Project Proposal

Team Members:
Sai Navyanth Penumaka (sp8138)
Geethika Rao Gouravelli (gg2879)
Sai Sandeep Mamidala (mss9430)

Project Title

ShareNote – Collaborative Real-Time Text Editing Platform

Abstract

ShareNote is a real-time collaborative text editor designed to allow multiple users to work simultaneously on shared documents through a secure and interactive Java-based application. The system focuses on showcasing advanced Java concepts such as multithreading, client-server networking, database integration, and GUI development. The primary goal is to simulate the core functionalities of popular collaborative tools, such as Google Docs, while demonstrating deep technical understanding of distributed systems and synchronization challenges.

ShareNote enables users to create, share, and co-edit documents while maintaining data consistency and providing features like role-based access, version control, and live cursor tracking. The application will also ensure smooth performance under concurrent edits, robust conflict resolution, and recovery in case of system failures.

Key Features

1. **User Authentication and Role Management**
   - Secure user registration and login using hashed credentials.
   - Role-based permissions such as owner, editor, and viewer.
   - Session management for authenticated access.
2. **Document Management System**
   - Create, open, edit, delete, and save documents.
   - Each document is stored in a database with metadata such as creator, timestamp, and collaborators.
   - Auto-save mechanism to prevent loss during unexpected shutdowns.
3. **Document Sharing and Collaboration**
   - Document owners can share access with other users (read-only or edit permissions).

- Shared documents appear in collaborators' dashboards automatically.
- Supports multiple simultaneous editors with minimal latency.

4. **Real-Time Editing and Synchronization**
   - Instant propagation of changes between connected clients using socket-based communication.
   - Live cursor tracking to visualize where each collaborator is typing.
   - Conflict resolution mechanisms to handle overlapping edits gracefully.
5. **Version History and Restoration**
   - Every save operation creates a version snapshot.
   - Users can view or revert to previous document versions.
   - Maintains complete change history for transparency and auditing.
6. **Graphical User Interface**
   - Modern, intuitive, and responsive user interface for seamless editing.
   - Real-time updates reflected dynamically without requiring page reloads.
   - Displays collaborator presence, document state, and system notifications.

Challenges

1. **Consistency Management:** Maintaining uniform document state across all active sessions during concurrent edits.
2. **Network Delays and Failures:** Handling temporary connectivity issues or high-latency networks without losing updates.
3. **Conflict Resolution:** Detecting and resolving edit conflicts to prevent overwriting user inputs.
4. **Concurrent Database Access:** Managing simultaneous read/write operations from multiple clients.
5. **Scalability and Performance:** Ensuring efficient synchronization and responsive interface even as user count increases.

Advanced Concepts to Meet Project Requirements

1. **Multithreading:**
   - Support multiple users by maintaining separate threads for each client connection.

- Coordinate parallel document modifications from different collaborators efficiently.
- Execute background processes for saving, synchronizing, and broadcasting updates without interrupting users' ongoing edits.

2. **Networking:**
   - Follows a client-server model using Java Sockets.
   - WebSocket-like communication ensures continuous and bidirectional data flow.
   - RESTful API endpoints are used for document CRUD operations.
3. **Database Integration:**
   - Uses a database for persistent data storage.
   - Stores user credentials, documents, version logs, and permissions.
   - Handles concurrent updates using transactions and synchronization.
4. **Graphical Interface Development:**
   - Clean, interactive, and user-friendly interface for seamless collaboration.
   - Event-driven design to reflect updates instantly as users edit shared documents.

Example Scenario: How ShareNote Works

Step 1: **User Registration:** Two users, *Alice* and *Bob*, register and log into the ShareNote platform.

Step 2: **Creating a Document:** Alice creates a new document titled *"Team Notes"*. The document is automatically stored in the database with Alice as the owner.

Step 3: **Sharing and Permissions:** Alice shares the document with Bob, granting him **edit access**. The document now appears in Bob's dashboard under "Shared with Me."

Step 4: **Collaborative Editing:**

- Both Alice and Bob open the document simultaneously.
- Alice writes "Meeting at 10 AM," and Bob immediately sees it appear in real time.
- Bob adds "Agenda: Project updates," and Alice sees the change instantly.
- Live cursors display who is editing which section.

Step 5: **Version Control:**

- Each edit triggers a version checkpoint.
- Later, Bob reverts the document to a previous version to undo accidental changes.

Step 6: Final Save: Once finished, the latest state is stored in the database. Both users can download or revisit it anytime.

Conclusion

ShareNote encapsulates the essence of a real-time distributed system through the lens of Java programming. It demonstrates mastery over threads, sockets, databases, and graphical interface development while tackling synchronization and consistency challenges central to collaborative applications.