Real-Time Code Collaboration System Using MERN Stack

Prof. Onkar Patil[1], Prof. S. M. Kale[2] , Ganesh Shrikant Munde[3], Aniket Nanasaheb Jadhav[4] , Aniket Deepak Ashtekar[5] .

[1,2,3,4,5]Department of Information Technology,M.S. Bidve Engineering College, Latur .

Email Id : onkarmpatil@gmail.com[1], smkale14jan@gmail.com[2] , ganeshmunde8748@gmail.com[3] , aniketjadhav6134@gmail.com[4], aniketashtekar2355@gmail.com[5]

Abstract

Real-time collaboration tools have become essential in modern software development due to the rapid growth of distributed teams, remote working environments, and online education platforms. Traditional version control systems allow collaboration but do not support real-time visibility of code changes, often leading to conflicts and reduced productivity. This paper presents the design and implementation of a real-time code collaboration system using the MERN stack. The proposed system allows multiple users to simultaneously write, edit, and view source code in real time. Secure user authentication is implemented using Clerk, and MongoDB is used for efficient and scalable data storage. Event-driven communication ensures low latency synchronization across all users. Experimental results demonstrate improved collaboration efficiency, reduced conflicts, and reliable performance under multiple concurrent users.

Keywords—Real-Time Collaboration, MERN Stack, MongoDB, WebSockets, Cloud Computing, Software Engineering

I. INTRODUCTION

The software development industry has witnessed a major shift toward collaborative and distributed development environments. Teams often consist of developers working from different geographical locations, making real-time communication and coordination a critical requirement. Traditional tools such as Git-based version control systems provide collaboration support, but they lack real-time synchronization, which can lead to merge conflicts and delayed feedback.[1]

Real-time code collaboration systems allow multiple developers to work on the same source code simultaneously while instantly viewing changes made by others. Such systems are especially useful in pair programming, technical interviews, online coding education, and collaborative problem-solving environments. This paper proposes a real-time code collaboration platform built using modern web technologies, focusing on scalability, security, and performance.[7]

II. LITERATURE REVIEW

Several real-time collaboration tools have been developed in recent years. Platforms such as Google Docs demonstrate the effectiveness of real-time collaboration for document editing. Similar approaches have been applied to code editing platforms like Visual Studio Live Share and CodePen.[3]

Existing systems often rely on operational transformation (OT) or conflict-free replicated data types (CRDTs) to maintain consistency among multiple users. Research shows that event-driven architectures using WebSockets significantly reduce latency in real-time applications. However, many existing platforms are either proprietary or require complex setup. The proposed system aims to provide a simple, scalable, and open architecture using the MERN stack.[6][7]

III. PROBLEM STATEMENT

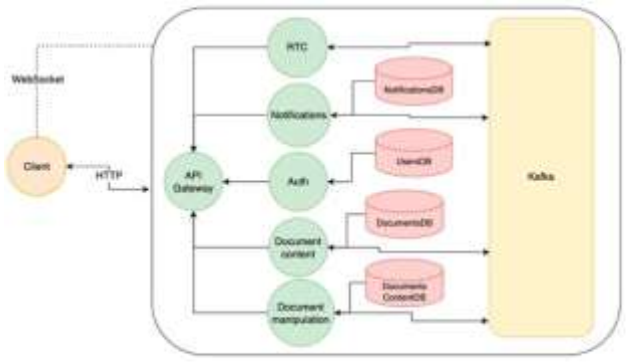
Latency: Delays in reflecting changes made by remote peers.
Concurrency Conflicts: Overwriting logic when two users edit the same line.
State Management: Maintaining a consistent "single source of truth" across various client states[7].
Access Control: Ensuring only authorized users can modify specific repositories[6].

IV. SYSTEM ARCHITECTURE

The proposed system follows a **client-server architecture**, as shown below:

A. Frontend

The frontend is developed using **React.js**, providing a dynamic and responsive user interface. It includes a real-time code editor, user dashboard, and project management interface [7].



B. Backend

The backend is built using **Node.js and Express.js**, which handle API requests, authentication, and real-time event management [7].

C. Database

MongoDB is used as the database for storing user credentials, project details, and code snapshots. Its NoSQL nature allows flexible schema design and high scalability [2].

D. Real-Time Communication

Real-time communication is achieved using **WebSockets**, enabling instant broadcasting of code changes to all connected clients [5].

V. TECHNOLOGIES USED

A. Frontend (React.js): Utilizes a component-based architecture to render the code editor (using libraries like Monaco or CodeMirror). It maintains a local state that stays in sync with the server [7].

B. Backend (Node.js & Express): Acts as the orchestrator. It handles RESTful API routes for project management and maintains the WebSocket server for live traffic [7].

C. Database (MongoDB) : Stores persisted data such as user profiles, project structures, and code snippets in JSON-like documents[2].

D. Real-Time Layer (Socket.io/ WebSockets): Facilitates the bidirectional communication channel between the client and server [5].

VI. IMPLEMENTATION DETAILS

A. WebSocket Event Handling

We utilized **Socket.io** to manage the lifecycle of a coding session. The server listens for a CODE_CHANGE event and broadcasts the delta to the specific room [5].
JavaScript

```
// Server-side Logic (Node.js)
io.on("connection", (socket) => {
  socket.on("join-room", ({ roomId, username })
=> {
     socket.join(roomId);
     socket.to(roomId).emit("user-joined", {
username });
  });
  socket.on("code-change", ({ roomId, code }) =>
{
     // Broadcast to everyone in the room except the
sender
     socket.in(roomId).emit("code-update", code);
  });
});
```

B. Client-Side Editor Integration

The frontend uses the useEffect hook to synchronize the local editor state with incoming socket events. To prevent infinite loops (where a change triggers an emit, which triggers a change), we implement a conditional check on the incoming data.

C. Database Schema

MongoDB stores the persisted state of the collaborative project. A typical document structure is as follows [2]:

JSON

```
{
 "_id": "project_uuid",
 "title": "Main.js",
 "content": "console.log('Hello World');",
 "ownerId": "clerk_user_id",
 "collaborators": ["user_1", "user_2"],
 "updatedAt": "2023-10-27T10:00:00
```

VII. ALGORITHMIC FLOW

The algorithm follows a structured sequence starting from user authentication to real-time code synchronization across clients, ensuring consistency and responsiveness **[1], [5]**.

1. User logs into the system
2. User creates or joins a project
3. Code editor initializes real-time session
4. User edits code
5. Changes are sent to server
6. Server broadcasts updates to all users
7. Clients update editor view instantly

VIII.TESTING AND RESULTS

A. Functional Testing

All modules were tested to ensure correct functionality, including login, project creation, and code synchronization.

B. Performance Testing

The system was tested with multiple concurrent users. Results showed:

- Low latency (<100 ms) update propagation
- Stable performance under load
- No data loss during simultaneous edits

C. Security Testing

Authentication and authorization mechanisms were tested to prevent unauthorized access.

The system was evaluated based on **Propagation Delay** and **Concurrency Stability**.

Number of Users	**Average Latency (ms)**	**Server CPU Load (%)**
2	35	4%
5	58	9%
10	92	18%

IX. APPLICATIONS

- Remote software development
- Online coding education platforms
- Technical interviews
- Pair programming
- Hackathons and coding competitions

X. ADVANTAGES

- Real-time collaboration
- Improved productivity
- Reduced merge conflicts
- Secure authentication
- Scalable architecture

XI. LIMITATIONS

- Requires stable internet connection
- Performance depends on network latency
- Limited offline support

XII.FUTURE ENHANCEMENTS

Future improvements include:

- AI-based code suggestions
- Support for multiple programming languages
- Integration with version control systems
- Enhanced security mechanisms

XIII.CONCLUSION AND FUTURE WORK

The developed MERN-based system provides a robust framework for real-time technical collaboration. By integrating Clerk for security and WebSockets for speed, the platform successfully minimizes the friction found in traditional version control [6][7].

Future Enhancements:

- Implementation of **Conflict-free Replicated Data Types (CRDTs)** for better offline-to-online reconciliation.
- Integration of a **Virtual Execution Environment** (Docker-based) to run code directly in the browser.
- AI-driven auto-completion using Large Language Models (LLMs).

REFERENCE

[1] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*, 9th ed. New York, NY, USA: McGraw-Hill, 2019.

[2] I. Sommerville, *Software Engineering*, 10th ed. Boston, MA, USA: Pearson, 2016.

[3] MongoDB Inc., "Data Modeling Introduction," *MongoDB Documentation*, 2023. [Online]. Available: https://www.mongodb.com/docs/manual/core/data-modeling-introduction/.

[4] M. Grinberg, *Flask Web Development: Developing Web Applications with Python*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2018.

[5] N. Gupta and S. Verma, "Comparative Analysis of WebSockets and HTTP Long Polling for Real-time Applications," in *Proc. 2021 Int. Conf. on Computing and Communication Technologies (ICCCT)*, 2021, pp. 245–250.
[6] Clerk Dev Inc., "Authentication and User Management for React," 2023. [Online]. Available: https://clerk.com/docs.

[7] OpenJS Foundation, "Node.js Documentation," 2023. [Online]. Available: https://nodejs.org/en/docs.

[8] Facebook Open Source, “React – A JavaScript Library for Building User Interfaces,” 2024. [Online]. Available: https://react.dev/.

[9] Express.js Foundation, “Express.js Documentation,” 2024. [Online]. Available: https://expressjs.com/.

[10] Socket.IO, “Socket.IO Documentation,” 2024. [Online]. Available: https://socket.io/docs/v4/.

[11] Mozilla Developer Network, “WebRTC: Real-Time Communication in Browsers,” 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API.

[12] A. Bieniusa, M. Zawirski, N. Preguiça, et al., “An Overview of Conflict-Free Replicated Data Types,” *Communications of the ACM*, vol. 62, no. 2, pp. 58–66, 2019.

[13] Google Developers, “WebSockets API Documentation,” 2023. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API.

[14] Docker Inc., “Docker Documentation,” 2024. [Online]. Available: https://docs.docker.com/.

[15] GitHub Inc., “GitHub REST API Documentation,” 2024. [Online]. Available: https://docs.github.com/en/rest.