Distributed Systems - IT559

Real-Time Collaborative Document Editor (LiveDocs)

❖ **Team-ID : 4**

Jaimin Prajapati	- 202201228
Dhrudeep Sharma	- 202201150
Het Gandhi	- 202201167

Prof. Amit Mankodi

❖ Problem Statement

➔ In modern collaborative environments, multiple users need to edit documents simultaneously in real time, regardless of their location. Traditional text editors fail to provide seamless multi-user collaboration, leading to issues such as inconsistent document states, conflicts due to concurrent edits, and system failures causing data loss.

❖ System Overview

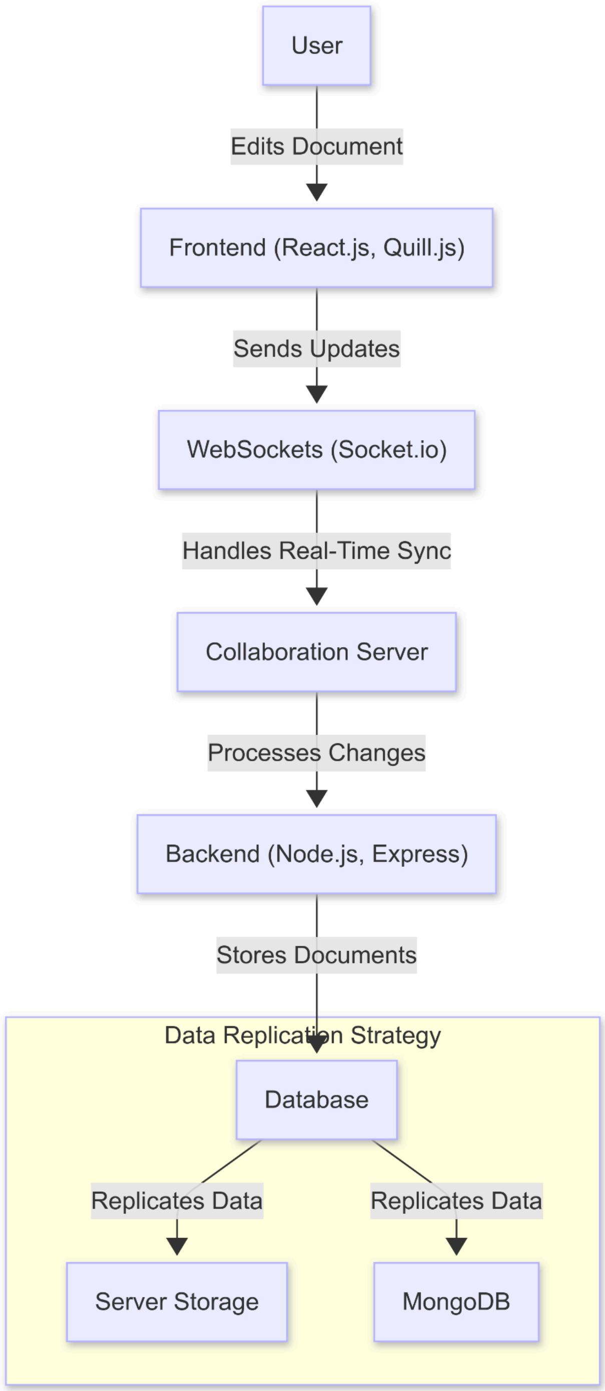
➔ The proposed real-time collaborative document editor allows multiple users to create, edit, and share documents concurrently from any location. Key features include:

1. **Real-Time Editing:** Utilizes WebSockets to broadcast changes instantly to all connected clients.

2. **Distributed Storage:** Uses replication and distributed storage techniques to ensure data availability and persistence.

3. **Concurrency Control:** Achieved implicitly by the real-time update mechanism in Socket.IO .
4. **Security and Access Control:** Offers role-based document sharing permissions.

❖ System Architecture and Components



❖ Distributed System Concepts Applied

1. **Real-Time Synchronization:**

   Every change is propagated in real time across all clients, ensuring that users see the most up-to-date document state instantly.

2. **Concurrent Multi-User Collaboration:**

   The system is engineered to allow multiple users to edit the same document simultaneously.

3. **Replication:**

   Data is consistently replicated across multiple nodes to ensure both high availability and data integrity. This redundancy also safeguards against data loss in case of individual server outages.

4. **Fault Tolerance:**

   By leveraging robust replication and automatic failover mechanisms, the system continuously maintains backups, ensuring that service remains uninterrupted even when individual components fail.

5. **Access Control:**

   It gives role-based permission to strictly manage who can view or edit documents, protecting sensitive information and ensuring that only authorized users interact with the data.

❖ Implementation Details

Frontend:

- **React.js with TypeScript** — Strongly-typed UI development.
- **Quill.js** — Rich-text editor.
- **Docx + FileSaver.js** — Exporting editor content as `.docx` files.
- **Css frameworks**— For responsive UI and modals.

Backend:

- **Node.js** with **Express** — Server-side logic.
- **Socket.io** — Real-time bi-directional communication.

Real-time Collaboration:

- **WebSockets** via **Socket.io** — Used to sync documents between users live

Programming Language:

- **TypeScript** — Used on both frontend and backend for type safety and better developer experience.

Database:

- **MongoDB** — For persistent document storage.

APIs and Libraries:

- **Socket.IO API:** Used to emit and listen to events between server and client.
- **Quill API:** Provides a robust interface for document editing and handling text-change events.
- **docx API:** Converts text content to Microsoft Word DOCX format.

❖ Challenges Faced and Solutions

1. Real-Time Synchronization and Concurrency

- **Challenge:**
  Multiple users editing the same document at once could lead to conflicting changes.
- **Basic Solution:**
  - **Socket.IO (Backend & Frontend):**
    Used to broadcast and receive changes in real time.

2. Replication and Fault Tolerance

- **Challenge:**
  Keeping all document copies updated across multiple servers can be difficult, especially during frequent edits or server failures.
- **Basic Solution:**
  - Replicate data to multiple servers and use automatic synchronization to ensure consistency and availability.

❖ Results and Performance Analysis

Please refer to the Demo section of the submitted zip file.

❖ Future Improvements and References (if any)

1. **Enhanced Collaboration Tools:**

   Future versions aim to enhance collaboration and user experience by integrating rich media support (images, videos), real-time commenting, undo/redo functionality, multi-user cursor tracking, and advanced text formatting options.

2. **AI-Powered Assistance:**

   Incorporate machine learning for real-time grammar checking, document summarization, and predictive text suggestions.

3. **Security Upgrades:**

   Implement stricter user authentication to enhance security and protect against unauthorized access.

THANK YOU