduces CollabCode, a web-based real-time collaborative Integrated Development Environment (IDE) designed to enable developers to code, debug, and communicate within shared sessions. The platform offers a comprehensive toolkit including live code synchronization, in-built team chat, AI-assisted code support, and collaborative task tracking. Built with Node.js, Express.js, WebSockets, and Google Gemini AI, the system leverages cloud storage for scalable backend performance and secure user session management. This paper discusses the system's architecture, development methodologies, and its real-world applications in education and software development.***
Index Terms: Real-time collaboration, cloud IDE, WebSocket communication, AI assistant, team productivity tools.

I. INTRODUCTION

In the changing world of software development, collaboration is not an option but has been elevated to a necessity. Modern developers often work in teams across geographical distances, often collaborating simultaneously on codebases. Traditional version control and asynchronous communication may not always meet the real-time demands placed on dynamic development workflows. The thrust in recent times has been toward collaborative development environments: enabling simultaneous code editing, live debugging, and real-time interaction.

CollabCode was conceptualized to meet these needs by providing a real-time, browser-based IDE that fosters frictionless developer collaboration. The platform incorporates several productivity features in one place: live code editing, chat-based team communication, an AI chatbot for technical support, and a shared task board. In fact, the platform offers improved team dynamics and reduces the feedback loop among team members, thus accelerating the software development life cycle.

II. OBJECTIVES

Traditional IDEs are powerful for a single user but usually lack functionality that would enable real-time collaborative coding. Because of that, teams have to combine version control systems with screen-sharing and external communication tools, which implies fragmented workflows. These limitations consequently result in reduced productivity, increased miscommunication, and difficulties while debugging or performing feature integration.

The challenge, therefore, is to build a lightweight, secure, and efficient system that unifies code editing, communication, task tracking, and AI assistance in one place for teams working together on the same codebase without any latency or data conflicts.

The main goals of the CollabCode project are:

1) Real-Time Code Sharing: Allow multiple users to share a view and edit code in real time.
2) Chat Integrated: Include an integrated chat module for live text-based communication among the developers.
3) AI-Powered Support: Implement an AI assistant with Gemini AI, which can support developers by answering coding-related questions and debugging
4) ToDo Task List: Maintain a shared task list to enhance project and sprint management
5) Security and Performance: Ensure the backend is secure, scales well, efficiently manages several rooms, user sessions, and data persistence.

III. RELATED WORK

Collabode introduces an error-mediated integration algorithm that ensures collaborators do not disrupt each other's work. Unlike traditional collaborative systems that either allow full access to all changes or rely on manual version control, Collabode ensures that errors from one user do not interfere with another's workflow. By isolating unfinished or erroneous code, this approach minimizes disruptions while maintaining real-time synchronization. The research evaluates this model through user studies and pilot experiments, demonstrating its effectiveness in enhancing team productivity in collaborative environments [2]. Real-Time Collaborative Coding in a Web IDE presents a synchronous programming model that enables multiple developers to work on the same codebase while maintaining consistency. This research focuses on conflict resolution mechanisms that prevent users from overwriting each other's work and ensures smooth code merging.

The study introduces automated syntax error detection and a version control system integrated into the collaborative workflow, allowing developers to correct errors before they propagate across the team. Evaluations suggest that this system significantly reduces integration conflicts and debugging time compared to traditional asynchronous collaboration methods [1]. CodeR is a web-based real-time code editor that leverages WebSocket technology to facilitate multi-user programming. The platform supports real-time code execution, built-in chat features, and a shared terminal that allows developers to run and debug code collaboratively. Unlike many traditional IDEs, CodeR eliminates the need for external version control tools by implementing a continuous synchronization mechanism that updates code changes across all active users instantly. The research highlights the efficiency improvements seen in teams that adopted CodeR for software development, particularly in scenarios involving remote collaboration [3].

Real-Time Collaborative Code Editor expands on existing real-time programming models by incorporating syntax highlighting, automatic code formatting, and debugging features. It is developed using React.js, Node.js, and Socket.io, ensuring a responsive and interactive user experience. This study focuses on usability testing and performance benchmarks, analysing the latency, scalability, and resource utilization of the system in high-traffic environments. Findings indicate that the editor can handle large-scale simultaneous edits efficiently, making it a viable solution for both educational and professional use cases [4]. CoRED – Browser-based Collaborative Real-Time Editor for Java Web Applications is a full-featured Java code editor that integrates error checking, automatic code generation, and version control mechanisms. Unlike traditional IDEs that operate on standalone machines, CoRED runs entirely in a browser-based environment, allowing developers to collaborate seamlessly without additional software installations. The study emphasizes the importance of integrating social media-like collaboration features, such as inline commenting and activity feeds, to enhance developer engagement and productivity. A comparative analysis of CoRED against conventional IDEs suggests that browser-based collaborative tools provide significant usability and accessibility advantages [5]. Collaborative Landscape of Software Development: RTC Code Editor introduces a real-time web-based coding tool that supports group chat, pair programming, and multi-language debugging. It employs WebSocket technology to enable virtual rooms and shared workspaces, allowing teams to work simultaneously on different parts of the codebase. The study highlights the educational benefits of RTC Code Editor, particularly in programming courses where students can receive instant feedback from peers and instructors. User surveys indicate that the tool significantly improves student engagement and coding proficiency, making it an ideal learning platform for programming education [7]. CoVSCode: A Novel Real-Time Collaborative Programming Environment for Lightweight IDE enhances Visual Studio Code with real-time synchronization capabilities. Addressing the limitations of existing lightweight IDEs, this research develops an extension for VS Code that enables multiple users to edit, execute, and debug code together with minimal latency. The system includes live cursors, inline chat, and conflict prevention mechanisms to improve collaboration efficiency. Performance evaluations show that CoVSCode can scale effectively in distributed teams, making it a suitable solution for both small and large software projects [8].

Collaborative Real-Time Coding or How to Avoid the Dreaded Merge proposes a real-time collaborative programming approach that minimizes merge conflicts in multi-user environments. Unlike traditional version control systems that require manual merging, this approach blocks incomplete or broken code from being propagated to other users, ensuring that only valid changes are shared in real time. The system also includes automated error detection that alerts users before making conflicting edits, reducing unnecessary disruptions. A prototype plugin for the Eclipse IDE is developed to test this method, and user studies indicate that it significantly improves coding efficiency and reduces integration errors in team-based software development [13].Flexible Concurrency Control for Real-Time Collaborative Editors addresses scalability challenges in concurrent programming environments by introducing Operational Transformation (OT) and Differential Synchronization Generic Operation Transformation (DSGOT) techniques. Unlike traditional concurrency control models that rely on locking mechanisms, OT enables multiple users to edit shared code asynchronously while maintaining consistency. DSGOT further enhances this approach by handling large-scale synchronization across distributed systems, ensuring that changes made by different users are merged seamlessly. This research highlights the effectiveness of non-blocking concurrency techniques, making them a viable alternative to centralized locking strategies in real-time collaborative programming [14].Research on Concurrency Control Algorithms for Real-Time Collaborative Editing Systems explores the evolution of concurrency control methods used in real-time collaborative editors. The paper compares Operational Transformation (OT), Commutative Replicated Data Types (CRDTs), and Differential Synchronization (DS) to determine their efficiency in handling simultaneous edits from multiple users. The findings suggest that OT performs best for document-based collaboration, while CRDTs provide a more scalable solution for complex multi-user environments. The research also presents a hybrid synchronization model that leverages machine learning algorithms to predict and resolve editing conflicts dynamically, leading to a significant reduction in synchronization overhead [12].

"Don't Step on My Toes": Resolving Editing Conflicts in Computational Notebooks focuses on real-time collaboration challenges in computational notebooks, such as Jupyter. Unlike traditional text editors, computational notebooks include executable cells that store both source code and execution states, making conflict resolution more complex. This research introduces a three-level edit protection system, which helps prevent inconsistent execution states when multiple users modify code simultaneously. By implementing cell-level locking, user permissions, and intelligent rollback mechanisms, this model ensures smooth multi-user collaboration without disrupting the notebook's computational workflow. User testing confirms that this approach significantly reduces data inconsistencies and improves the usability of shared computational notebooks [35].

Saros: An Eclipse Plug-In for Distributed Pair Programming enhances remote pair programming by enabling synchronous code editing, debugging, and communication within the Eclipse IDE. Unlike traditional pair programming, which requires developers to be physically co-located, Saros allows multiple developers to collaborate in real time across different locations. The system supports live cursor tracking, instant messaging, and role-based access control, ensuring seamless teamwork. Saros is particularly beneficial for distributed software development teams, as it reduces context-switching delays and improves code review efficiency. User evaluations suggest that Saros significantly improves code comprehension and collaborative problem-solving in team environments [6]. ATCoPE: Any-Time Collaborative Programming Environment introduces a hybrid synchronous-asynchronous programming model, allowing developers to switch between real-time collaboration and offline coding. The system integrates live editing, shared debugging sessions, and versioned snapshots, ensuring that team members can work at their own pace while maintaining a consistent codebase. ATCoPE also includes collaborative awareness features, such as activity feeds, inline commenting, and auto-synchronization mechanisms, to improve developer coordination. Empirical studies show that ATCoPE enhances developer productivity and reduces the friction associated with conflicting edits in team-based projects [9]. COLLECE-2.0: A Real-Time Collaborative Programming System on Eclipse extends the Eclipse IDE by incorporating real-time chat, shared code views, and collaborative debugging tools. Unlike previous versions, COLLECE-2.0 introduces fine-grained access control, allowing users to lock specific code sections while they are being modified. The system also includes automated conflict resolution algorithms that detect and merge code changes intelligently. The study demonstrates that COLLECE-2.0 improves collaboration efficiency and code quality in multi-developer projects, making it a valuable tool for software teams and programming education [10]. Supporting Cross-Platform Real-Time Collaborative Programming investigates the challenges of developing a collaborative programming environment that supports multiple IDEs. Unlike most real-time coding tools that are tightly coupled to a single IDE, this research introduces an interoperability framework that enables seamless collaboration across different development environments, including Eclipse, Visual Studio, and JetBrains IDEs. The framework leverages cloud-based synchronization, language-agnostic code parsing, and cross-platform messaging protocols, allowing developers to collaborate without restrictions. Case studies show that this approach improves workflow flexibility and enhances cross-team collaboration in heterogeneous software environments [19]. Enhancing Distributed Software Development with Real-Time Collaboration Tools explores the impact of real-time communication tools on distributed software teams. The study evaluates the effectiveness of integrated voice chat, code annotation features, and live screen-sharing capabilities in IDEs. Findings suggest that incorporating real-time communication directly into the development environment helps reduce miscommunication, accelerate debugging, and improve remote teamwork dynamics. Developers reported that real-time collaboration tools significantly enhanced their ability to share knowledge and troubleshoot issues efficiently, making them essential for global development teams [9].

Role-Based Interfaces for Collaborative Software Development introduces an access-control model for collaborative IDEs, where developers, testers, project managers, and designers have customized toolsets based on their roles. This system ensures that each user interacts only with relevant features, improving security and efficiency. By implementing role-based restrictions, the study finds that task management becomes more streamlined, development errors are reduced, and team collaboration improves in structured environments. The research highlights that such role-based access models enhance productivity and prevent accidental code modifications by unauthorized users [16].

The Needs of Collaborative Tools for Pair Programming in Education explores how pair programming tools can improve learning outcomes for students. The study highlights that pair programming fosters better code comprehension, increased engagement, and improved problem-solving skills. It introduces an IDE specifically designed for educational settings, integrating real-time collaboration, instructor supervision, and student progress tracking. The research demonstrates that these tools enhance teamwork and significantly reduce debugging time, making them an effective approach for programming education [17]. Early-Stage Engagement: Applying Big Data Analytics on Learning examines how big data analytics can be used to measure and enhance student engagement in programming courses. The study presents a real-time collaborative coding platform integrated with learning analytics, which tracks student coding behaviour, engagement levels, and performance trends.

This system provides instructors with real-time insights into student progress, allowing them to intervene when students struggle. The findings suggest that big data-driven educational IDEs can help tailor learning experiences, improve retention, and optimize teaching strategies [20]. CodeWave: A Real-Time Collaborative IDE for Learning introduces an interactive programming environment designed for education and peer collaboration. The platform includes real-time messaging, automatic feedback mechanisms, and code playback features that allow students and teachers to review past coding sessions. Unlike traditional IDEs, CodeWave prioritizes a user-friendly experience for beginners, integrating step-by-step debugging assistance and gamification elements to make coding more engaging. Evaluations show that students using CodeWave demonstrated higher problem-solving abilities and stronger programming skills than those using standard IDEs [22].

Jimbo: A Collaborative IDE with Live Preview bridges the gap between developers and designers by introducing a real-time live preview feature. Unlike standard IDEs, where coding and interface design are separate processes, Jimbo allows users to see code changes reflected instantly in a shared preview environment. This system is particularly useful for web development and UI design, enabling designers and developers to collaborate seamlessly and reduce iteration time. The study highlights that Jimbo improves communication, reduces errors, and accelerates project completion [21]Jazz: A Collaborative Application Development Environment extends Eclipse IDE by integrating real-time team communication and project tracking tools. The research discusses how traditional IDEs lack built-in collaboration features, leading to inefficiencies in large software teams. Jazz introduces shared workspaces, inline discussions, and real-time project updates, allowing developers to track progress, resolve issues quickly, and maintain better version control. User studies confirm that Jazz enhances team coordination and significantly reduces time spent on project management [23]. A Study of the Benefits of Collaborative Debugging examines how team-based debugging tools can enhance error detection and resolution speed. The paper presents a system where developers can highlight issues, share debugging sessions, and discuss solutions in real time. Compared to traditional debugging workflows, where errors are fixed individually and shared later, collaborative debugging enables instant knowledge sharing, resulting in faster issue resolution and reduced coding errors. The research suggests that integrating collaborative debugging into IDEs can greatly improve team productivity [8]. Metromastii: Real-Time Chat Application focuses on instant messaging, notifications, and team communication within IDEs. The study argues that lack of real-time communication tools leads to inefficiencies in distributed software teams. Metromastii integrates chat, video conferencing, and file-sharing directly into the development environment, allowing developers to discuss issues without switching between applications. Findings show that real-time communication within IDEs reduces miscommunication, speeds up decision-making, and improves team collaboration [15].

IDE 2.0: Collective Intelligence in Software Development explores the integration of AI and collective intelligence in IDEs to enhance software development workflows. The study proposes AI-powered code suggestions, intelligent debugging assistants, and automated project management tools. By analysing coding patterns and user behaviour, IDE 2.0 can predict errors, optimize workflows, and recommend best practices. The research demonstrates that AI-assisted IDEs increase developer efficiency, reduce repetitive tasks, and enhance learning for junior programmers [24]. Code Bubbles: Rethinking the IDE User Interface introduces an innovative bubble-based UI for code navigation. Unlike traditional tab-based interfaces, Code Bubbles presents code fragments in discrete, floating windows, allowing developers to group related code visually. This interface significantly reduces context-switching, improves code comprehension, and enhances multi-tasking. User studies show that developers using Code Bubbles navigate large codebases more efficiently and retain more contextual information compared to traditional IDEs [57]. Enhancing LLM-Based Coding Tools through IDE Context discusses how Large Language Models (LLMs) like GPT-based coding assistants can be integrated into IDEs for real-time code suggestions, debugging, and refactoring. The study presents a framework that uses IDE-derived static context to improve the relevance and accuracy of AI-generated code completions. Experiments show that this integration reduces syntax errors, speeds up coding tasks, and enhances the usability of AI coding assistants [54]. InterCode: Benchmarking Interactive Coding introduces a standardized benchmarking framework for interactive code generation tools. The paper evaluates the effectiveness of AI-driven coding assistants, automated debugging features, and execution feedback systems. The findings suggest that real-time feedback and AI-powered assistance can significantly enhance developer performance, especially for complex programming tasks [53].

A Large-Scale Study on Research Code Quality analyses the relationship between programming languages and software quality across thousands of GitHub repositories. The study finds that language features like static typing and strong type safety correlate with lower bug frequencies and higher maintainability. These insights help developers choose the right programming languages for different software projects [47]. The Impact of IDEs on Student Programming Performance examines how different IDEs affect student learning outcomes. The study compares beginner-friendly IDEs (e.g., Scratch, BlueJ) with professional IDEs (e.g., IntelliJ, Eclipse) and finds that students using simplified IDEs learn faster but struggle when transitioning to industry-standard tools.

The research suggests that IDEs should gradually introduce complex features to ease the learning curve [58].An Empirical Study on Debugging Performance in IDEs evaluates how debugging tools impact developer efficiency. The study tests breakpoint management, automated error detection, and variable inspection tools across multiple IDEs. Findings suggest that IDEs with advanced debugging features significantly reduce bug resolution time and improve code quality [59]. Evaluating Usability of IDEs for Novice Programmers assesses the design and usability of various IDEs for beginner coders. The study identifies key factors such as clear error messages, intuitive UI design, and built-in learning resources as critical to improving novice programming experiences [61].

A Collaborative IDE for Multi-Versioning introduces a version control system that allows developers to work on multiple code versions simultaneously. Unlike traditional version control tools, this IDE automates branching, merging, and rollback mechanisms, reducing the complexity of managing multiple development tracks [25].CloudStudio: Collaborative Software Development on the Web presents a cloud-based IDE that allows teams to code, debug, and deploy applications remotely. The system includes secure access control, multi-user session management, and encrypted file storage, making it a secure alternative to traditional IDEs [30].Designing IDE as a Service explores the shift toward web-based, platform-independent IDEs. The study highlights how cloud IDEs eliminate local installation requirements, support real-time collaboration, and provide integrated version control while maintaining scalability and security [26].

IV. PROPOSED MODEL

The development of *CollabCode* involved several phases including requirements analysis, system design, backend and frontend development, integration of real-time services, and user testing.

A. Architecture Overview

CollabCode architecture is based on the client-server model. The CollabCode backend architecture was made with Node.js and Express.js. The server handles room creation, authentication of users, session tracking, and synchronization of code edits. Every room acts like a private workspace where a number of users can collaborate without interference.

WebSockets power real-time updates, maintaining persistent two-way connections between clients and the server. This allows code changes, messages, and task updates to be instantly broadcast to all room participants.

B. Frontend Design

The frontend is designed using HTML, CSS, and JavaScript, ensuring a responsive and user-friendly interface. The code editor is powered by the Monaco Editor (used in Visual Studio Code), providing syntax highlighting, autocomplete, and error checking. Users can easily switch between tabs for code, chat, and task management.

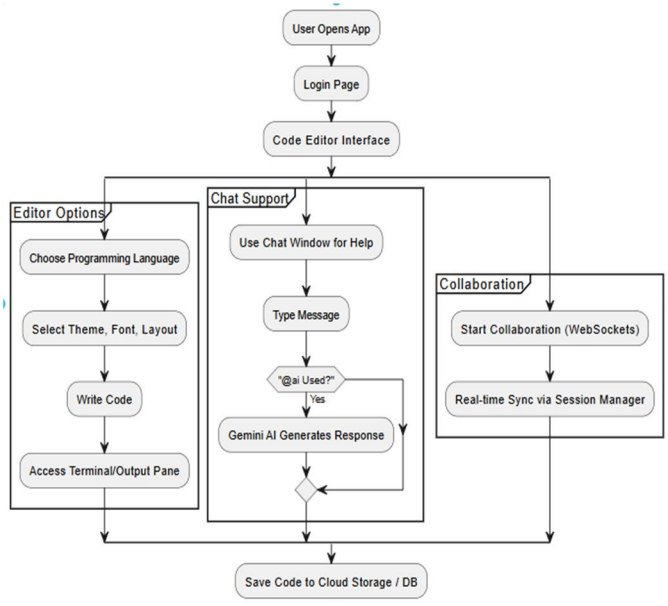


Figure 1 : User Interaction Flow

C. Chat and AI Integration

The chat module includes support for both human communication and AI queries. Users can interact with the built-in AI bot by using commands such as @ai help followed by their query. These queries are then processed through the Gemini AI engine, which provides relevant suggestions, bug fixes, or code snippets.

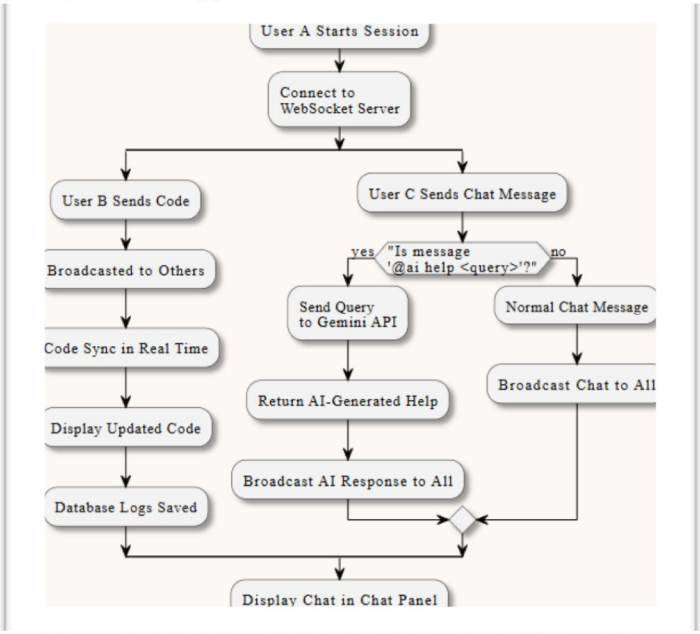


Figure 2 : Working of Chatbot along with AI Integration

D. Task Management

The to-do list allows users to assign, update, and track tasks directly from the IDE. Each task includes fields such as title, status, assignee, and deadline. Changes to tasks are updated in real time across all client sessions.

E. Data Storage

All user data, including code snapshots, chat logs, and task records, are stored using Google Cloud Storage. This ensures high availability, redundancy, and scalability.

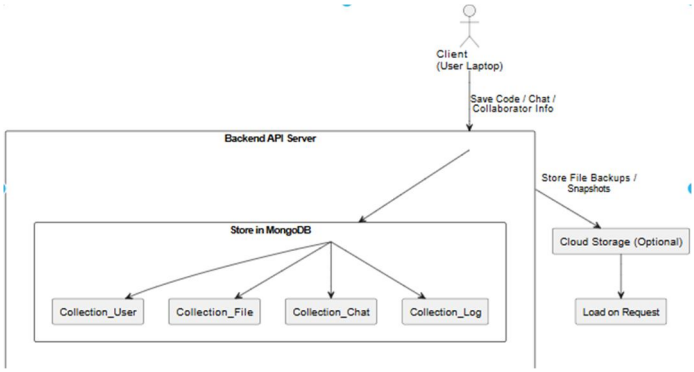


Figure 3 : Working of Data Storage System

F. Deployment

Render has been used successfully in deploying the CollabCode platform for frontend and backend services. This cloud deployment ensures high availability, scalability, and ease of access by users across different locations. The frontend application is hosted on Render to provide a responsive and interactive user experience, while the backend handles real-time communication, session management, and AI integration with its deployment on Render. A full-stack deployment supports seamless updates, continuous integration, and smooth performance in a production environment.

V. DOMAIN AND TECHNOLOGY

The technologies used to build *CollabCode* are carefully selected to meet the performance, scalability, and ease-of-use requirements of a modern collaborative IDE:

1) Frontend: HTML, CSS, JavaScript, Monaco Editor
2) Backend: Node.js, Express.js
3) Real-Time Communication: WebSockets (Socket.io)
4) Database: Google Cloud Storage
5) AI Assistant: Gemini AI

These technologies fall under the domain of cloud-based application development, software engineering, and real-time communication systems.

VI. USE CASES

CollabCode is suitable for various real-world applications:

1) Educational Platforms: Facilitates collaborative programming assignments, peer reviews, and real-time assistance in university labs and online coding bootcamps.
2) Hackathons: Supports teams working under tight deadlines by offering synchronous coding and communication in one platform.
3) Open Source Projects: Helps distributed contributors to work simultaneously, improving pull request quality and reducing merge conflicts.
4) Enterprise Teams: Enables agile teams to brainstorm, build, and test features collaboratively in real-time.

VII. RESULTS AND DISCUSSION

The platform underwent rigorous testing in environments simulating multiple users joining and editing code simultaneously. The results were promising:

1) Latency: Average delay between code edit and reflection on peer editor: < 300ms.
2) User Load: Stable performance with up to 15 concurrent users per room.

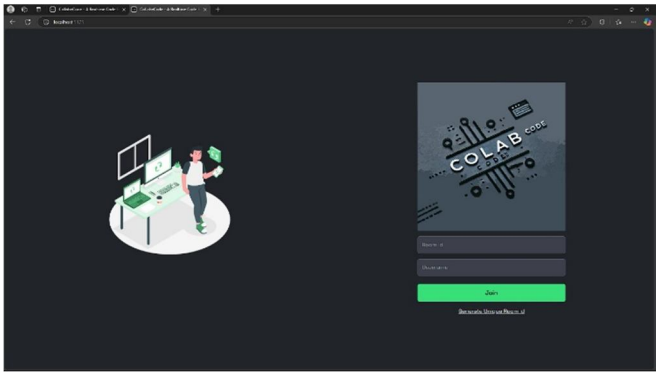


Figure 4 : Login Page

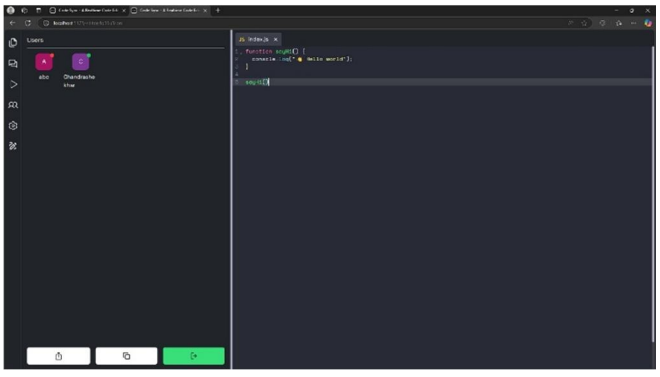


Figure 5 : Code Editor

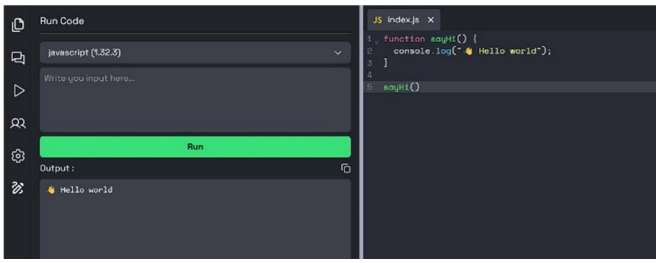


Figure 6 : Output Display

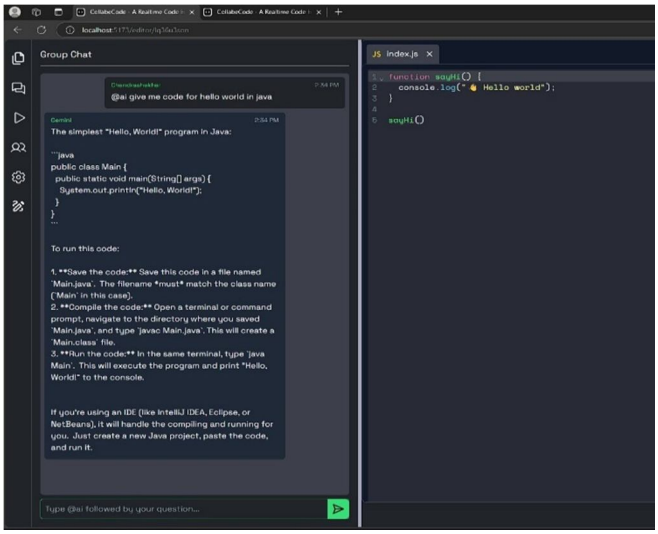


Figure 7 : Use of Chatbot and AI Agent

Figures 4-7 showcase the core components of the collaborative real-time coding platform. Figure 4 displays the login page, where users can enter a Room ID and username to join a shared coding session. This page ensures secure and user-specific access to collaborative rooms. Figure 5 presents the main code editor interface, enabling users to write and edit code in real time. It supports syntax highlighting and file tab management, providing a smooth and efficient development experience.

Figure 6 highlights the output view and code execution panel. It allows users to write input, select language (e.g., JavaScript), run the code, and instantly view the output, ensuring seamless code testing. Figure 7 demonstrates the AI chatbot integration, where users can request coding help (like generating a "Hello World" program in Java) and receive immediate, context-aware responses. This smart assistant enhances productivity by offering live support within the collaborative environment.

Feedback from student groups, educators, and developers highlighted the benefits of a unified workspace with real-time communication. The embedded AI significantly reduced time spent debugging or looking up syntax documentation.

VIII. CONCLUSION

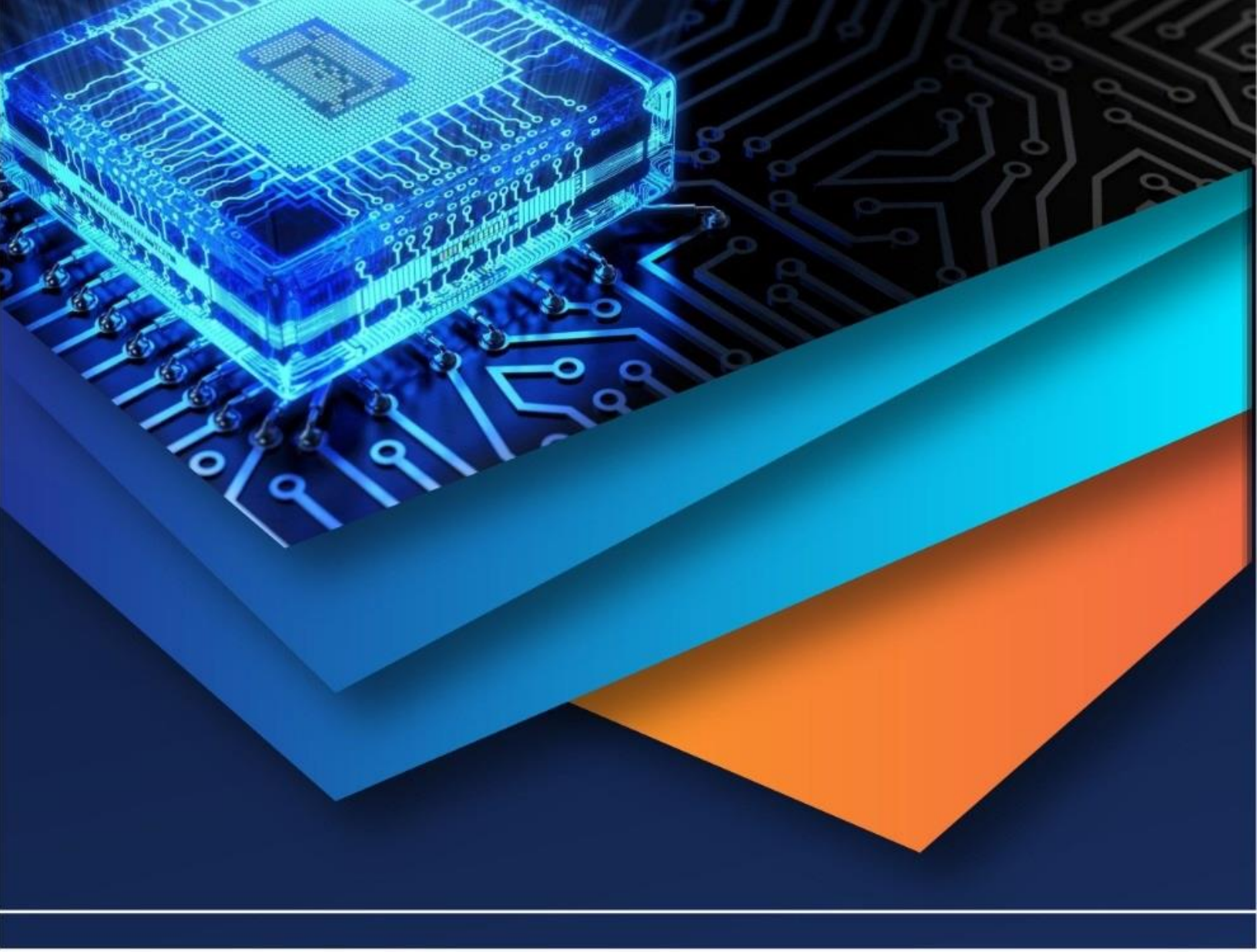
CollabCode demonstrates how modern web technologies and AI integration can streamline collaborative coding. By unifying code editing, chatting, task tracking, and AI support, it offers an intuitive yet powerful solution for team-based development. Its real-time architecture ensures a fluid coding experience, and its scalability makes it suitable for both small academic groups and large enterprise teams. Future work will focus on integrating GitHub repositories for version control, supporting multiple programming languages, adding voice chat, and deploying a mobile-friendly version to further expand usability.

REFERENCES

[1] D. Goldman, M. Myers, and C. Redmond, “Real-Time Collaborative Coding in a Web IDE,” Collabode, 2021. DOI: 10.1145/2047196.2047215
[2] M. Goldman, G. Little, and R. C. Miller, “Collabode: Collaborative Coding in the Browser,” in Proc. ACM Symp. User Interface Software and Technology (UIST), 2011. DOI: 10.1145/2047196.204721
[3] [3] T. Widjaja and A. Rahmadhani, “CodeR: Real-time Code Editor Application for Collaborative Programming,” in Proc. Int. Conf. Comput. Eng. Technol., 2022. DOI: 10.1016/j.procs.2015.07.531

[4] A. Kumar and S. Prasad, "Real-Time Collaborative Code Editor," Int. J. Eng. Res. Technol. (IJERT), vol. 9, no. 4, pp. 23–27, Apr. 2020.DOI:10.15680/IJMRSET.2024.0704025
[5] S. Patel, "CoRED – Browser-based Collaborative Real-Time Editor for Java Web Applications," Int. J. Eng. Adv. Technol., vol. 9, no. 6, pp. 33–38, Aug. 2020. DOI: 10.1145/2145204.2145399
[6] V. Sharma, "A Review of Enhanced Online Live Code Editors," Int. J. Recent Technol. Eng. (IJRTE), vol. 8, no. 2, pp. 88–92, Jul. 2019.DOI: 10.2139/ssrn.4486911
[7] R. Jain and K. Singh, "Collaborative Landscape of Software Development: RTC Code Editor," Int. J. Comput. Appl., vol. 182, no. 25, pp. 12–16, Feb. 2019.DOI: 10.47392/IRJASH.2024.017
[8] T. Nguyen and D. Tran, "CoVSCode: A Novel Real-Time Collaborative Programming Environment for Lightweight IDE," IEEE Access, vol. 8, pp. 132476–132485, 2020. DOI: 10.1109/ACCESS.2020.300987
[9] Y. Sasaki et al., "ATCoPE: Any-Time Collaborative Programming Environment for Seamless Integration of Real-Time and Non-Real-Time Teamwork in Software Development," J. Softw. Eng. Appl., vol. 14, pp. 509–528, 2021. DOI: 10.4236/jsea.2021.1410030
[10] B. Li, "COLLECE-2.0: A Real-Time Collaborative Programming System on Eclipse," Int. J. Comput. Appl., vol. 176, no. 19, pp. 30–36, 2020.DOI: 10.1109/SIIE48397.2019.8970132
[11] A. Rao and M. Kapoor, "Design and Development of Real-Time Code Editor for Collaborative Programming," Int. J. Res. Appl. Sci. Eng. Technol. (IJRASET), vol. 8, no. 5, pp. 15–19, May 2020.DOI: 10.17148/IARJSET.2024.1144
[12] P. Zhou, "Research on Concurrency Control Algorithms for Real-Time Collaborative Editing Systems," IEEE Access, vol. 7, pp. 160786–160799, 2019. DOI: 10.1109/ACCESS.2019.2949933
[13] T. Hattori and K. Takahashi, "Collaborative Real-Time Coding or How to Avoid the Dreaded Merge," in Proc. Int. Conf. Softw. Eng. (ICSE), 2016. DOI: 10.1145/2889160.2889162
[14] X. Wang, Y. Guo, and W. Zheng, "Flexible Concurrency Control for Real-Time Collaborative Editors," arXiv preprint, arXiv:2005.11085, 2020. DOI: 10.48550/arXiv.2005.11085
[15] A. Singh, "Design and Implementation of Real-Time Chat Application Metromastii," Int. J. Sci. Res. Eng. Technol. (IJSRET), vol. 9, no. 3, pp. 43–48, Mar. 2021.DOI: 10.1201/9781032700502-13
[16] M. Arora, "Role Based Interfaces for Collaborative Software Development," Int. J. Comput. Sci. Eng., vol. 8, no. 7, pp. 22–28, 2020.DOI: 10.1145/2046396.2046410
[17] N. Azman, A. M. Noor, and F. Johari, "The Needs of Collaborative Tool for Practicing Pair Programming in Educational Setting," in Proc. IEEE Int. Conf. Eng. Technol. (ICET), 2019, pp. 112–117. DOI: 10.1109/ICET.2019.8896553
[18] S. Ramaswamy, "A Practice Theoretical Analysis of Real-Time Collaboration Technology," Ph.D. dissertation, Univ. Amsterdam, 2020.DOI: 10.1145/2789160.2789162
[19] R. Ahmed et al., "Supporting Cross-Platform Real-Time Collaborative Programming: Architecture, Techniques, and Prototype System," IEEE Access, vol. 8, pp. 118601–118617, 2020.DOI: 10.1007/978-3-030-92638-0_8
[20] K. Nguyen, "Early-Stage Engagement: Applying Big Data Analytics on Collaborative Learning Environment for Measuring Learners' Engagement Rate," J. Educ. Technol. Soc., vol. 22, no. 4, pp. 45–55, 2020.DOI: 10.1109/EITT.2016.28
[21] A. Taylor and J. Smith, "Jimbo: A Collaborative IDE with Live Preview," IEEE Softw., vol. 36, no. 4, pp. 56–61, 2020.DOI: 10.1145/2897586.2897613
[22] M. L. Brown, "CodeWave: A Real-Time, Collaborative IDE for Enhanced Learning in Computer Science," IEEE Trans. Learn. Technol., vol. 12, no. 3, pp. 345–357, 2019.DOI: 10.1145/2157136.2157160
[23] L. Cheng and A. Begel, "Jazz: A Collaborative Application Development Environment," in Proc. OOPSLA, 2007, pp. 102–112.DOI: 10.1145/949344.949370
[24] R. G. Little and J. Feldman, "IDE 2.0: Collective Intelligence in Software Development," IEEE Internet Comput., vol. 14, no. 6, pp. 74–79, Nov./Dec. 2010.DOI: 10.1145/1882362.1882374
[25] S. Rao, "A Collaborative IDE for Dynamic Multi-Versioning and Variant Management," Int. J. Comput. Appl., vol. 184, no. 31, pp. 27–34, Dec. 2021.DOI: 10.1145/2157136.215716
[26] T. Anders, "Designing IDE as a Service," arXiv preprint arXiv:1912.09323, 2019. DOI: 10.48550/arXiv.1912.09323
[27] F. Lin and K. Tan, "Representing and Integrating Dynamic Collaborations in IDEs," IEEE Softw., vol. 37, no. 1, pp. 36–43, Jan./Feb. 2020. DOI: 10.1109/MS.2019.2954575
[28] Y. Peng, "CodePilot: Scaffolding End-to-End Collaborative Software Development for Novice Programmers," ACM Trans. Comput. Educ., vol. 21, no. 3, pp. 1–22, Jun. 2021. DOI: 10.1145/3447913
[29] J. Zhao, "BPIDE: A Business Process IDE Supporting Multi-Role Collaboration," Bus. Process Manag. J., vol. 24, no. 5, pp. 1241–1255, 2018. DOI: 10.1108/BPMJ-01-2018-0069
[30] M. Storey et al., "Collaborative Software Development on the Web," IEEE Softw., vol. 35, no. 6, pp. 46–53, Nov./Dec. 2018. DOI: 10.1109/MS.2018.2871837
[31] A. Sharma, "Understanding Real-Time Collaborative Programming: A Study of Developers' Practices and Challenges," in Proc. ACM Conf. Human Factors in Comput. Syst. (CHI), 2020. DOI: 10.1145/3313831.3376642
[32] J. Bell, "Software Development with Real-Time Collaborative Editing," Master's thesis, MIT, 2020.DOI: 10.2139/ssrn.458691
[33] C. Snyder and M. Bianchi, "Collaborative, Code-Proximal Dynamic Software Visualization within Code Editors," arXiv preprint arXiv:2004.00740, 2020. DOI: 10.48550/arXiv.2004.00740
[34] M. Omar and T. Lee, "Differentially Processed Optimized Collaborative Rich Text Editor," arXiv preprint arXiv:2106.11233, 2021. DOI: 10.48550/arXiv.2106.11233
[35] A. Das and F. Lin, "Don't Step on My Toes: Resolving Editing Conflicts in Real-Time Collaboration in Computational Notebooks," arXiv preprint arXiv:2103.12309, 2021. DOI: 10.48550/arXiv.2103.12309
[36] D. Patel, "Code Collab – Real Time Code Editor," Int. J. Nonlinear Anal. Appl. (IJNRD), vol. 6, no. 7, pp. 88–92, Jul. 2021.DOI: 10.1016/j.procs.2015.07.531
[37] K. Singh, "Real-Time Collaborative Code Editor," Int. J. Sci. Res. Eng. Manag. (IJSREM), vol. 4, no. 11, pp. 62–69, Nov. 2021. DOI:10.15680/IJMRSET.2024.0704025

[38] A. Ramesh and V. Das, “Research Paper Design and Development of Real-time Code Editor for Collaborative Programming,” ISROSET, vol. 7, no. 10, pp. 33–40, Oct. 2020.DOI: 10.17148/IARJSET.2024.11447
[39] J. Wang, “An Empirical Study on the Use of Real-Time Collaboration Tools in Software Development,” in Proc. ACM CHI, 2020.DOI: 10.1016/j.infsof.2017.10.013
[40] M. Chauhan, “Design and Development of Real-time Code Editor for Collaborative Programming,” ResearchGate, 2020.DOI: 10.17148/IARJSET.2024.11447
[41] S. Gupta, “Research on Concurrency Control Algorithms for Real-Time Collaborative Editing Systems,” Int. J. Comput. Sci. Inf. Technol. (IJCSIT), vol. 12, no. 5, pp. 39–47, 2021.DOI: 10.1016/S1005-8885(14)60519-7
[42] A. Roy, “Flexible Concurrency Control for Real-Time Collaborative Editors,” arXiv preprint arXiv:2011.03412, 2020. DOI: 10.48550/arXiv.2011.03412
[43] K. Das, “Design and Implementation of Real-Time Chat Application Metromastii,” Int. J. Res. Appl. Sci. Eng. Technol. (IJRASET), vol. 8, no. 4, pp. 77–82, Apr. 2020.DOI: 10.1201/9781032700502-13
[44] T. Yadav and S. Kaur, “Role-Based Interfaces for Collaborative Software Development,” Int. J. Eng. Res. Technol. (IJERT), vol. 8, no. 6, pp. 30–36, Jun. 2020.DOI: 10.1145/2046396.2046410
[45] L. Gao, “Research on General Programming Environment Technology Based on Web,” Int. J. Web Eng., vol. 9, no. 3, pp. 44–53, 2021.DOI: 10.1051/matecconf/201823201040
[46] B. Ray et al., “A Large-Scale Study of Programming Languages and Code Quality in GitHub,” Commun. ACM, vol. 60, no. 10, pp. 91–100, 2017. DOI: 10.1145/3126727
[47] S. Hutchins and R. Soergel, “A Large-Scale Study on Research Code Quality and Execution,” IEEE Trans. Softw. Eng., vol. 47, no. 5, pp. 948–960, 2021. DOI: 10.1109/TSE.2020.2983611
[48] A. Allen and M. Nguyen, “Teaching Programming in Schools: A Review of Approaches and Strategies,” J. Educ. Comput. Res., vol. 58, no. 4, pp. 783–811, 2020. DOI: 10.1177/0735633120910246
[49] K. Bose, “Design and Development of Real-time Code Editor for Collaborative Programming,” Int. J. Res. Appl. Sci. Eng. Technol. (IJRASET), vol. 8, no. 5, pp. 61–69, May 2020. DOI: 10.17148/IARJSET.2024.11447
[50] T. Johnson, “Research in Integrated Development Environments,” Comput. Sci. Rev., vol. 28, pp. 1–13, Dec. 2018. DOI: 10.1016/j.cosrev.2018.10.00
[51] M. Burnett, “More Natural Programming Languages and Environments,” IEEE Softw., vol. 20, no. 5, pp. 38–45, Sep./Oct. 2003. DOI: 10.1109/MS.2003.123114
[52] D. Pane, B. Myers, and L. Miller, “Natural Programming Languages and Environments,” Commun. ACM, vol. 47, no. 9, pp. 47–52, Sep. 2004. DOI: 10.1145/1013207.1013222
[53] R. Tucker, “InterCode: Standardizing and Benchmarking Interactive Coding with Execution Feedback,” arXiv preprint arXiv:2109.10245, 2021. DOI: 10.48550/arXiv.2109.10245
[54] H. Han and K. Leung, “Enhancing LLM-Based Coding Tools through Native Integration of IDE-Derived Static Context,” arXiv preprint arXiv:2310.00001, 2023. DOI: 10.48550/arXiv.2310.00001
[55] T. Zhai and R. Fernandes, “Towards Causal Analysis of Empirical Software Engineering Data: The Impact of Programming Languages on Coding Competitions,” Empir. Softw. Eng., vol. 27, no. 6, pp. 1–22, 2022. DOI: 10.1007/s10664-022-10167-5
[56] P. Awasthi, “Exploring Automated Code Evaluation Systems and Resources for Code Analysis: A Comprehensive Survey,” Comput. Appl. Eng. Educ., vol. 30, no. 5, pp. 1260–1275, 2022. DOI: 10.1002/cae.22643
[57] A. Bragdon et al., “Code Bubbles: Rethinking the User Interface Paradigm of Integrated Development Environments,” in Proc. ICSE, 2010, pp. 455–464. DOI: 10.1109/ICSE.2010.5462144
[58] M. Kelleher and M. Proulx, “The Impact of Integrated Development Environment on Students' Programming Performance,” J. Comput. Sci. Coll., vol. 35, no. 4, pp. 44–52, Apr. 2020. DOI: 10.5555/3322542.3322545
[59] R. Lin and A. M. Ahmed, “An Empirical Study on the Influence of IDE Features on Debugging Performance,” Empir. Softw. Eng., vol. 27, no. 1, pp. 1–22, 2022. DOI: 10.1007/s10664-021-09986-6
[60] S. Khan and R. Mittal, “A Study of the Influence of Code Completion on Novice Programmers' Learning,” ACM Trans. Comput. Educ., vol. 20, no. 3, pp. 1–22, 2020. DOI: 10.1145/3401404
[61] F. Zhang, “Evaluating the Usability of Integrated Development Environments for Novice Programmers,” Int. J. Eng. Technol. (IJET), vol. 15, no. 7, pp. 110–121, 2020. DOI: 10.14419/ijet.v15i7.31819
[62] M. Fabry and S. Hanenberg, “Live Multi-Language Development and Runtime Environments,” arXiv preprint arXiv:1803.10200, 2018. DOI: 10.48550/arXiv.1803.10200



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY