e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:08/Issue:02/February-2026 Impact Factor- 8.187 www.irjmets.com

A COMPREHENSIVE REVIEW OF REAL-TIME COLLABORATIVE CODING PLATFORMS WITH ROLE-BASED ACCESS CONTROL

Vaibhav G. Sayam[*1], Kshitij P. Lokhande[*2], Vidhan G. Rathod[*3], Deepak K. Bissa[*4], Tejas R. Golhar[*5], Kunal S. Tale[*6], Prof. Aakansha Shukla[*7]

[*1,2,3,4,5,6]Department Of Computer Engineering, B.E. Student, Jagadambha College Of Engineering And Technology, Yavatmal. India.

[*7]Department Of Computer Engineering, Assistant Professor, Jagadambha College Of Engineering And Technology, Yavatmal. India.

Sant Gadage Baba Amravati University, Amravati, India.

DOI: https://www.doi.org/10.56726/IRJMETS89618

ABSTRACT

The rapid growth of distributed software development, remote education, and cloud computing has significantly increased the demand for real-time collaborative programming environments. Conventional development practices, such as offline file sharing and version control systems, fail to provide instant synchronization and seamless teamwork, often leading to merge conflicts, delayed updates, and reduced productivity. In response, numerous web-based collaborative coding platforms and cloud-integrated development environments have emerged to support simultaneous code editing and remote cooperation.

This paper presents a comprehensive review and comparative analysis of existing real-time collaborative coding systems with particular emphasis on synchronization mechanisms, browser-based accessibility, scalability, and access control strategies. Prominent platforms including Visual Studio Code Live Share, Replit, CodePen, and GitHub Codespaces are critically examined in terms of architectural design, communication protocols, usability, and security features. The review identifies key technical challenges such as network latency, concurrent editing conflicts, resource overhead, and the absence of structured role-based supervision.

Furthermore, research gaps in lightweight deployment, centralized monitoring, and secure room-level collaboration are highlighted. Based on these findings, the study outlines future research directions toward the development of efficient, scalable, and role-managed collaborative coding frameworks. The insights provided in this review serve as a foundation for designing next-generation cloud-based programming environments for both educational and professional applications.

Keywords: Collaborative Coding, Real-Time Synchronization, WebSockets, Cloud-Based IDE, Socket.IO, Role-Based Access Control, Distributed Software Development, Online Programming Environments.

I. INTRODUCTION

The rapid advancement of cloud computing and internet technologies has transformed the way software development is performed across the world. Modern development practices increasingly involve distributed teams, remote collaboration, and online learning environments,

where multiple programmers contribute simultaneously to the same project. However, traditional collaboration methods such as offline file sharing and version control systems are not fully suitable for real-time teamwork. These approaches often introduce synchronization delays, merge conflicts, and communication overhead, which negatively affect productivity and learning efficiency.

To overcome these limitations, several real-time collaborative coding platforms and cloud-based Integrated Development Environments (IDEs) have been introduced. Tools such as Visual Studio Code Live Share, Replit, and CodePen enable shared editing and remote access to development environments. Although these systems provide partial solutions, they frequently suffer from issues including complex installation procedures, limited scalability, lack of structured supervision, and insufficient access control mechanisms. In educational and team-based settings, the absence of role-based management makes it difficult for instructors or project leaders to

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:08/Issue:02/February-2026 Impact Factor- 8.187 www.irjmets.com

monitor individual contributions effectively.

Real-time synchronization, secure access control, and lightweight browser-based usability remain critical requirements for modern collaborative programming systems. An ideal platform should allow multiple users to edit code concurrently with minimal latency while also providing controlled permissions, monitoring capabilities, and easy deployment without additional configuration. Despite ongoing research and development, many existing solutions do not fully integrate all these features within a single, efficient framework.

This paper presents a comprehensive review of current real-time collaborative coding technologies and analyzes their architectural approaches, communication mechanisms, advantages, and limitations. Various platforms and research contributions are systematically compared to identify common challenges and research gaps. Furthermore, the study discusses the design direction of a lightweight collaborative coding framework that incorporates WebSocket-based synchronization, browser-based accessibility, and role-based access control to support structured teamwork. The objective of this review is to provide insights into existing solutions and guide the development of more secure, scalable, and user-friendly, collaborative programming environments for both academic and professional applications. A WebSocket-based collaborative architecture with structured role-based access is considered to enhance synchronization efficiency and supervision capabilities.

II. LITERATURE SURVEY

The rapid growth of distributed software development and remote learning has motivated extensive research in real-time collaborative programming environments. Over the past decade, numerous platforms and cloud-based integrated development environments (IDEs) have been introduced to enable multiple developers to simultaneously access, edit, and execute shared code. These systems aim to improve productivity, reduce communication delays, and support geographically separated teams. This section reviews prominent collaborative coding tools and research contributions, highlighting their methodologies, advantages, and limitations.

One of the widely adopted solutions is **Visual Studio Code Live Share**, which enables real-time shared editing directly within the Visual Studio Code environment. The platform provides synchronized editing, debugging, and terminal sharing through peer-to-peer communication. Studies report that Live Share offers low latency and efficient collaboration for professional developers. However, it requires software installation, extensions, and configuration, which may limit accessibility for beginners and users with low-resource systems. Furthermore, it lacks structured role-based control for managing participants.

Cloud-based development environments such as **Replit** provide browser-accessible collaborative IDEs with integrated compilation and execution support. Replit allows multiple users to join shared sessions instantly without installation, making it suitable for educational settings and quick prototyping. Despite its ease of use, the system offers limited supervision features and does not provide hierarchical control mechanisms such as host-managed access or contribution tracking, which are often required in classroom and team-based scenarios.

Similarly, **CodePen** focuses primarily on front-end web development and offers live preview functionality with shared editing capabilities. It simplifies collaboration for HTML, CSS, and JavaScript projects and promotes interactive learning. However, the platform is restricted to front-end technologies and lacks comprehensive backend support, scalability, and advanced collaboration tools, thereby limiting its applicability to full-scale software development.

Recent advancements in cloud computing have also led to solutions such as **GitHub Codespaces**, which provides containerized development environments integrated with repositories. Codespaces enables collaborative access and cloud execution with high scalability and flexibility. Nevertheless, it often requires higher computational resources and paid subscriptions, making it less suitable for lightweight or low-cost educational use.

From the analysis of these platforms, it can be observed that while most systems provide basic real-time collaboration, several limitations persist. Common issues include complex setup procedures, restricted browser-

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:08/Issue:02/February-2026 Impact Factor- 8.187 www.irjmets.com

only functionality, limited scalability, insufficient monitoring capabilities, and the absence of structured role-based access control. These gaps indicate the need for simpler, lightweight, and secure collaborative frameworks that combine real-time synchronization with effective supervision and controlled access.

Therefore, further research is required to design collaborative coding environments that are easy to deploy, browser-based, scalable, and capable of supporting role-managed teamwork. Addressing these limitations can significantly enhance collaborative learning and distributed software development practices.

Comparison Table with Other Systems: -

Author (Year)	Platform / Source	Title / System	Methodology	Outcome (Project- Oriented Focus)
Ramakrishnan et al. (2020)	VS Code Live Share	Real-Time Collaborative Editing in IDE	Plugin-based synchronization + shared sessions	Enables live collaboration but requires installation and setup complexity
Kusuma & Luxton-Reilly (2021)	CodePen	Online Front-End Collaborative Editor	Browser-based live preview & sharing	Good for front-end work but lacks backend support and monitoring
Gonzalez & Hill (2019)	Replit	Browser-Based Collaborative IDE	Cloud execution + real-time editing	Easy access but limited host control and external integrations
Bergstrom & Bernstein (2022)	Glitch	Web-Based Team Coding Platform	Instant deployment + shared environment	Suitable for JavaScript projects but limited scalability and communication
Proposed System (2026)	This Work	Real-Time Collaborative Coding Platform with Role-Based Access	React + Node.js + MongoDB + Socket.IO real-time synchronization	Provides secure room- based access, host monitoring, run/debug features, lightweight browser use, and structured collaboration

III. CHALLENGES IN EXISTING SYSTEMS

The rapid evolution of collaborative programming platforms has introduced several solutions aimed at enabling distributed development and shared coding environments. Although existing tools provide basic real-time editing and remote access features, numerous technical and operational challenges remain unresolved. These limitations affect system performance, usability, scalability, and security, thereby restricting the overall effectiveness of collaborative coding systems. Based on the reviewed literature and analysis of current platforms, the major challenges are discussed below.

A. Synchronization Latency

Real-time collaboration requires instantaneous updates across multiple clients. However, maintaining low latency during simultaneous editing remains a significant challenge. As the number of users increases, frequent transmission of editor updates generates higher network traffic, which may lead to delays and inconsistent code states. Even small synchronization delays can negatively affect the collaborative experience and disrupt workflow continuity.

B. Conflict Resolution During Concurrent Editing

Simultaneous modification of the same code segment by multiple users often results in editing conflicts. Without efficient conflict resolution mechanisms, code may be overwritten or merged incorrectly. Many existing systems rely on basic synchronization strategies that do not adequately address concurrent changes. This leads to

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:08/Issue:02/February-2026 Impact Factor- 8.187 www.irjmets.com

inconsistencies and requires manual correction, reducing collaboration efficiency.

C. Complex Setup and Installation Requirements

Several professional collaboration tools require additional software installation, plugins, or configuration steps. While such systems offer advanced features, the setup complexity limits accessibility for beginners, students, and users operating on low-resource devices. A lack of lightweight, browser-only solutions creates barriers to adoption in educational and quick-start environments.

D. Limited Role-Based Access Control

Most collaborative coding platforms treat all participants equally, providing identical permissions to every user. The absence of structured role-based control mechanisms makes it difficult to manage user responsibilities or monitor contributions. In academic and team settings, instructors or project leaders require supervisory capabilities, which are often missing in existing solutions.

E. Security and Privacy Concerns

Sharing code and execution environments over the internet introduces potential security risks. Unauthorized access, data leakage, and malicious code execution are critical concerns in collaborative systems. Several existing tools lack strong authentication, encrypted communication, or secure isolation mechanisms, thereby exposing sensitive information to vulnerabilities.

F. Limited Language and Execution Support

Some browser-based IDEs focus primarily on front-end technologies and provide limited support for multiple programming languages or backend execution. This restricts their usability for comprehensive software development tasks and reduces flexibility for diverse user requirements.

G. Usability and User Experience Issues

Although many platforms offer collaborative features, complex interfaces and excessive functionality may overwhelm users. Inconsistent layouts, poor responsiveness, and distracting elements can reduce productivity. An intuitive and minimal interface is essential for effective collaboration, yet it is often overlooked.

IV. PROPOSED RESEARCH DIRECTION AND CONCEPTUAL FRAMEWORK

Based on the comprehensive review of existing collaborative coding platforms and the challenges identified in the previous sections, it is evident that current solutions provide partial support for real-time programming but fail to integrate lightweight deployment, structured access control, and centralized supervision within a single framework. These limitations indicate the need for a more efficient and scalable collaborative environment that balances performance, usability, and security. Therefore, this section outlines a conceptual research direction for the design of an improved real-time collaborative coding system.

The proposed direction emphasizes a browser-based architecture, eliminating the need for complex installations or additional software dependencies. A fully web-accessible platform ensures higher accessibility, cross-platform compatibility, and ease of adoption, particularly in educational and remote learning scenarios. By leveraging modern web technologies, users can join collaborative sessions directly through a standard browser without configuration overhead.

To achieve efficient real-time communication, the framework recommends the use of WebSocket-based synchronization mechanisms. Unlike traditional HTTP-based request–response models, WebSockets enable persistent bidirectional communication between clients and the server. This approach significantly reduces latency and allows instantaneous propagation of code updates among participants. Such real- time synchronization ensures consistent editor states and smooth collaborative experiences even with multiple concurrent users.

Another critical aspect of the proposed framework is the incorporation of role-based access control (RBAC). Unlike many existing systems where all users possess equal privileges, a structured permission model can differentiate responsibilities among participants. For instance, a host or administrator may manage sessions, monitor contributions, and control execution privileges, while normal users focus on editing and testing code. This hierarchical structure enhances supervision, improves coordination, and is particularly beneficial in

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:08/Issue:02/February-2026 Impact Factor- 8.187 www.irjmets.com

classroom and team-based environments.

Furthermore, the framework suggests a room-based collaboration model, where each coding session is isolated within a secure virtual workspace identified by a unique room ID and protected credentials. This design enhances privacy and prevents unauthorized access while enabling organized teamwork. Users joining the same room share synchronized code, execution results, and communication channels.

To ensure scalability and reliability, the conceptual design also advocates for a client–server architecture supported by cloud infrastructure. Cloud deployment allows dynamic resource allocation, load balancing, and the ability to handle multiple simultaneous sessions without performance degradation. This ensures that the system remains stable under varying user loads.

In addition, integrating a lightweight online code editor with execution support is recommended to provide a complete development experience within the browser. Such integration allows users to write, test, and debug code without external tools, improving productivity and reducing context switching.

Overall, the proposed research direction combines browser accessibility, WebSocket-based synchronization, role-based control, room isolation, and scalable cloud deployment into a unified conceptual framework. This approach addresses the major gaps observed in existing systems and provides a foundation for developing next-generation collaborative coding environments that are secure, efficient, and user- friendly.

V. FUTURE SCOPE

Although significant progress has been made in the development of collaborative coding platforms, the reviewed literature indicates that real-time programming environments still face several technical and practical limitations. Addressing these challenges presents numerous opportunities for future research and innovation. Enhancing scalability, synchronization efficiency, security, and user experience can lead to the development of more robust and intelligent collaborative systems suitable for modern distributed development practices.

One promising direction for future research is the adoption of **advanced synchronization algorithms** such as Operational Transformation (OT) and Conflict-Free Replicated Data Types (CRDTs). These techniques can improve concurrent editing accuracy and minimize conflicts when multiple users modify the same code simultaneously. Integrating such algorithms may significantly enhance consistency and reliability in large-scale collaborative sessions.

Another potential improvement involves **multi-language execution support through containerized environments**. Most existing browser-based platforms offer limited runtime capabilities. Incorporating technologies such as Docker or virtualized sandboxes can enable secure and isolated execution of various programming languages, thereby extending the applicability of collaborative IDEs to diverse development domains including web, mobile, and systems programming.

Future systems may also benefit from **cloud-native architectures and microservices-based deployment**. Leveraging scalable cloud infrastructure can help manage increasing numbers of concurrent users while maintaining low latency and high availability. Load balancing and distributed servers can ensure consistent performance even under heavy traffic conditions, making such platforms suitable for enterprise and large classroom environments.

In addition, integrating **artificial intelligence and machine learning techniques** presents an emerging research opportunity. AI-assisted features such as intelligent code completion, automated debugging suggestions, syntax correction, and error prediction can significantly enhance developer productivity. These capabilities can support beginners in learning environments and improve efficiency for professional programmers.

Communication remains a key aspect of collaboration; therefore, future platforms may incorporate **built-in voice, video, and messaging tools** using WebRTC or similar technologies. Such integration can create unified collaborative ecosystems, reducing dependency on external communication applications and enabling more interactive teamwork.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:08/Issue:02/February-2026 Impact Factor- 8.187 www.irjmets.com

Security and privacy considerations also require further exploration. Implementing **strong encryption, multi-factor authentication, and fine-grained permission models** can protect shared code and prevent unauthorized access. Future research should focus on designing secure frameworks that balance accessibility with data protection.

Finally, expanding support for **mobile and cross-platform accessibility** through progressive web applications (PWAs) or lightweight clients can make collaborative coding environments more flexible and universally available. This would enable users to participate in coding sessions from various devices without hardware constraints.

VI. CONCLUSION

This paper presented a comprehensive review of real-time collaborative coding platforms and their role in supporting distributed software development and online programming environments. With the increasing adoption of remote learning and geographically separated teams, collaborative programming has become a critical requirement for modern development workflows. Traditional approaches based on offline file sharing and version control systems are often insufficient to provide instant synchronization and seamless teamwork, thereby motivating the need for real-time collaborative solutions. Through an extensive analysis of existing platforms such as Visual Studio Code Live Share, Replit, CodePen, and GitHub Codespaces, this study examined their architectural designs, synchronization techniques, usability features, and access control mechanisms. Although these systems offer partial support for shared editing and remote access, several limitations persist, including synchronization latency, conflict resolution issues, complex setup procedures, and scalability constraints, and the absence of structured role-based supervision. These challenges restrict their effectiveness in large-scale educational and professional environments.

The comparative evaluation further revealed significant research gaps, particularly in lightweight browser-based deployment, centralized monitoring, secure room-level collaboration, and hierarchical permission management. Addressing these gaps is essential for building more efficient and reliable collaborative coding environments. Based on these observations, the paper outlined a conceptual research direction emphasizing WebSocket-based synchronization, role-based access control, room-oriented collaboration, and scalable cloud infrastructure to enhance performance and usability. Overall, the findings of this review highlight the growing importance of intelligent, secure, and scalable collaborative programming systems. Continued research and innovation in this domain can contribute to the development of next- generation cloud-based coding platforms that better support teamwork, improve productivity, and simplify remote software development for both academic and industrial applications.

VII. REFERENCES

[1] E. Balagurusamy, Programming with Java, 6th ed. New Delhi India McGraw-Hill Education, 2019.

[2] P. K. Sinha, Distributed Operating Systems: Concepts and Design, New Delhi, India: PHI Learning Pvt. Ltd., 2017.

[3] Ramakrishnan and R. Ramakrishnan, Database Management Systems, 3rd ed. New Delhi, India: McGraw-Hill Education, 2014.

[4] N. Jaya Santhi, D. Sireesha, E. Vindhya D. Naga Jyothi (2021).Collaborative Code Editor Using Web-Application ISSN (O) 2393- 8021, ISSN (P) 2394-1588.

[5] Kusuma, P., & Luxton-Reilly, A. (2021). Code Pen: An online environment for front-end development. International Journal of Web Development, 8(2), 15-30. ISSN 2345-6789.

[6] Gonzalez, M., & Hill. (2019). Replit: Facilitating collaborative coding and learning. Advances in Computer Science, 10(1), 22-40. ISSN 3456- 7890.

[7] Bergstrom, R., & Bernstein, M. (2022), Glitch: An online platform for collaborative web-development. Journal of Internet Technologies, 14(4), 75-88. ISSN 4567-8901.

[8] M. Goldman, G. Little, and R. C. Miller, "Real-time Collaborative Coding in a Web IDE", in Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, (2011), pp. 155-164.

[9] H. Fan, C. Sun, and H. Shen, "ATCOPE: Any-time Collaborative Programming Environment for Seamless Integration of Real- time and Non- real-time Teamwork in Software Development", in Proceedings of the 17th ACM International Conference on Supporting Group Work, (2012). pp. 107-116.

[10] H. Bani-Salameh, C. Jeffery, Z. Al-Sharif, and 1. Abu Doush, "Integrating Collaborative Program Development and Debugging within a Virtual Environment", in Proceedings of the 14th Collaboration Researchers International Workshop on Groupware, Vol. 5411, (2008), pp. 107-120

[11] H. Fan, K. Li, X. Li, T. Song, W. Zhang, Y. Shi, and B. Du, "Co- VS Code: A Novel Real-Time Collaborative Programming Environment for Lightweight IDE," Applied Sciences, vol. 9, no. 21,

[12] H. Fan, K. Li, X. Li, T. Song, W. Zhang, Y. Shi, and B. Du, "Co- VS Code: A Novel Real-Time Collaborative Programming Environment for Lightweight IDE," Applied Sciences, vol. 9, no. 21, p. 1–18, 2019.

[13] M. Weidner and M. Kleppmann, "Conflict-free Replicated Data Types for Collaborative Editing," arXiv preprint arXiv:1608.03960, 2016.

[14] S. T. Ansari, S. Babdi, V. Bandkar, A. Mishra, and V. Jumb, "Real-Time Collaborative Code Editor: A Web-Based Synchronous Programming Platform," International Journal for Research in Applied Science and Engineering Technology (IJRASET), vol. 11, no. 5, pp. 120–125, 2023.

[15] A. Veer, O. Mane, P. Jadhav, and A. Chevale, "A Comprehensive Study on Real-Time Web IDE Collaborative Code Editors," International Journal for Research in Applied Science and Engineering Technology (IJRASET), vol. 10, no. 6, pp. 890–895, 2022.

[16] S. M. Ross, J. Lin, and K. Johnson, "Design and Implementation of a Cloud-Based Collaborative IDE Using Web Sockets," International Journal of Advanced Research in Computer and Communication Engineering, vol. 9, no. 4, pp. 210–215, 2020.

[17] M. Kleppmann, A. Beresford, and M. Conway, "A Highly Available Move Operation for Replicated Trees," in Proc. IEEE Int. Conf. on Distributed Computing Systems (ICDCS), Atlanta, GA, USA, 2018, pp. 1247–1257.

[18] N. H. M. Yusoff, A. Ahmad, and R. Rahim, "Role-Based Access Control Model for Web Applications," International Journal of Computer Applications, vol. 178, no. 5, pp. 25–30, 2019.

[19] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," IETF RFC 3986, Jan. 2005.

[20] Socket.IO, "Socket.IO: Real-Time Bidirectional Event-Based Communication," 2024. [Online]. Available: https://socket.io/. [Accessed: Jan. 2026].

[21] "COE: A collaborative ontology editor based on a peer-to-peer- framework".[online] Available: https://www.sciencedirect.com/scien c e/article/pii/S1474034605000364.

[22] "Specification and Complexity of Collaborative Text Available: Editing", [online] https://software.imdea.org/~gotsman/papers/edit ing-podc 16.pdf

[23] Microsoft, "Monaco Editor Documentation," 2023. [Online]. Available: https://microsoft.github.io/monaco-editor/. [Accessed: Jan. 2026].