International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 04 | April - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

PAIRON – AN ADAPTIVE PEER-MATCHING ENGINE FOR COLLABORATIVE PROJECT DEVELOPMENT

Mrs. Hemalatha S V[#1], Jaikisan Jegadeesan[#2], Harissh K S[#3], Anbuselvi M[#4]

[#1] As Associate Professor, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India.

[#2, #3, #4,] UG Students, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India.

Abstract — PairOn is an intelligent, real-time collaborative developer platform designed to enhance coding, learning, and interaction entirely within a web browser. Built using React 18 and TypeScript, the platform integrates a full-featured in-browser IDE powered by Monaco Editor and WebContainers, enabling code execution without any local setup. It utilizes Socket.IO for real-time bidirectional communication, allowing seamless file synchronization, shared terminal access, and live collaboration between users. The system also incorporates a smart matchmaking engine that connects developers based on skill level, programming language, and session preferences. Additionally, an AI assistant powered by the Groq API provides real-time code review, debugging support, and intelligent suggestions. With secure authentication using JWT and scalable data management via MongoDB Atlas, PairOn delivers a unified, efficient, and interactive development environment. Future enhancements include video/voice communication, multi-language support, cloud-based file storage, and deeper AI-driven pair programming features.

Keywords— Collaborative Coding, Real-Time Systems, Pair Programming, WebContainers, Socket.IO, AI Assistant, Monaco Editor, Developer Platform, Browser IDE, Matchmaking System, Web Application.

I. INTRODUCTION

The advancement of web technologies has transformed the way developers collaborate, making it possible to build and share projects in real time from anywhere. Modern development environments now focus on accessibility, speed, and seamless collaboration without requiring complex local setups. At the same time, the integration of Artificial Intelligence (AI) has improved coding efficiency by providing instant suggestions, debugging support, and intelligent assistance.

However, most existing platforms rely on multiple tools for coding, communication, and project management, which leads to inefficiencies and constant context switching. Developers often need separate applications for writing code, discussing ideas, and managing tasks, resulting in a fragmented workflow. Additionally, many platforms lack a proper system to connect developers with suitable collaborators in real time.

To address these challenges, this project introduces **PairOn**, a browser-based collaborative developer platform that combines coding, communication, and AI assistance into a single environment. The system enables real-time collaboration using Socket.IO, while WebContainers allow code execution directly in the browser without any local installation. The platform also includes a smart matchmaking system that pairs users based on their skills and preferences.

The system is built using React 18 and TypeScript for the frontend and Node.js with Express.js for backend services. Secure authentication is handled using JWT, and MongoDB Atlas is used for data storage and management. An integrated AI assistant provides real-time code suggestions and debugging support, improving overall productivity.

This paper presents the design and implementation of PairOn, highlighting its features, performance, and effectiveness in creating a unified and efficient collaborative coding environment, along with potential future enhancements.

II. LITERATURE SURVEY

Recent advancements in web technologies, real-time systems, and collaborative tools have significantly

improved the way developers interact and build software together. Research shows that integrating coding, communication, and intelligent assistance into a single platform can greatly enhance productivity and learning. Modern systems focus on real-time synchronization, cloud-based environments, and AI-driven support to simplify development workflows and reduce dependency on multiple tools.

1.Evolution of Collaborative Development Platform

Early development environments were primarily local and isolated, requiring manual sharing of code through version control systems. With the rise of cloud-based platforms, tools such as browser IDEs and remote workspaces enabled developers to collaborate more efficiently. Features like live code sharing and remote debugging improved teamwork, but most platforms still require external communication tools and prior connections between collaborators.

2. Browser-Based IDEs and Code Execution

Modern browser technologies have enabled the development of full-featured IDEs that run entirely online. Platforms like Web Containers allow developers to execute code directly

within the browser without relying on server-side infrastructure. This approach reduces setup complexity and improves accessibility. However, many systems still lack seamless integration with real-time collaboration and advanced development tools.

3. Real-Time Communication and Synchronization

Real-time collaboration relies on efficient communication protocols such as WebSockets, which enable bidirectional data exchange with low latency. Technologies like Socket.IO are widely used to synchronize file changes, cursor positions, and terminal outputs between users. Ensuring consistency, handling conflicts, and maintaining performance are key challenges in building scalable real-time systems.

4. AI in Software Development

Artificial Intelligence has become an important component in modern development environments. AI-powered tools assist developers by providing code suggestions, debugging help, and automated code reviews. Large language models have further improved the ability to generate context-aware responses. However, integrating AI effectively into collaborative systems remains a challenge, especially in maintaining accuracy and usability.

5. Matchmaking and Developer Discovery

Most existing platforms assume that users already know their collaborators. Research indicates a lack of systems that actively connect developers based on skills, interests, and availability. A smart matchmaking mechanism can improve collaboration by pairing compatible users, especially for learning and peer programming scenarios.

6. Data Management and System Architecture

Efficient data storage and management are essential for collaborative platforms. Cloud databases such as MongoDB Atlas provide scalable solutions for storing user data, session information, and interaction history. Real-time systems require optimized architectures to handle concurrent users, maintain session consistency, and ensure fast data retrieval.

7. Evaluation Metrics and System Performance

Technical: Real-time file synchronization latency, AI assistant response time, event delivery reliability, and overall system responsiveness.

System Performance: Ability to handle concurrent users, maintain stable real-time connections, efficient browser-based

code execution, and smooth editor performance.

UX/Adoption: User satisfaction, ease of collaboration, session duration, and engagement levels during coding sessions.

Reliability: System uptime, error handling, data consistency, and scalability under increasing user load.

8. Security and Privacy Considerations

Collaborative platforms must ensure secure handling of user data and project files. Authentication mechanisms such as JWT and encryption techniques are commonly used to protect user sessions. Privacy concerns also arise when sharing sensitive files, making it important to implement secure data handling practices.

9. Challenges and Open Problems

Key research and engineering challenges include:

Real-Time Consistency: Ensuring accurate file synchronization across users without conflicts or data loss during simultaneous edits.

Latency & Performance: Maintaining low-latency communication and smooth performance under varying network conditions and high user load.

AI Accuracy: Ensuring AI-generated code suggestions are correct, relevant, and context-aware to avoid misleading outputs.

Scalability: Handling a growing number of users and sessions while maintaining system stability and responsiveness.

Security & Privacy: Protecting sensitive data such as project files and environment variables while enabling seamless collaboration.

10. Research Gaps and Motivation

The literature highlights the need for a unified platform that combines real-time collaboration, intelligent assistance, and developer discovery in a single environment. Most existing systems address only part of this problem. PairOn aims to fill this gap by integrating matchmaking, browser-based code execution, real-time synchronization, and AI assistance into a cohesive and efficient platform.

III. EXISTING SYSTEM

Several existing developer collaboration platforms provide tools for coding and project development; however, most systems lack integration of real-time collaboration, intelligent assistance, and developer discovery within a single environment.

These limitations result in fragmented workflows, reduced productivity, and dependency on multiple tools for communication, coding, and project management.

- **Replit**

Replit offers a browser-based coding environment with support for multiple languages and basic collaboration features. While it enables code execution without local setup, advanced real-time collaboration and multiplayer features are often limited or restricted. Additionally, it does not provide any intelligent matchmaking system to connect developers based on skills or preferences.

- **GitHub Codespaces**

GitHub Codespaces provides a cloud-based development environment integrated with GitHub repositories. Although it supports remote development, it lacks built-in real-time communication tools such as chat or live interaction. Users must rely on external platforms for collaboration, and the system does not support spontaneous pairing of developers.

- **VS Code Live Share**

VS Code Live Share enables real-time code sharing between developers using the VS Code editor. While it provides efficient collaboration features, it requires local installation and setup, making it less accessible compared to browser-based solutions. Communication is also dependent on external tools like Discord or Microsoft Teams.

Evaluation

The evaluation of existing systems highlights several key limitations. Most platforms lack a unified environment that combines coding, communication, and collaboration in one place. Real-time developer discovery and matchmaking features are absent, requiring users to already know their collaborators. Additionally, dependency on local setups or multiple tools reduces accessibility and efficiency. The absence of integrated AI assistance further limits productivity, making existing solutions less effective for modern collaborative development needs.

IV. PROPOSED SYSTEM

The proposed system aims to develop a real-time collaborative developer platform that enhances coding, learning, and interaction within a single browser-based environment. It integrates modern technologies such as WebContainers for in-browser code execution, Socket.IO for real-time communication, and an AI assistant for intelligent coding support. Built using React 18 and TypeScript for a scalable frontend, along with a Node.js and Express.js backend, the platform provides a seamless and efficient development experience without requiring local installation.

To achieve these goals, the proposed system includes:

- Real-Time Collaborative Coding: The platform enables developers to work together simultaneously using a shared coding environment. File changes, cursor positions, and terminal outputs are synchronized in real time, allowing smooth pair programming without delays or conflicts.
- Browser Based Code Execution: Using Web Containers, the system allows users to run Node.js applications directly within the browser. This eliminates the need for local setup and reduces dependency on server-side execution, making the platform more accessible and scalable.
- AI Assisted Development: An integrated AI assistant provides real-time code suggestions, debugging help, and explanations. This feature improves productivity and supports users in resolving issues quickly, especially beneficial for beginners and learners.
- Smart Matchmaking System: The platform includes a matchmaking engine that connects developers based on programming language, skill level, and session preferences. This enables users to find suitable collaborators instantly without requiring prior connections.
- Integrated Communication and Interface: The system includes built-in chat and collaboration tools within a responsive and user-friendly interface. Developers can communicate, share ideas, and manage tasks without switching between multiple applications

The proposed system combines collaboration, communication, and intelligent assistance into a unified platform, improving efficiency, accessibility, and the overall development experience compared to traditional tools.

V. RESULTS AND DISCUSSION

The developed PairOn platform transforms collaborative software development by integrating real-time coding, communication, and AI assistance into a single browser-based environment. Unlike traditional development tools that require multiple applications, PairOn provides a unified workspace where users can code, collaborate, and interact seamlessly. Built using React 18, TypeScript, and modern web technologies, the platform ensures high performance, smooth navigation, and an efficient development experience without requiring any local setup.

The platform enables real-time collaboration through synchronized file editing, shared terminals, and instant communication powered by Socket.IO. Using WebContainers, it allows developers to execute code directly within the browser, eliminating dependency on server-side execution. The integrated AI assistant supports users with code suggestions, debugging help, and contextual explanations, improving productivity and learning.

The system also includes a smart matchmaking feature that connects developers based on their skills and preferences, making collaboration more accessible and efficient.

The system demonstrates strong performance in terms of responsiveness, usability, and scalability. It significantly reduces the need for multiple tools by combining coding, communication, and project management features into a single interface. However, further improvements can be made in areas such as multi-language support, advanced AI capabilities, and enhanced collaboration features like voice or video communication.

Overall, the project highlights how modern web technologies and AI can transform traditional development workflows into a more interactive, efficient, and collaborative experience, making it suitable for students, developers, and teams working in distributed environments.

VI. CONCLUSION

The PairOn platform successfully demonstrates how modern web technologies can enable a fully functional, real-time collaborative developer environment within a web browser. By integrating WebContainers for in-browser code execution, Socket.IO for real-time synchronization, Monaco Editor for a professional IDE experience, and the Groq API for AI-powered

assistance, the system provides a seamless platform where users can code, communicate, and collaborate without any local setup. This unified environment significantly improves productivity and simplifies the overall development process.

The platform effectively addresses the limitations of existing systems by introducing smart matchmaking for partner discovery, real-time collaboration features, and secure handling of sensitive data. It enhances user experience through instant collaboration, AI-assisted debugging, and an integrated workflow that eliminates the need for multiple tools. Overall, PairOn establishes a scalable, efficient, and user-friendly solution for modern developer collaboration, with strong potential for future enhancements such as video communication, multi-language support, and cloud-based storage.

VII. REFERENCES

[1] Groq. (2024). *Groq API Documentation: LLaMA 3 70B Inference.* console.groq.com/docs.

[2] Socket.IO. (2023). *Socket.IO Documentation: Rooms, Namespaces, and Authentication.* socket.io/docs.

[3] Meta AI. (2023). *LLaMA 3: Open Foundation and Fine-Tuned Chat Models.* arXiv:2307.09288.

[4] Microsoft. (2023). *Monaco Editor Documentation: Getting Started and API Reference.* microsoft.github.io/monaco-editor.

[5] WebContainer API. (2023). *@webcontainer/api — Running Node.js in the browser.* webcontainers.io/docs.

[6] GitHub. (2023). *GitHub Codespaces Documentation.* docs.github.com/codespaces.

[7] Replit. (2023). *Replit Multiplayer: Collaborative Coding Features.* replit.com/site/multiplayer.

[8] StackBlitz. (2022). *WebContainers: Running Node.js natively in the browser using WebAssembly.* StackBlitz Engineering Blog.

[9] Zawacki-Richter, O., et al. (2019). *Systematic review of research on artificial intelligence applications in higher education.* International Journal of Educational Technology in Higher Education, 16(39).

[10] Nichols, D. A., Curtis, P., Dixon, M., & Lamping, J. (1995). *High-latency, low-bandwidth windowing in the Jupiter collaboration system.* Proceedings of UIST '95.