Cairo University
Faculty of Engineering
Credit Hour System

APT Project

Team W-21

Team Members:
Ahmed Sameh Salah - 4230138
Youssef Ahmed AbdelMoneim - 1220211
Mahmoud Mohamed Attia - 4230175
Mohamed Ashraf Mohamed - 4230167

Date: 5/05/2025

Introduction

This report outlines the primary weaknesses identified in the current Java-based collaborative plain text editor when deployed at large scale. It highlights architectural limitations, performance bottlenecks, security vulnerabilities, reliability gaps, and deployment challenges. Addressing these issues is crucial to ensure robustness, scalability, and maintainability in production environments.

Scaling Problems

- **Single Server Only:** All user connections go through one WebSocket server. If many people use the editor, that one server will slow down or crash.
- **Sessions Stored in Memory:** We keep track of who is editing in the server's RAM. That means we can only handle as many users as the server's memory allows, and we can't spread users across more servers.
- **Fallback to Memory on Database Error:** If the database (MongoDB) is down, we save edits in memory only. That risks losing data if the server stops.

Speed and Efficiency Problems

- **Slow CRDT Method:** We use a tree-based method (TreeSet) to keep text in order. This gets slow when documents get big.
- **Sending Whole Document on Every Change:** Each time someone types, we send the entire document to all users instead of just the new text. This wastes network and delays updates.
- **Small Undo History:** We only remember the last 50 changes. If users work for a long time, they can't undo older edits.

Security Issues

- **Database Passwords in Code:** The MongoDB login details are written directly in the code. Anyone with access to the code can see them.
- **No Encryption on Connections:** WebSocket connections are not protected by TLS/SSL, so someone could intercept or change data in transit.
- **Basic Permissions:** We only have “editor” or “viewer” rights. There’s no way to give more detailed access, like editing only parts of a document.

Reliability Problems

- **Simple Error Handling:** When errors happen, we just log them. We don’t try to fix or retry operations, so problems can go unnoticed.
- **No Reconnect Logic:** If a user’s internet drops, the editor doesn’t try to reconnect smoothly or save unsent edits.
- **Manual Conflict Fixes:** Our method for merging changes can fail when two people edit the same place at the same time.

Deployment Issues

- **Hardcoded Paths in Scripts:** The startup script (run_server.bat) uses a fixed folder path (e.g., D:\APT), so it won’t work on other machines easily.
- **No Docker Support:** Without containers, installing and updating the editor on different servers is error-prone.
- **No Monitoring:** We don’t collect any data on how the editor performs in production, so we can’t spot issues before users do.

Suggestions

1. Move to a stateless server setup with multiple instances and use a shared session store (like Redis).
2. Improve the CRDT so it handles big documents faster and only sends small changes (deltas) over the network.
3. Store passwords and other settings outside the code (e.g., in environment variables or a secret vault) and turn on TLS/SSL.
4. Add retry logic, automatic reconnection, and better user messages when things go wrong.
5. Use Docker for easy, consistent deployments and add monitoring tools (e.g., Prometheus, Grafana) to watch server health.

Conclusion

By fixing these issues, our collaborative text editor will be faster, safer, and more reliable for large groups of users.