

REAL-TIME COLLABORATIVE TEXT EDITOR

Presented by Team 43

Ashutosh Srivastava (2021101056)

Ishit Bansal (2021101083)

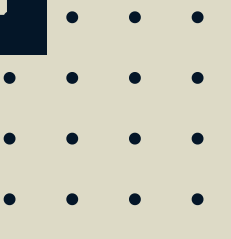


Ashish Chokhani (2021102016)



INTRODUCTION



In today's world, seamless collaboration across remote teams is vital. Our project aims to build a real-time, peer-to-peer collaborative text editor that functions effectively even with limited or no internet connectivity. Using CRDTs (Conflict-Free Replicated Data Types) and WebRTC, we ensure that edits from multiple users converge consistently without conflicts, offering a smooth and robust editing experience.



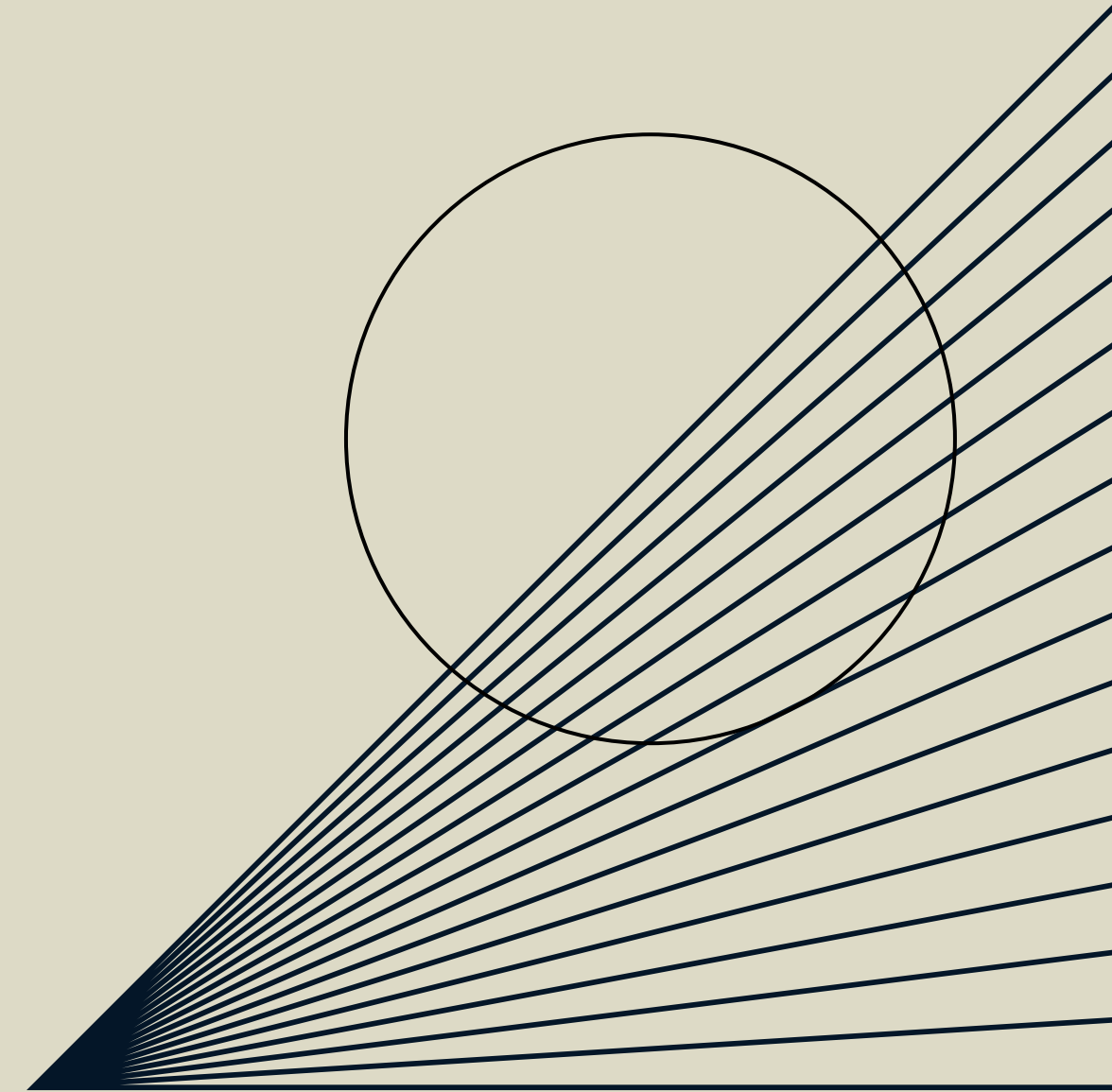


✦ MOTIVATION

Traditional collaborative editing systems rely heavily on central servers, which introduces latency, dependency, and synchronization issues.

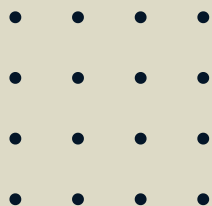
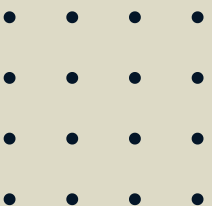
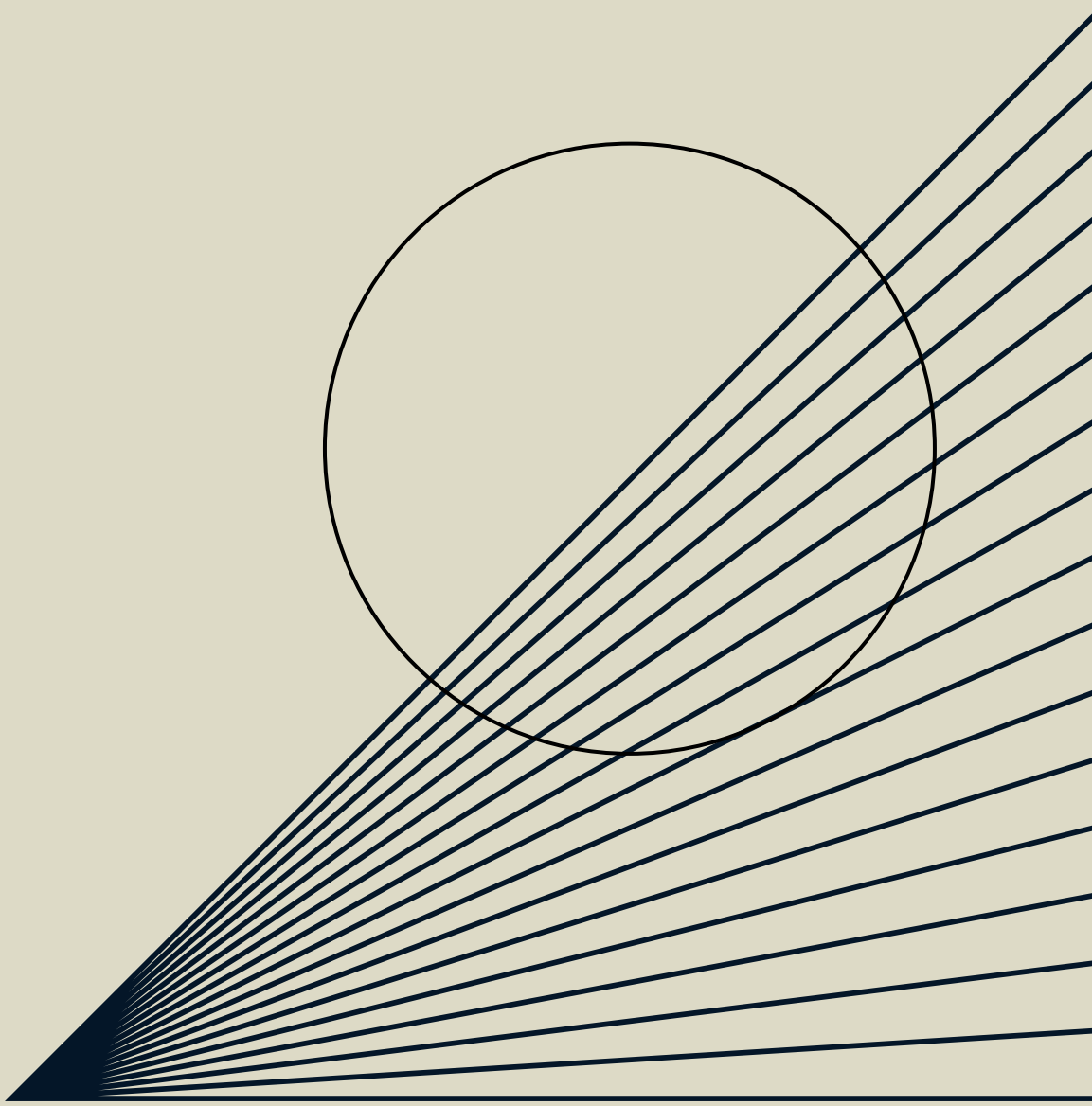
These systems often struggle with merge conflicts and manual resolution, reducing productivity.

Our motivation is to create a system that eliminates these issues using distributed system principles, allowing users to collaborate effortlessly in real-time, with built-in resilience to network delays and offline usage.





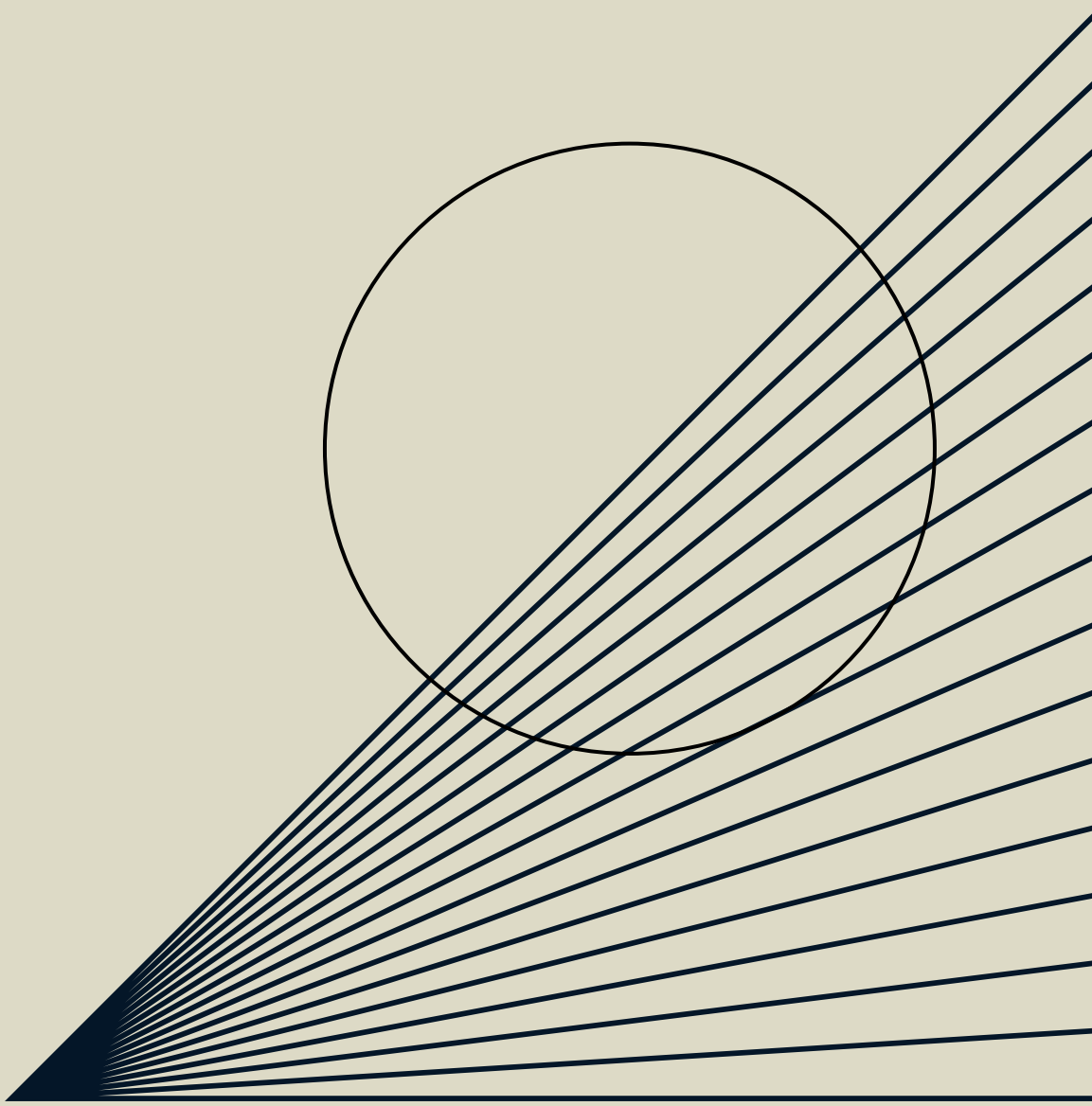


◆ OBJECTIVES

- **Real-Time Synchronization:** Edits by one user should be instantly visible to all other users.
 - **Conflict Resolution:** Automatic merging of concurrent changes through the CRDT paradigm, preventing data loss.
 - **Offline Editing & Sync:** Let users continue editing without a network connection, then reconcile changes once reconnected.
 - **Undo Mechanisms:** Provide intuitive user features to revert document states.
- 
- 
- 



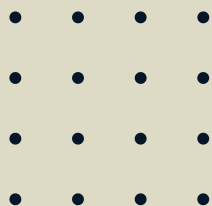
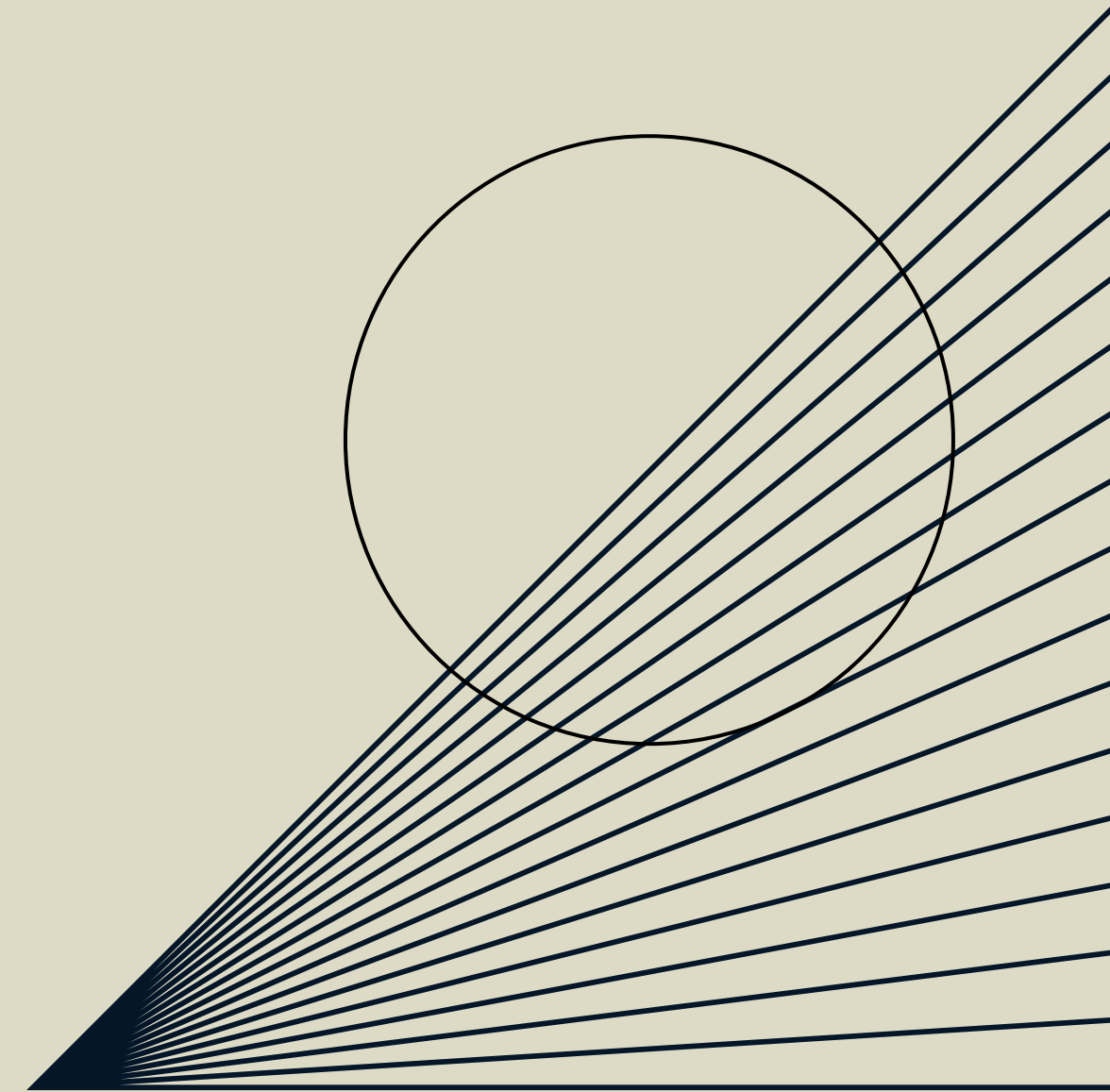
◆ OBJECTIVES

- **Access Control:** Restrict read/write access to authorized users only.
 - **Rich Text Formatting:** Support for basic formatting (bold, italic, lists, etc.)
 - **Export Options:** Users can download documents as PDF or Markdown, ensuring flexibility.
 - **Performance Testing:** Conduct scale testing to ensure system performance with multiple simultaneous clients.
- 
- 
- 



◆ SYSTEM ARCHITECTURE

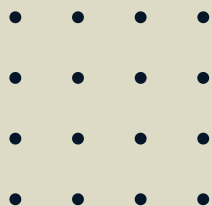
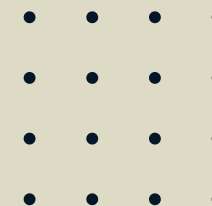
- **Frontend:** Built using React and Quill.js, providing a rich and modular text editing experience.
- **Communication:** Real-time collaboration is enabled through WebRTC, and data consistency is ensured via Yjs (CRDT engine).
- **Offline Storage:** Uses IndexedDB to store operations locally during disconnections.





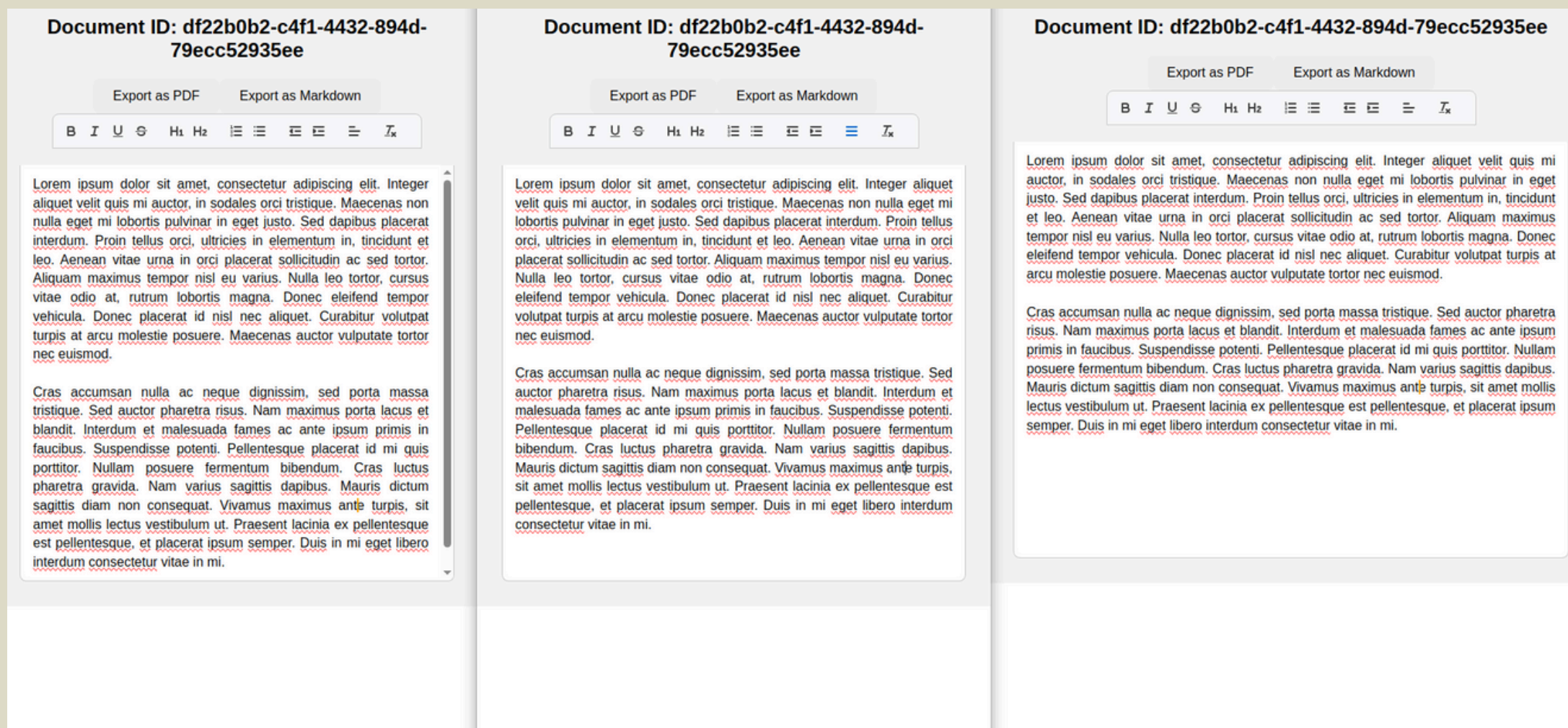
✦ CRDT – YATA ALGORITHM

The YATA (Yet Another Transformation Algorithm) is a CRDT specially designed for text editing. It models the document as a doubly linked list where each character has a globally unique ID. Key benefits include:

- **Operational consistency** through deterministic merging
 - **Idempotent operations:** No duplication even if an update is received multiple times
 - **Efficient performance:** $O(\log(H))$ for insert/delete operations
 - Supports undo/redo while preserving concurrent user changes
- 
- 



REAL-TIME SYNC – WEBRTC



We employ WebRTC to establish direct, encrypted peer-to-peer connections between collaborators. The initial handshake is managed by a signaling server, after which peers communicate independently. Benefits include:

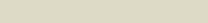
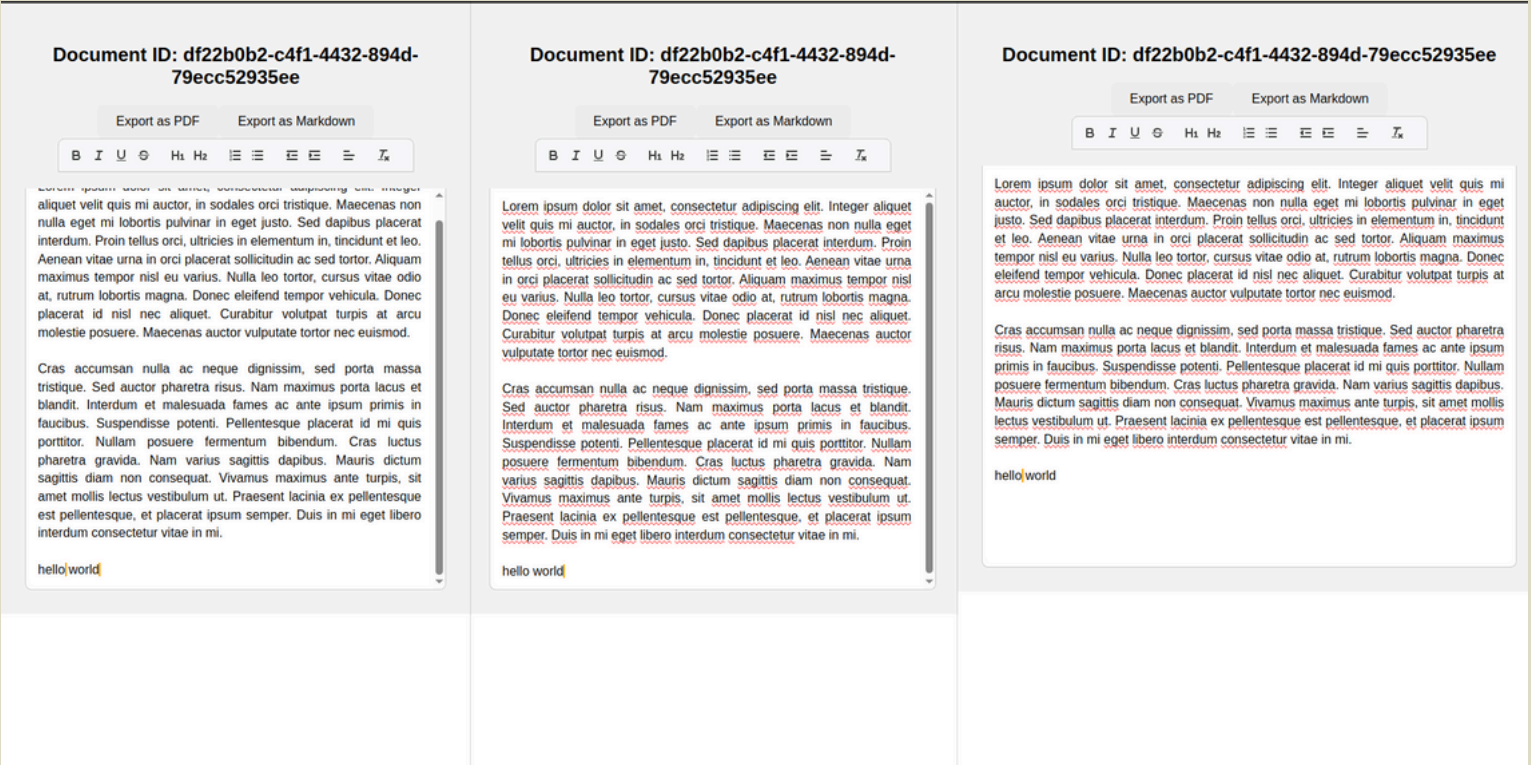
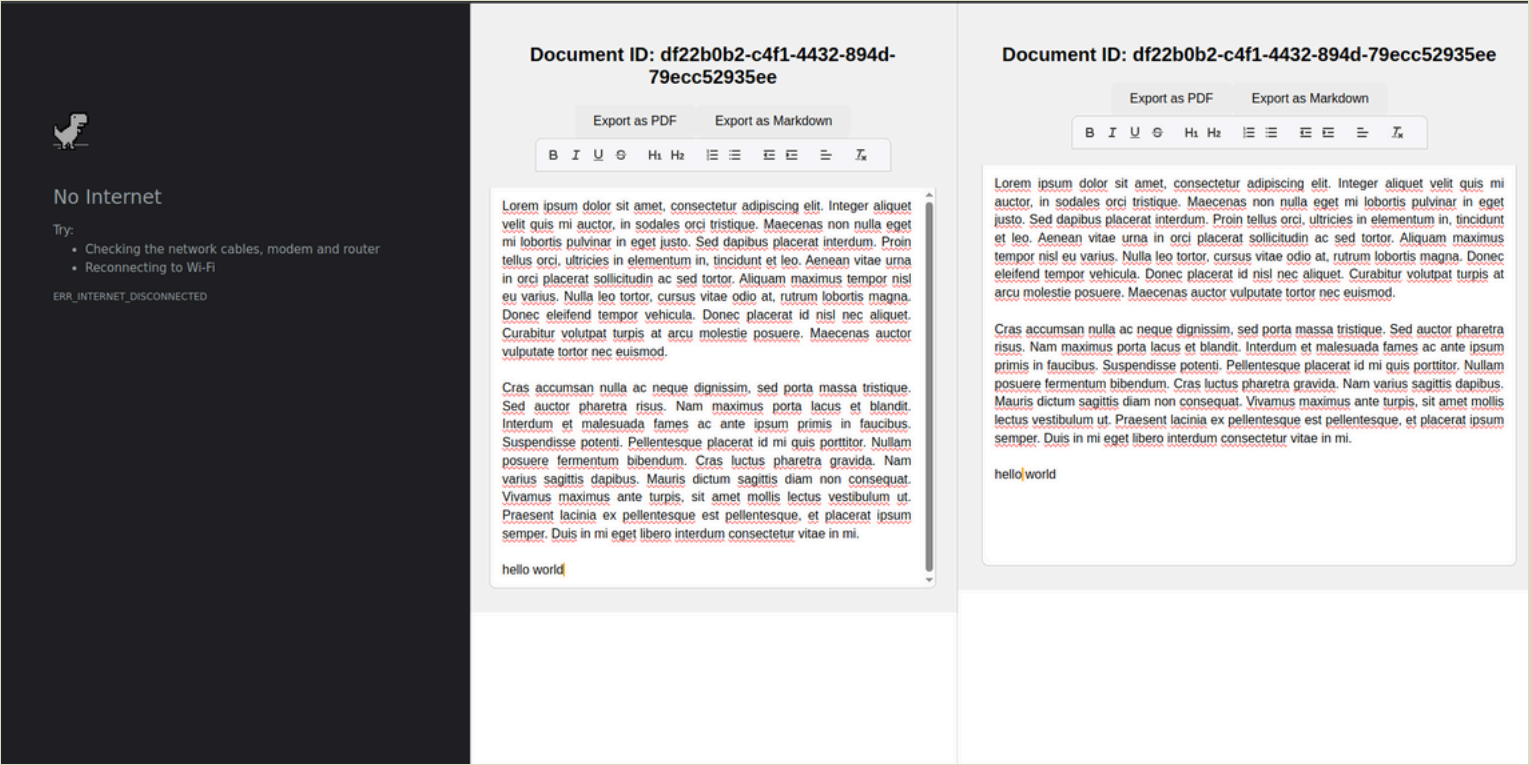
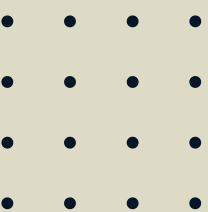
- Lower latency and faster sync
- Enhanced privacy since content isn't routed through central servers
- Fault tolerance, removing the single point of failure
- Scalability by reducing server workload



◆ OFFLINE EDITING & SYNC

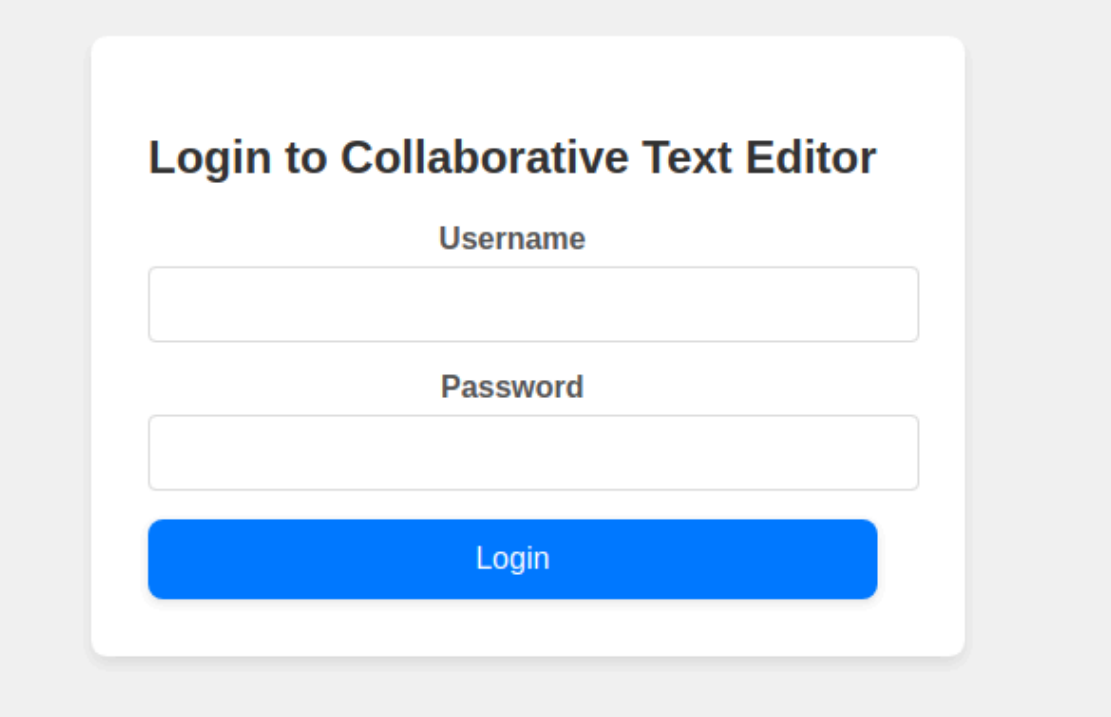
Our editor supports offline-first behavior. Users can continue working without internet, and their edits are saved in IndexedDB:

- Upon reconnection, a merge queue sends local operations to peers.
- CRDT ensures conflict-free merging, even when many users make simultaneous offline changes.
- No manual syncing or version conflict resolution is needed—everything is automatically reconciled.

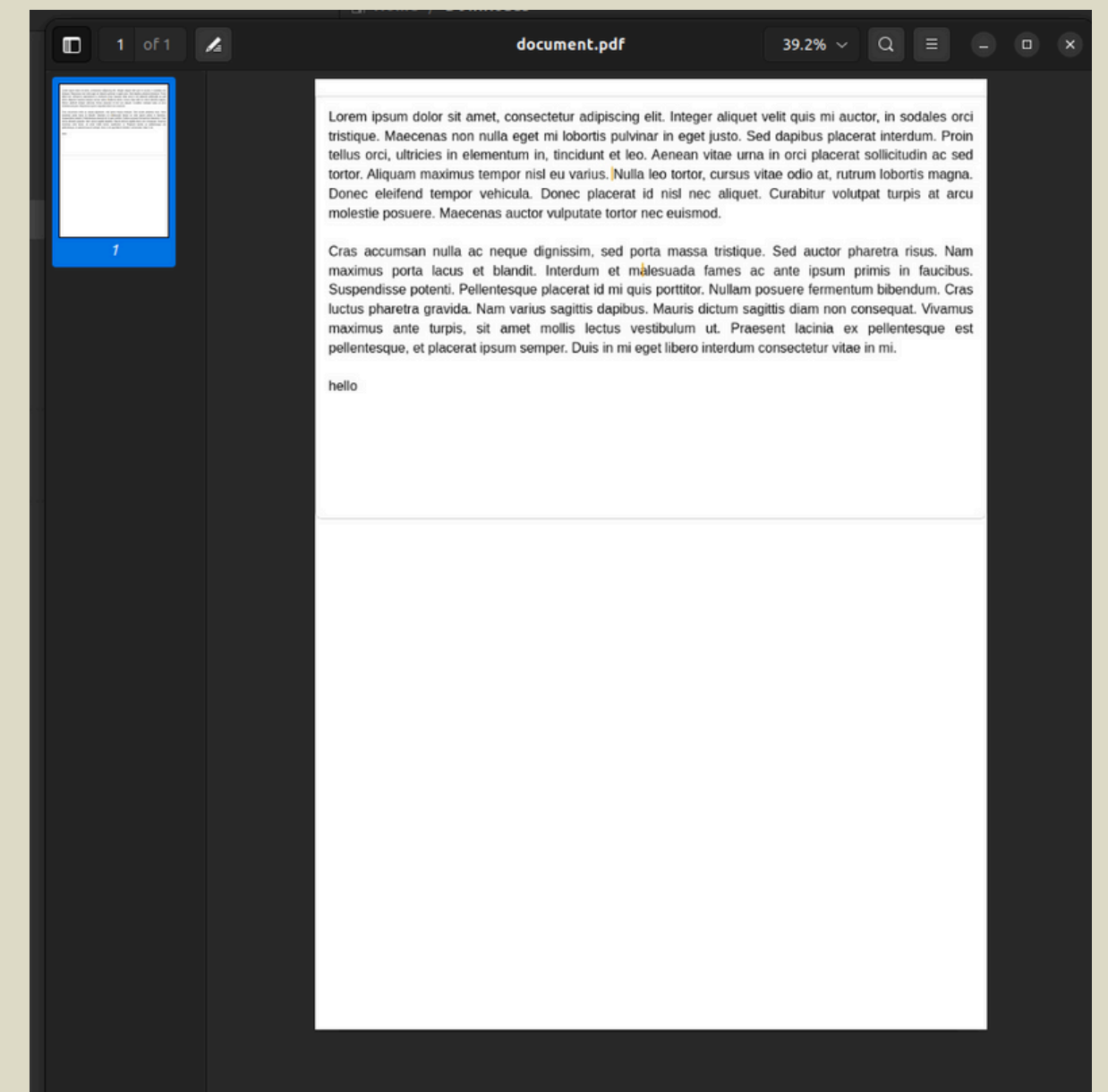


✦ AUTHENTICATION & EXPORT FUNCTIONALITY

- A simple login system is implemented using a users.json file for credential verification, with the authenticated username stored in localStorage.
- The login check ensures that only authenticated users can access the editor.
- Export options are available within the editor.



A login form titled "Login to Collaborative Text Editor". It features two input fields: "Username" and "Password", each with a light gray border and a small eye icon on the right. Below the fields is a solid blue button with the text "Login" in white. The form is centered on a light gray background.



◆ SCALE TESTING – CPU & MEMORY USAGE

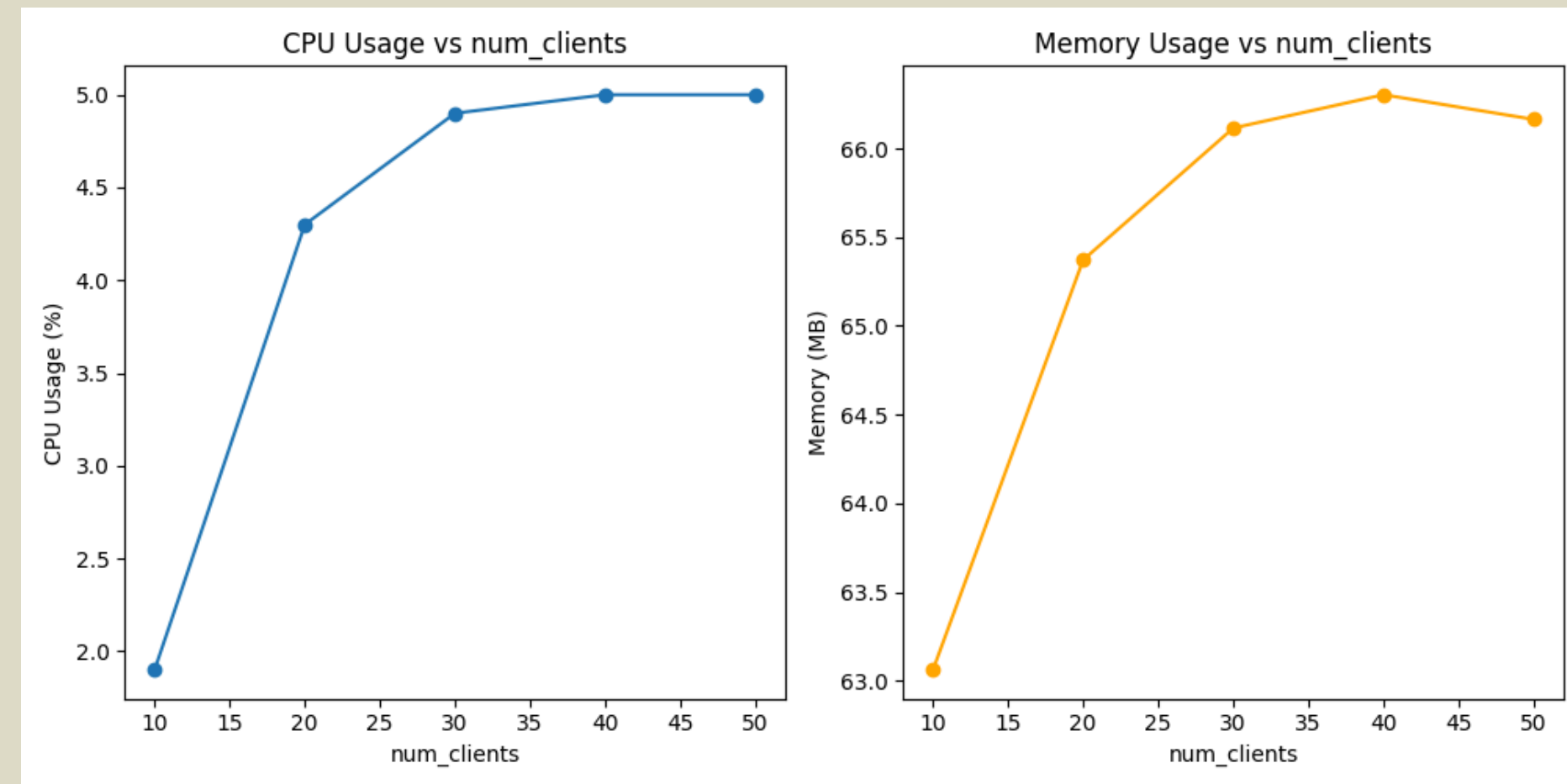
To assess scalability, we simulated 10 to 50 concurrent clients using a Node.js + Puppeteer script.

Setup:

- Clients opened collaborative sessions via Puppeteer.
- System resource usage was monitored using psutil for 10 seconds per test.
- Metrics tracked: CPU (%) and Memory (MB).

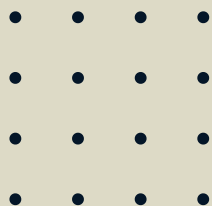
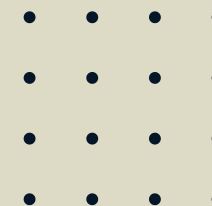
Key Observations:

- CPU Usage scaled up smoothly, peaking around 5% at 50 clients.
- Memory Usage increased to ~66 MB, with minor variance beyond 40 clients.
- System remained responsive and stable, indicating good efficiency for mid-scale use.





✦ CHALLENGES

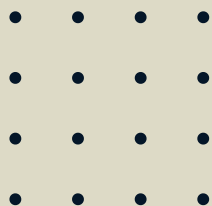
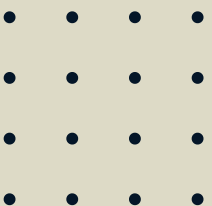
- **Merging Concurrent Edits:** Designing a custom CRDT that preserves user intent was complex.
 - **Undo in P2P Context:** Avoiding unintended rollbacks while maintaining collaborative consistency.
 - **WebRTC Debugging:** Setting up stable peer connections required fine-tuning signaling and ICE configurations.
 - **Offline Edit Queues:** Ensuring correct order and replay logic for queued ops when users come back online.
- 
- 



✦ CONCLUSION & FUTURE WORK

This project successfully demonstrates a working prototype of a distributed real-time text editor. Using CRDTs, peer-to-peer networking, and offline synchronization, we achieved our goal of building a fault-tolerant, responsive, and scalable system.

Future enhancements:

- Add commenting and annotations
 - Implement version history and rollback
 - Extend mobile and tablet compatibility
 - Use JWT and hashed credentials for production-grade security
- 
- 

THANK
YOU

