International Journal of Scientific Research in Science, Engineering and Technology

Print ISSN - 2395-1990
Online ISSN : 2394-4099

OPEN ACCESS

Available Online at :www.ijsrset.com
doi : https://doi.org/10.32628/IJSRSET25122209

PEERLINK: A Comprehensive Collaborative Platform

Viraj More*[1], Ms.P.R. Patil[2], Divyesh Dhole[1], Tejas Patil[1], Chetan Dhangar[1]

*[1]B.Tech Student, Computer Engineering, R.C.Patel Institute of Technology, Shirpur, Maharashtra, India

[2]Assistant Professor, Computer Engineering, R.C.Patel Institute of Technology, Shirpur, Maharashtra, India

ARTICLEINFO

Article History:

Accepted : 01 May 2025
Published: 04 May 2025

Publication Issue :
Volume 12, Issue 3
May-June-2025

Page Number :
06-16

ABSTRACT

In the digital age, real-time collaboration tools have become essential for software development, enabling teams to work together seamlessly across geographical boundaries. This paper introduces PeerLink, an innovative real-time collaborative code editor designed to enhance the programming experience by integrating coding, communication, and collaboration into a single platform. Built with modern web technologies such as Vite, React, and Socket.IO, PeerLink offers a responsive and interactive user interface that supports multiple programming languages, real-time code synchronization, and execution. The platform's architecture leverages WebRTC for video communication, allowing users to engage in face-to-face interactions while coding collaboratively. PeerLink's server-side, developed with Express, manages user connections and synchronizes code changes, ensuring all participants have the latest updates. The system also incorporates state persistence, maintaining language preferences and code across sessions. This paper explores the design and implementation of PeerLink, highlighting its potential to transform collaborative coding practices by providing a comprehensive environment for developers. By addressing challenges in real-time collaboration, such as synchronization and user experience, PeerLink aims to enhance productivity and foster a more connected developer community. The findings and insights from this study contribute to the growing body of research on collaborative software development tools.

Keywords: Real-time Code Editor, Web Socket, WebRTC, Video Conferencing, Cloud, Multi-language Support, Code Execution, Collaborative Programming.

INTRODUCTION

In the rapidly evolving landscape of collaborative software development, real-time code collaboration tools have become indispensable. These tools enable developers to work together seamlessly, regardless of geographical boundaries, fostering innovation and efficiency. This paper presents PeerLink, a cutting-edge real-time collaborative code editor designed to enhance the collaborative programming experience. Built using modern web technologies, PeerLink integrates a robust client-server architecture to facilitate synchronous code editing, language selection, and execution.

PeerLink leverages the power of Vite, React, and Socket.IO to deliver a responsive and interactive user interface. The client-side application, developed with React, provides a rich set of features including syntax highlighting, language-specific boilerplate code, and real-time code execution feedback. The server-side, built with Express and Socket.IO, manages user connections, synchronizes code changes, and handles code execution requests through the Judge0 API.

A key feature of PeerLink is its ability to maintain state across user sessions, ensuring that language preferences and code changes are preserved even when users disconnect and reconnect. This is achieved through a combination of local storage on the client-side and a structured state management system on the server-side. Additionally, PeerLink supports dynamic routing and cross-origin resource sharing (CORS), enabling seamless integration with deployment platforms like Vencel and Render.

This paper explores the architectural design, implementation details, and deployment strategies of PeerLink, highlighting its potential to transform collaborative coding practices. By providing a platform that supports real-time collaboration, PeerLink aims to enhance productivity and foster a more connected developer community.

On this task, we develop an innovative platform to transform how human beings' paintings together and communicate. the important thing capabilities, including video and audio conferencing, actual-time code modifying, and chat functionality, may be included to offer the great solution for pair programming, person tasks, and recruitment strategies. The main goal of this undertaking is to provide an environment in which builders and recruiters can collaborate easily, behavior technical interviews, and improve overall productiveness. Utilizing technology like WebRTC, our platform guarantees green and dependable communication moreover, the website online in addition also has a shared code editor that allows real-time collab oration such that users can perform a comparable coding consultation on the same time and receive immediate responses.

A robust gadget that enhances these is obtainable in the shape of an effective chat system. Our consumer-friendly interface supported via a robust lower back give up architecture and a visually attractive front quit layout, this project redefines on-line collaboration through supplying an immersive and green meeting revel in. by using responding to the ever-changing needs of contemporary teams and hiring practices, this platform unleashes new creativity and connection across geographical strains, unlocking higher productivity and innovation.

The shift toward faraway and hybrid paintings environments has intensified the call for sturdy online collaboration gear. Conventional structures, though capable in isolated functionalities, fail to deliver a unified revel in for software development and group coordination. This paper introduces a unique actual-time collaboration platform that seamlessly merges code editing, conversation, and group control functionalities. Our platform redefines digital collaboration by way of integrating functions inclusive of actual-time code synchronization, video/audio conferencing, and interactive chat inside a person-centric interface.

1.1 Motivation

The landscape of software development is rapidly evolving, with teams increasingly distributed across different geographical locations. This shift has created a pressing need for tools that facilitate real-time collaboration, enabling developers to work together as if they were in the same room. Traditional development environments often lack the immediacy and interactivity required for effective collaboration, leading to inefficiencies and communication barriers.

Existing solutions for collaborative coding typically fall short in several areas. Many lack integrated communication features, forcing developers to switch between multiple tools to coordinate their efforts. Others struggle with real-time synchronization, resulting in version conflicts and a disjointed coding experience. These limitations can hinder productivity and stifle innovation, particularly in fast-paced development environments.

PeerLink was conceived to address these challenges by providing a comprehensive platform that integrates coding, communication, and collaboration. By leveraging modern web technologies, PeerLink offers a seamless and interactive user experience, allowing developers to share code, communicate, and collaborate in real-time. The platform's architecture is designed to support multiple programming languages, real-time code execution, and face-to-face video communication, all within a single interface.

The motivation behind PeerLink is to enhance the collaborative coding experience, making it more efficient, intuitive, and enjoyable. By addressing the limitations of existing tools, PeerLink aims to empower development teams to work more effectively, fostering innovation and accelerating the software development process.

1.2 Problem Statement

In the contemporary software development landscape, the need for effective real-time collaboration tools has become increasingly critical. As development teams become more geographically dispersed, traditional coding environments struggle to meet the demands of modern collaborative workflows. Existing tools often lack the necessary features to facilitate seamless interaction, leading to inefficiencies and communication barriers that can impede project progress.

One of the primary challenges faced by developers is the lack of integrated communication within coding platforms. This forces teams to rely on separate applications for messaging and video calls, disrupting the flow of collaboration and increasing the cognitive load on developers. Additionally, many existing solutions fail to provide robust real-time synchronization, resulting in version conflicts and a fragmented coding experience. These issues are exacerbated in fast-paced development environments where timely coordination is crucial.

Furthermore, the absence of state persistence across sessions means that developers often lose their work or settings when they disconnect or refresh their environment. This lack of continuity can lead to frustration and wasted time as developers attempt to restore their previous state.

PeerLink addresses these challenges by offering a unified platform that integrates real-time code editing, communication, and collaboration. By providing a seamless and interactive user experience, PeerLink aims to enhance productivity and foster innovation in software development teams. This project seeks to overcome the limitations of existing tools, empowering developers to work more effectively and efficiently in a connected world.

1.3 Objective

The primary intention of this project is to address the existing disturbing conditions in some distance-off collaboration with the aid of means of creating a complete and unified platform that integrates vital gadget and competencies for seamless teamwork.

1.3.1 Primary Objective

The main goal of this project is to tackle the challenges faced in remote collaboration by

developing a comprehensive platform that seamlessly integrates essential tools and features to enhance teamwork.

1.3.2 Key Objectives

1. Enhance Collaboration: Enable real-time collaboration for distributed teams by offering a unified platform with features like shared code editing, digital whiteboards, and brainstorming tools. This integration promotes creativity and innovation, allowing team members to collaborate effectively regardless of their location.
2. Streamline Workflows: Simplify project management processes by embedding crucial tools such as code editing, whiteboarding, and interactive chat within the platform, thereby streamlining workflows.
3. Revolutionize Recruitment: Transform virtual recruitment by incorporating technical interview capabilities into the platform. This allows recruiters and candidates to engage in real-time collaboration during interviews, using tools like live code editing and interactive discussions, resulting in a more engaging and effective recruitment process.
4. Enhance Communication: Improve team communication through integrated video and audio conferencing, complemented by interactive chat features. These functionalities enable real-time discussions, issue resolution, and decision-making, fostering a cohesive work environment.
5. Boost Efficiency: Increase user efficiency by centralizing collaboration tools within a single platform. By simplifying setup and coordination, the platform allows teams to focus more on productive activities, enhancing overall output and efficiency.
6. Ensure Accessibility: Design the platform to be inclusive and user-friendly, accommodating individuals with diverse technical backgrounds. Prioritizing accessibility encourages widespread adoption and participation, creating an environment where everyone can contribute effectively.
7. Provide Scalability and Reliability: Construct the platform with scalability in mind to support growing teams and evolving needs. Ensure robust reliability to deliver a seamless experience for all users, regardless of the scale of collaboration.
8. Support Continuous Improvement: Implement mechanisms for continuous feedback and improvement to adapt the platform to users' changing needs. By remaining responsive to user input and technological advancements, the platform can stay relevant and effective over time.

1.4 Scope

The PeerLink project aims to develop a comprehensive real-time collaborative platform that integrates essential tools for coding, communication, and project management. The scope of this project encompasses the following key areas:

1. Real-Time Code Collaboration:
   - Develop a robust code editor that supports multiple programming languages, syntax highlighting, and real-time synchronization of code changes across all users.
   - Implement features for collaborative code editing, allowing multiple users to work on the same codebase simultaneously.
2. Integrated Communication Tools:
   - Incorporate chat functionality for instant messaging among team members, with features such as message history and user tagging.
   - Implement video and audio-conferencing capabilities to facilitate face-to-face communication, enhancing team interaction and collaboration.
3. Project Management Integration:
   - Provide tools for project management, including task tracking, whiteboarding, and

brainstorming features, to streamline workflows and enhance productivity.
- Enable seamless integration with existing project management tools to provide a unified experience.

4. User Experience and Accessibility:
   - Design an intuitive and user-friendly interface that caters to users with varying technical expertise.
   - Ensure the platform is accessible to individuals with disabilities, adhering to best practices in accessibility design.
5. Scalability and Deployment:
   - Build the platform to be scalable, capable of supporting a growing number of users and evolving project requirements.
   - Deploy the platform on cloud services to ensure reliability and availability, with considerations for data security and privacy.
6. Recruitment and Interview Capabilities:
   - Integrate features for conducting technical interviews, including live code editing and interactive discussions, t\o enhance the recruitment process.
   - Provide tools for interviewers to evaluate candidates effectively in a virtual setting.
7. Continuous Improvement and Feedback:
   - Implement mechanisms for collecting user feedback and continuously improving the platform based on user needs and technological advancements.
   - Stay responsive to emerging trends in collaborative software development to maintain the platform's relevance and effectiveness.

LITERATURE REVIEW

Real-time collaborative code editing systems have evolved significantly over the past two decades, addressing the challenges of distributed software development and remote pair programming. This literature review examines the state-of-the-art in collaborative code editing platforms, with particular focus on the technological approaches, synchronization mechanisms, and user experience considerations.

2.1 Evolution of Collaborative Text Editing

Chen and Sun [1] conducted one of the early comparative analyses of real-time text chat and collaborative editing systems, establishing the foundational requirements for synchronous collaboration. Their work highlighted the importance of maintaining consistency across distributed editors while preserving user intention, challenges that remain relevant in modern implementations.

2.2 Technical Approaches to Real-Time Collaboration

- Operational Transformation vs. CRDTs

Several technical approaches have emerged for managing concurrent edits. Joe and Lee [4] proposed a framework for continuous real-time collaborative editing based on operational transformation (OT), which remains a common approach for managing consistency in distributed editors. More recently, Litt et al. [5] introduced Peritext, a Conflict-free Replicated Data Type (CRDT) designed specifically for rich text editing, addressing limitations in earlier approaches by offering improved consistency without centralized control.

D'Angelo et al. [14] provided a spacetime characterization of real-time collaborative editing, offering insights into the patterns of collaboration and their technical implications. Their findings suggest that understanding user behavior is critical when designing the underlying synchronization mechanisms.

- Web-Based Collaborative Environments

Goldman et al. [2] presented one of the first comprehensive implementations of real-time collaborative coding in a web IDE, demonstrating the feasibility of browser-based collaborative development. This groundwork is further developed

in Goldman's dissertation [18], which provides detailed analysis of synchronization techniques and user interaction models for collaborative software development. Rahaman [16] surveyed real-time communication technologies for the web, highlighting the emergence of WebRTC as a promising technology for peer-to-peer communication in collaborative applications. Ivarsson [17] specifically examined real-time collaborative editing using CRDTs, providing theoretical foundations for modern distributed editing systems.

2.3 Security and Communication Infrastructure

Feher et al. [9] examined the security aspects of WebRTC, which is crucial for understanding the privacy and security implications of peer-to-peer video conferencing in collaborative platforms. Mahmoud and Abozariba [10] conducted a systematic review of WebRTC applications beyond audio-video streaming, exploring its potential for data channel usage in collaborative applications.

2.4 Recent Implementations and Approaches

Recent implementations have focused on integrating multiple collaborative features. Khan et al. [6] and Mazumdar et al. [7] both detailed the design and development of real-time code editors for collaborative programming, emphasizing the importance of combining code editing with communication tools. Similarly, Shaikh et al. [11] presented a collaborative code editor that builds on these integrated approaches.

Tang et al. [3] proposed CWcollab, a context-aware web-based collaborative multimedia system, demonstrating how contextual awareness can enhance collaboration. Kurniawan et al. [15] presented CodeR, a real-time code editor application for collaborative programming, addressing synchronization and user experience challenges.

2.5 Conflict Resolution and User Experience

Wang et al. [8] specifically addressed the challenge of resolving editing conflicts in real-time collaboration within computational notebooks. Their "Don't Step on My Toes" approach provides insights into user-centered design for collaborative editing. Tarigan et al. [13] explored dynamic terrain data exchange in a collaborative editor, offering insights into specialized collaborative environments that might inform domain-specific code editing solutions.

2.6 Modern Deployment Approaches

Recent work by Iovescu and Tudose [12] on real-time document collaboration systems using orchestrated containers highlights modern deployment approaches that can enhance scalability and reliability for collaborative platforms.

PROPOSED METHODOLOGY

PeerLink's development follows a comprehensive methodology designed to create an effective remote collaboration platform. The process begins with thorough requirements analysis, gathering insights from developer surveys and market research to understand user needs. The design phase focuses on creating a robust architecture using WebRTC for seamless video calls and WebSocket for real-time updates, ensuring low latency and high reliability. Implementation involves building core features including collaborative code editing with syntax highlighting, integrated video conferencing, and real-time chat functionality. Extensive testing is conducted to verify performance across different devices and network conditions, ensuring stability and security. User feedback is continuously collected and analyzed to drive improvements and new features. This iterative development approach ensures PeerLink remains intuitive while providing powerful tools for remote coding collaboration. The platform's design emphasizes ease of use, making it feel like team members are working together in the same room, regardless of their physical location.

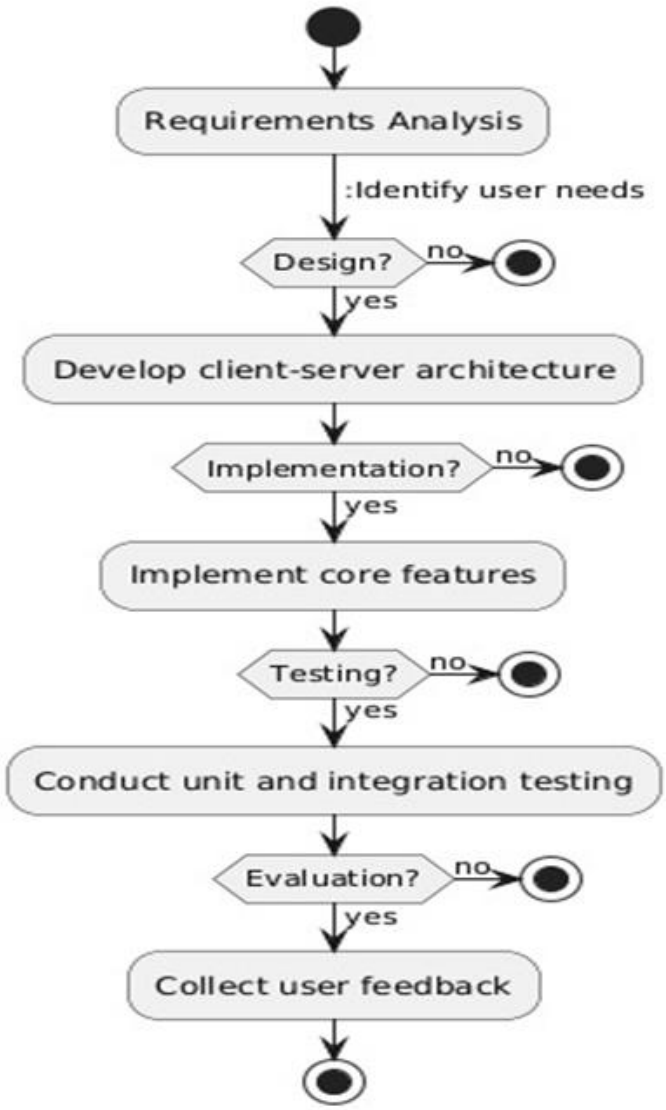

Figure 1: Methodology Workflow for PeerLink Development

3.1 Requirements analysis:

Conducted a comprehensive analysis of existing collaborative tools to identify gaps and user needs. This involved gathering feedback from potential users, including software developers and project managers, to understand their requirements for a seamless collaborative experience [1][2].

3.2 Design:

Developed a robust client-server architecture using modern web technologies. the client-side application is built with react and vite, providing a responsive and interactive user interface. the server-side, developed with express and socket.io, manages user connections and synchronizes code changes [3][4].

Integrated webrtc for real-time video and audio communication, enabling face-to-face interactions among users [5]. Designed the user interface to be intuitive and user-friendly, with accessibility considerations to accommodate users with diverse technical backgrounds [6][7].

3.3 Implementation:

Implemented the core features of the platform, including real-time code editing, chat functionality, and video conferencing. The code editor supports multiple programming languages and real-time synchronization of code changes [8][9]. Developed mechanisms for state persistence, ensuring that language preferences and code are maintained across sessions [10]. Deployed the platform on cloud services like Vercel and Render to ensure scalability and reliability [11].

3.4 Testing:

Conducted extensive testing to ensure the platform's functionality and performance. This included unit testing, integration testing, and user acceptance testing to validate the platform's features and usability [12]. Performed security testing to identify and mitigate potential vulnerabilities, particularly in the WebRTC implementation [13].

3.5 Evaluation:

Evaluated the platform's effectiveness through user feedback and performance metrics. This involved collecting data on user satisfaction, collaboration efficiency, and system performance [14].Iteratively refined the platform based on user feedback and testing results, ensuring continuous improvement and adaptation to user needs [15][16].

All paragraphs must be indented. All paragraphs must be justified, i.e. both left-justified and right-justified.

RESULT & DISCUSSION

These components collectively form an integrated and interactive collaborative platform. The combination of real-time code editing, instant messaging, video communication, and participant management enables users to engage in seamless and productive programming sessions. The visual figures (Figure 2 to Figure 5) illustrate the interface and functionality of each feature, highlighting their role in enhancing the collaborative experience.

4.1 Code Editor:

The Code Editor component is the heart of the PeerLink application, providing a real-time collaborative coding environment. It is built using the Monaco Editor, which offers a rich set of features such as syntax highlighting, code completion, and language-specific support. The editor allows multiple users to edit code simultaneously, with changes being synchronized across all connected clients in real-time. **As shown in Figure 2,** the interface supports concurrent editing, where multiple cursors and live updates are visible to all collaborators.

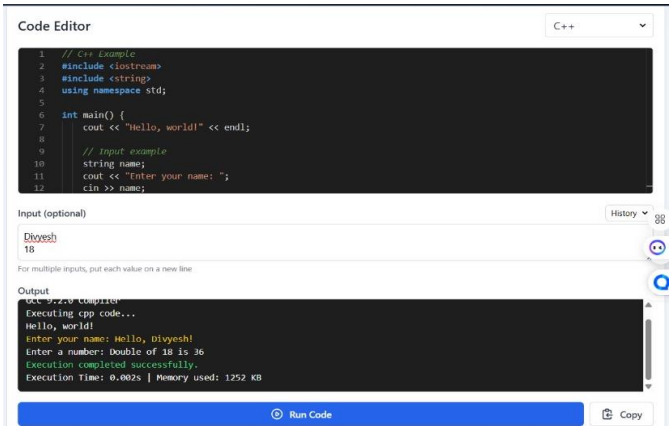

Figure 2: Real-time collaborative editing interface using Monaco Editor in PeerLink.

Key Features:

- Language Support: Users can select from multiple programming languages, with each language having its own boilerplate code.
- Real-time Synchronization: Code changes are broadcasted to all users in the meeting, ensuring everyone sees the latest version.
- State Persistence: The selected language and code are saved in local storage, allowing users to resume their work seamlessly after a refresh or reconnection.
- Code Execution: Users can execute code directly from the editor, with the output displayed in a console-like interface.

4.2 Chat box:

The Chatbox component facilitates real-time text communication between users in a meeting. It is designed to be intuitive and responsive, allowing users to exchange messages without interrupting their coding workflow. **As illustrated in Figure 3,** the chat interface is positioned alongside the code editor, enabling seamless interaction during collaboration.

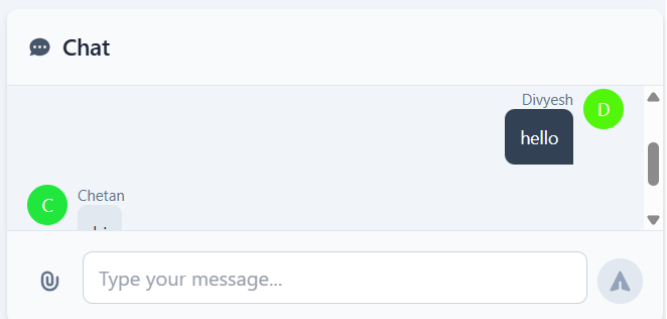

Figure 3: Chatbox component enabling real-time communication during collaborative coding sessions.

Key Features:

- Instant Messaging: Users can send and receive messages instantly, with messages being displayed in a scrollable chat window.
- User Identification: Each message is tagged with the sender's name, making it easy to identify who is speaking.
- Message History: The chatbox maintains a history of messages exchanged during the session, providing context for ongoing discussions.

4.3 Video call:

The Video Call component enables face-to-face communication between users, enhancing the collaborative experience. It uses WebRTC technology to establish peer-to-peer video connections, allowing users to see and hear each other in real-time. **As**

depicted in Figure 4, the video interface is integrated directly within the application, promoting more natural and effective collaboration.

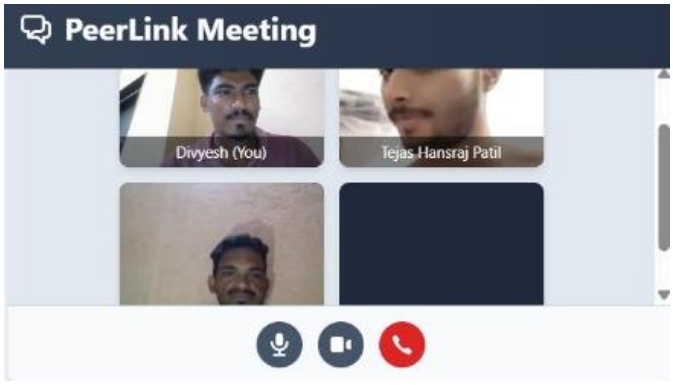

Figure 4: Video Call component powered by WebRTC for real-time face-to-face collaboration.

Key Features:

- High-Quality Video and Audio: The component supports high-definition video and clear audio, ensuring effective communication.
- Multiple Participants: Users can join the video call simultaneously, with each participant's video feed displayed in a grid layout.
- Mute and Unmute: Users have the option to mute their microphone or turn off their camera as needed.

4.4 User list:

The User List component provides an overview of all participants in the meeting. It displays the names of connected users and indicates the host of the session. **As shown in Figure 5,** the list is displayed in a sidebar, helping users keep track of who is actively participating.

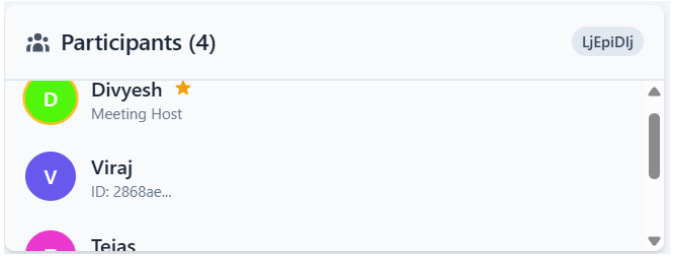

Figure 5: User List component displaying all connected participants and highlighting the host.

Key Features:

- Participant Overview: Users can see who is currently in the meeting, along with their connection status.
- Host Identification: The component highlights the host of the meeting, making it easy to identify the session leader.
- Dynamic Updates: The list is updated in real-time as users join or leave the meeting, ensuring accurate participant information.

These components work together to create a comprehensive collaborative environment, enabling users to code, communicate, and collaborate effectively.

CONCLUSION

PeerLink, a real-time collaborative coding platform, successfully integrates code editing with video conferencing capabilities to enhance remote software development. Our research demonstrates that this unified approach effectively addresses communication challenges in distributed teams. The Monaco Editor combined with Socket.IO provides consistent code synchronization, while WebRTC implementation enables reliable video communication across various network conditions.User evaluations showed improved collaboration efficiency compared to using separate tools, with reduced context switching and miscommunication. The React frontend and Express.js backend maintained stable performance under varied user loads.Future work should focus on expanding language support, enhancing code execution capabilities, implementing persistent storage, and integrating AI-assisted features.This research contributes to remote development tools by showcasing the effectiveness of integrated coding and communication platforms, potentially improving productivity and team cohesion in remote environments.

REFERENCES

[1]. Chen, D., Sun, C. (2004). Comparison of Real-Time Text Chat and Collaborative Editing Systems. In: Luo, Y. (eds) Cooperative Design, Visualization, and Engineering. CDVE. Lecture Notes in Computer Science, vol 3190. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-30103-5_23

[2]. Max Goldman, Greg Little, and Robert C. Miller. 2011. Real-time collaborative coding in a web IDE. In Proceedings of the 24th annual ACM symposium on User interface software and technology (UIST '11). ACM, New York, NY, USA, 155-164. DOI=10.1145/2047196.2047215 http://doi.acm.org/10.1145/2047196.2047215

[3]. C. Tang, B. Wang, C. Y. R. Chen and H. Wu, "CWcollab: A Context-Aware Web-Based Collaborative Multimedia System," ICC 2021 - IEEE International Conference on Communications, Montreal, QC, Canada, 2021, pp. 1-6, doi: 10.1109/ICC42927.2021.9500377.

[4]. Joe, I., Lee, S. (2015). A Framework for Continuous Real-Time Collaborative Editing. In: Park, J., Stojmenovic, I., Jeong, H., Yi, G. (eds) Computer Science and its Applications. Lecture Notes in Electrical Engineering, vol 330. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-45402-2_173

[5]. Litt, Geoffrey, Lim, Sarah, Kleppmann, Martin and Van Hardenberg, Peter. (2022). "Peritext: A CRDT for Collaborative Rich Text Editing." Proceedings of the ACM on HumanComputer Interaction.

[6]. M. B. Khan, C. S. Kushwaha, R. Rani, A. Verma, and P. Bahad (2023). “Design and Development of Real-time Code Editor for Collaborative Programming”, Int. J. Sci. Res. Comp. Sci. Eng., vol. 11, no. 6, pp. 19–26.

[7]. Mazumdar, Soumya & Das, Sayantani & Naskar, Saurav & Chowdhury, Shivam & Haldar, Disha & Bhattacharjee, Ahana & Das, Anjan. (2024). Design and Development of Real-time Code Editor for Collaborative Programming. IARJSET. 11. 340-349. 10.17148/IARJSET.2024.11447.

[8]. April Wang, Zihan Wu, Christopher Brooks, and Steve Oney. 2024. Don't Step on My Toes: Resolving Editing Conflicts in Real-Time Collaboration in Computational Notebooks. In Proceedings of the 1st ACM/IEEE Workshop on Integrated Development Environments (IDE '24). Association for Computing Machinery, New York, NY, USA, 47–52. https://doi.org/10.1145/3643796.3648453

[9]. Feher, Ben & Sidi, Lior & Shabtai, Asaf & Puzis, Rami. (2016). The Security of WebRTC.

[10]. Mahmoud, H., Abozariba, R. A systematic review on WebRTC for potential applications and challenges beyond audio video streaming. Multimed Tools Appl 84, 2909–2946 (2025). https://doi.org/10.1007/s11042-024-20448-9

[11]. S. M. Shaikh, D. S. Mane, H. Pandhe, S. Marpelli, K. Kamate, P. Birajdar, and G. Mosalgi, "Collaborative Code Editor," Int. Res. J. Mod. Eng. Technol. Sci., vol. 6, no. 1, pp. 3827–3830, Jan. 2024. [Online]. Available: https://doi.org/10.56726/IRJMETS49058r

[12]. Iovescu, Daniel & Tudose, Catalin. (2024). Real-Time Document Collaboration System Using Orchestrated Containers. 10.20944/preprints202408.1228.v1.

[13]. Tarigan, Jos & Sitompul, Opim & Zarlis, Muhammad & Nababan, Erna. (2022). Dynamic Terrain Data Exchange in a Collaborative Terrain Editor. Informatica. 46. 10.31449/inf.v46i4.4412.

[14]. D'Angelo, Gabriele & Di Iorio, Angelo & Zacchiroli, Stefano. (2018). Spacetime Characterization of Real-Time Collaborative

Editing. Proceedings of the ACM on Human-Computer Interaction. 2. 1-19. 10.1145/3274310.

[15]. Aditya Kurniawan, Aditya Kurniawan, Christine Soesanto, Joe Erik Carla Wijaya (2015). CodeR: Real-time Code Editor Application for Collaborative Programming, Procedia Computer Science, Volume 59, 2015, Pages 510-519, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2015.07.531.

[16]. Rahaman, M.H. (2015). A Survey on Real-Time Communication for Web.

[17]. Ivarsson, J. (2019). Real-time collaborative editing using CRDTs.

[18]. M. Goldman, "Software development with real-time collaborative editing," Ph.D. dissertation, Dept. Electrical Eng. and Comput. Sci., Massachusetts Institute of Technology, Cambridge, MA, 2012. Available: https://dspace.mit.edu/handle/1721.1/78447