
# NoteSpace

## Multi-platform Web Application for Real-Time Document Collaboration & Sharing

Guilherme Ferreira, n.º 49472, e-mail: a49472@alunos.isel.pt, tel.: 916682543
Ricardo Costa, n.º 49511, e-mail: a49511@alunos.isel.pt, tel.: 967322385

Supervisors: Paulo Pereira, e-mail: paulo.pereira@isel.pt

March 2024

## Introduction

For a long time, project collaboration often relied on the exchange of local copies on each version. However, with the emergence of Git, a distributed version control system, a significant shift was made, by enabling asynchronous collaborative work while tracking each user's changes. Still, this approach lacked the possibility to work synchronously on the same documents.

Then, Google Docs emerged as a groundbreaking solution to the challenges posed by traditional project collaboration methods. With its real-time collaboration features, multiple users could now edit the same document simultaneously from anywhere with an internet connection, across multiple devices.

What we aim to achieve, is a sort of combination between Google Docs and GitHub, combining the best of both worlds: the **real-time collaboration, document organization** and **rich text editing** of Google Docs, with the **version control, conflict resolution** and **public sharing of documents** of GitHub.

## Objectives

The goal of this project is to develop not just a robust collaborative text editor that enables real-time collaboration among multiple users, while ensuring data consistency and minimizing conflicts, but a platform that enhances teamwork and organization, featuring the following user stories:

- **Rich text editor:** to create and edit documents with formatted text, markdown support and other multimedia elements;
- **Real-time collaboration:** to edit documents simultaneously with other users, with all eventually reaching the same consistent state;
- **Workspace management:** to organize documents in folders within workspaces;
- **Note sharing:** to share public or private documents with others or publish them on the platform for others to view and fork;
- **Cross-platform compatibility:** to access and edit documents from any device;
- **Offline functionality:** to access and edit documents even when not connected to the internet or with an unstable internet connection;
- **Revision history:** to track changes and revert to previous document versions;
- **Canvas** (Optional): to better express ideas through drawings and diagrams.

## Why NoteSpace?

There are a lot of collaborative text editing platforms out there, but many share the same limitations. Most of them lack an explicit version control system, offline editing, public sharing of document repositories and restrict resource sharing through links.

Overall, NoteSpace is more than just a text editing platform. It's a comprehensive solution that empowers seamless collaboration and productivity, allowing users to share their documents with everyone, in a GitHub-like environment.

## Research

Developing a live collaboration platform brings many challenges, mainly regarding managing concurrent edits effectively to ensure data consistency across all users. This requires a mechanism to do so - a conflict resolution algorithm. After some research, we have come across two main types of algorithms for conflict resolution:

- **Operational Transformation (OT)**, where operations are transformed when a conflict is detected, by a centralized server, depending on what order they arrive. This, however, requires hoisting state in the server and limits clients into communicating through the same server instance, creating a bottleneck.
- **Conflict-Free Replicated Data Type (CRDT)**, where it is claimed that to avoid conflicts, operations must be commutative and idempotent, so that the same state is always reached, independently of the order operations. Rather than implementing a complex algorithm to handle conflicts, it uses a more complex data structure to do so. This also allows users to communicate with multiple server instances.

Since the CRDT algorithm is more scalable and changes sent to the server by offline clients can be arbitrarily delayed, we decided this was a more viable option. More specifically, we needed an algorithm that was fully robust for offline editing. A common anomaly that occurs in both OT and most CRDT algorithms is called "interleaving", that happens when two users concurrently or in an online/offline scenario insert text at the same position in the document. The merged outcome may interleave the inserted text, resulting in an unreadable state. To fix that, we will implement a known CRDT algorithm that maximizes non-interleaving, called Fugue. Fugue implements a unique mechanism that ensures the correct merging of concurrent edits, thereby maintaining data integrity even in complex editing scenarios. It relies on a tree data structure, where each character is considered a node in the tree, which allows for systematic comparison and resolution, ensuring conflicts are addressed granularly.

## Architecture

The system architecture will be composed by the following components:

- **Single Page Application:** implemented in TypeScript and React, using the Vite bundler, as a progressive web app (PWA), for multi-platform and offline support;
- **Web API:** implemented in TypeScript and Express, using HTTP and WebSockets with Socket.io;
- **PostgreSQL Database:** for storing relationships between entities and additional data;
- **Firestore Database:** for storing the documents' contents and versions (changes);

Both the front-end and the back-end servers will be powered by the Node.js runtime environment, and the deployment will be done using Docker.

## Challenges Ahead

There will be multiple challenges to overcome in this project, such as to ensure seamless collaboration among multiple users editing the same document concurrently, without conflicts or data loss, develop a robust offline mode that allows users to access and edit documents without an internet connection, while ensuring synchronization when reconnecting, build a scalable solution that can handle a large number of users and documents without compromising performance or user experience and ensure secure authentication and authorization mechanisms to prevent unauthorized access to private documents.

## Keywords

Collaborative editing, collaborative text editing, conflict resolution algorithms, conflict-free replicated data type, crdt, fugue, multi-platform, document sharing, offline editing, operational transformation, ot, pwa, real-time collaboration, rich text editor, spa, version control, web api, web application, websockets.

# Project Plan

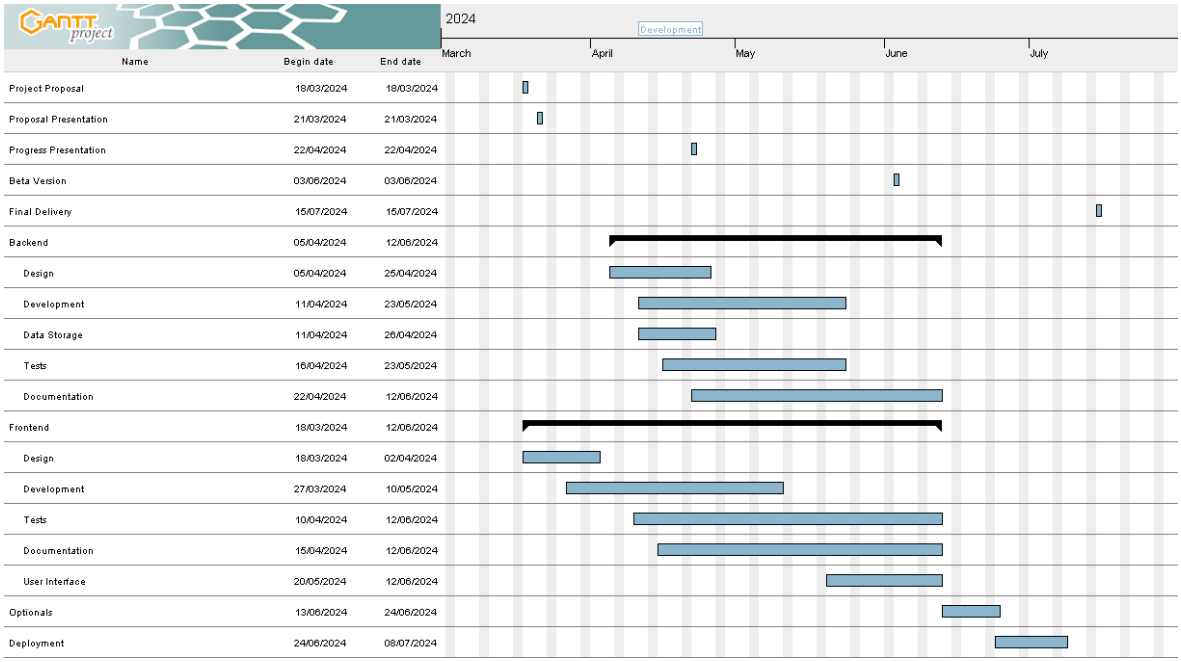


Figure 1: Gantt Chart Project Plan

# References


[1] Node.js. `https://nodejs.org`. Accessed on 17/3/2024.

[2] Typescript. `https://www.typescriptlang.org`. Accessed on 16/3/2024.

[3] React. `https://react.dev`. Accessed on 16/3/2024.

[4] Express. `https://expressjs.com`. Accessed: 16/3/2024.

[5] Socket.io. `https://socket.io`. Accessed: 26/2/2024.

[6] Firestore. `https://firebase.google.com/docs/firestore`. Accessed: 16/3/2024.

[7] Postgresql. `https://www.postgresql.org`. Accessed: 16/3/2024.

[8] Docker. `https://www.docker.com`. Accessed: 17/3/2024.

[9] Vite. `https://vitejs.dev/`. Accessed: 17/3/2024.

[10] Matthew Weidner. The enigma of collaborative editing. `https://medium.com/@mehulgala77/concurrent-collaborative-editing-d10192e55d2e`, 2022. Accessed: 26/02/2024.

[11] Matthew Weidner and Martin Kleppmann. The art of the fugue: Minimizing interleaving in collaborative text editing. `https://arxiv.org/pdf/2305.00583.pdf`, 2023. Accessed: 26/02/2024.

[12] Matthew Weidner. Fugue: A basic list crdt. `https://mattweidner.com/2022/10/21/basic-list-crdt.html`, 2022. Accessed: 26/02/2024.