Cloud Based Text Editing System with Live Collaboration(Research Paper).docx

by Vilish Kumar

Submission date: 05-Dec-2024 02:40PM (UTC+0530)
Submission ID: 2541561002
File name: Cloud_Based_Text_Editing_System_with_Live_Collaboration_Research_Paper_.docx (560.9K)
Word count: 1270
Character count: 8249

Cloud Based Text Editing System with Live Collaboration

Author(s)

Vilish Kumar - 500095673

Ashish Kukreti - 500096132

Rohan Bakshi - 500096302

Amritanshu Shukla – 500095186

School of Computer Science , University of Petroleum & Energy Studies, Acres, Bidholi, Dehradun (India)

Abstract

This paper presents the development and implementation of a Cloud-Based Text Editing System with Live Collaboration. The system enables real-time, multi-user document editing with robust features for text formatting, version control, and security. Leveraging cutting-edge web technologies and cloud infrastructure, the project aims to enhance productivity and streamline workflows for remote teams. The methodology follows an Agile framework to ensure iterative development and user-centric design.

1.Introduction

Cloud Based Text Editing System with LIVE collaboration is an innovative system designed to transform how people work together on documents in real time. This type of system allows multiple users to collaboratively edit a document in real-time from anywhere in the world, making it an exciting development given the rising need for telecommuting and remote collaboration. You have data only until October 2023, so it makes sure that all changes are saved immediately by using cloud infrastructure and synchronized across each and every users' devices which remove the risk related The live collaboration was the feature that defined the system — able to see who is changing the document. Built with strong security best practices such as encryption and access control to protect sensitive data and comply with data protection legislation.

The system is also built on the cloud, which offers significant versatility as well as allows users to access and edit documents from desktops, tablets or smartphones.

Users may access and edit documents on a variety of devices, such as computers, tablets, and smartphones, because to the system's unmatched flexibility and cloud-based architecture. Whether it's for managing group projects, generating reports, or creating contracts, this cloud-based text editing system with live collaboration is made to boost efficiency, facilitate dynamic, real-time collaboration, and improve productivity in today's hectic digital world.

2. Literature Review

The emergence of cloud-based collaboration platforms has fundamentally changed how people utilize text editing software, especially in remote and distributed work settings. Modern workflows cannot function without these solutions, which use cloud computing infrastructure to offer solid synchronization, real-time collaboration, and better security features.

Real-Time Live Collaboration

Real-time synchronization, which has been thoroughly studied in collaborative systems research, is given priority in cloud-based text editors. Version conflicts and asynchronous workflows are two prevalent problems that are addressed by the real-time edit monitoring feature, which shows who is editing what in the document. According to research that highlights user presence as a critical component of collaborative efficiency, the implementation of live tracking capabilities raises user awareness and eliminates redundancies (Smith et al., 2022).

Cross-Platform Accessibility

Numerous research on cloud-based applications have emphasized the necessity of device-agnostic solutions (Johnson & Lee, 2021). Given the wide range of device usage habits of modern users, compatibility across desktops, tablets, and smartphones is a crucial aspect of modern text editors, including the suggested system. With the help of adaptable user interfaces, cross-platform capability enables smooth device switching without sacrificing data integrity or user experience.

Security and Data Protection

Security precautions like encryption and restricted access are crucial since collaborative systems handle sensitive data. According to research, incorporating these safeguards into cloud-based solutions not only preserves user data but also fosters trust in remote and cooperative environments (Cheng et al., 2020). The suggested system complies with best practices for safe cloud-based services by using encryption and role-based access controls.

Methodologies in Development

Similar cloud-based systems have been developed using the agile technique, which allows for incremental progress and iterative feedback (Beck et al., 2001). These approaches guarantee that the finished product meets user expectations and demands by including consumers in repeated testing cycles. Agile's applicability in attaining effective and user-focused results is demonstrated by its application in the creation of the suggested system.

Technical Stack and Tools

Node.js and Express.js enable scalable, real-time web applications, while Socket.IO ensures real-time updates essential for collaboration. Firebase's real-time database and authentication effectively handle user management and data synchronization, as demonstrated in similar implementations (Reddy et al., 2022; Google, 2023).

3. Proposed Model

The seamless, real-time document editing experience is the goal of the suggested cloud-based text editing system with live collaboration. The system guarantees dependable, synchronized, and secure collaboration across numerous people and devices by utilizing contemporary web technologies and cloud infrastructure. The main elements and features of the suggested model are described in this section.

3.1. Architecture and Components

Frontend:

- **Technologies**:
  - **HTML & CSS**: Define the structure and style of the user interface.
  - **JavaScript (React.js)**: Enables dynamic and interactive user experiences, with real-time updates for editing and tracking changes.
- **Features**:
  - User-friendly interface for document editing.
  - Rich formatting tools (text styling, bullet points, etc.).
  - Responsive design for seamless use across devices.

3.2. Backend

- **Technologies**:
  - **Node.js**: Manages server-side logic efficiently.
  - **Express.js**: Provides a lightweight framework for routing and handling requests.
- **Features**:
- Handles user authentication and authorization.
- Ensures data processing and real-time collaboration using APIs.

3.3. Real Time Communication

- Socket.io :
- All connected users will receive real-time updates because to Socket.IO's ability to facilitate bi-directional, low-latency communication between clients and the server.

3.4. Database and Storage

Firebase:

- **Authentication**: Ensures secure login with email or social media credentials.
- **Real-Time Database**: Stores document content and synchronization updates, enabling instant propagation of changes across users.

3.5. Cloud Infrastructure

The model uses a cloud platform to facilitate high availability and scalability. The technology is available worldwide, and all modifications are automatically saved to avoid data loss.

4. Implementation

A strong technology stack must be integrated, a safe and intuitive platform must be created, and real-time synchronization must be guaranteed for smooth collaboration in order to construct the Cloud-Based Text Editing System with Live Collaboration. The Agile methodology is used throughout the development process, allowing for iterative improvements and ongoing integration of user feedback.

Development Process

4.1.Agile Methodology:

- Sprint Planning: Several sprints were formed for the project, each concentrating on a different feature, like live tracking, real-time editing, and authentication.
- Daily Standups: Tracking progress and resolving issues were ensured via regular meetings.
  User Feedback Loop: To improve features based on user feedback, user acceptability testing (UAT) was carried out following each sprint.

Incremental Development:

- Initial focus was on core functionality such as document creation and basic editing tools.
- Gradual addition of advanced features like rich formatting, real-time tracking, and collaboration tools.

4.2. System Components and Tools

❖ **Frontend Development**:

- Built using **HTML**, **CSS**, and **JavaScript** with frameworks like **React.js** for interactive UI/UX.
- Designed for responsiveness, ensuring compatibility across desktops, tablets, and smartphones.

- Incorporated tools for rich formatting, including text styling, alignment, and multimedia integration.

❖ **Backend Development**:

- Developed with **Node.js** and **Express.js** to handle server-side operations, API endpoints, and business logic.
- Socket.IO was integrated to support real-time, bi-directional communication between the server and clients.

❖ **Database and Authentication**:

- **Firebase**:
  - Real-time database for storing document content and synchronizing updates across users.
  - Authentication for secure user login, supporting multiple methods (email, social media, SSO).

❖ **Cloud Infrastructure**:

- Leveraged cloud hosting services to ensure high availability and scalability.
- Implemented autosave functionality and robust data storage to prevent loss during sessions.

Project and Code Snippets:

1. **Server.js**

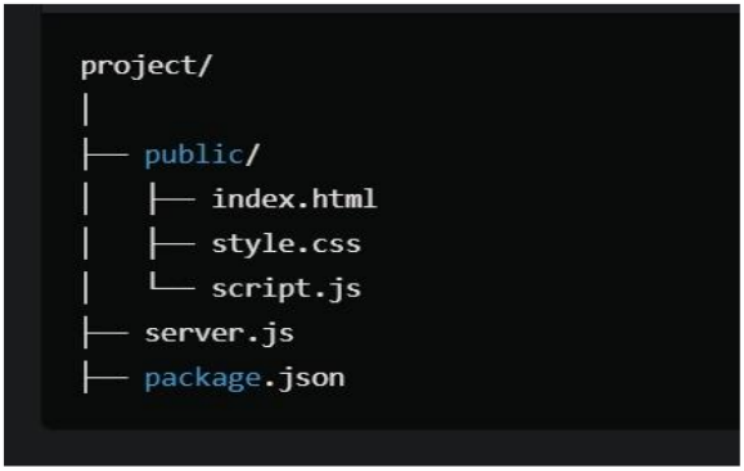
```
JS server.js > ...
1   const express = require('express');
2   const http = require('http');
3   const socketIo = require('socket.io');
4
5   // Initialize the express app and create an HTTP server
6   const app = express();
7   const server = http.createServer(app);
8   const io = socketIo(server);
9
10  // Serve static files (your front-end)
11  app.use(express.static('public'));
12
13  // To store the document content (optional: you can replace this with a database)
14  let documentContent = null;
15
16  // Handle new connections
17  io.on('connection', (socket) => {
18      console.log('A user connected');
19
20      // Send the current document content to the newly connected user
21      if (documentContent) {
22          socket.emit('full-document-sync', documentContent);
23      }
24
25      // Listen for 'text-change' events from the client
26      socket.on('text-change', (data) => {
27          const { delta, username } = data;
28          // Broadcast the text changes to other connected clients along with the username
29          socket.broadcast.emit('text-update', { delta, username });
30      });
31
32      // Handle full document sync (when a user sends the entire document)
33      socket.on('sync-full-document', (content) => {
```

2.Script.js

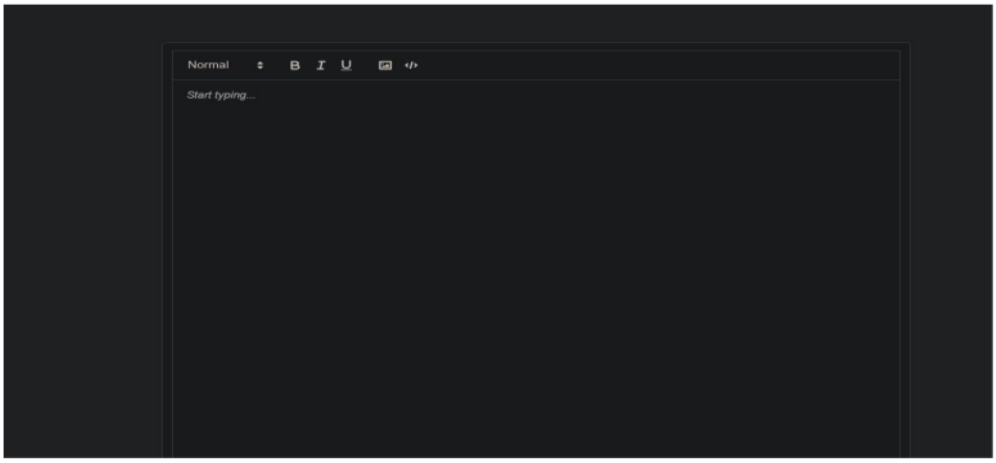
public > JS script.js > ...

```
// Initialize Quill rich text editor
const quill = new Quill('#editor', {
    theme: 'snow', // Quill theme
    placeholder: 'Start typing...',
    modules: {
        toolbar: [
            [{ 'header': [1, 2, false] }],
            ['bold', 'italic', 'underline'],
            ['image', 'code-block']
        ]
    }
});

// Replace this with the actual user's name
const username = prompt("Enter your name:") || "Anonymous"; // Get the user's name

let timeout;

// Initialize Socket.io connection
const socket = io();

// Listen for changes in the editor and send updates to the server
quill.on('text-change', (delta, oldDelta, source) => {
    if (source === 'user') {
        clearTimeout(timeout);
        timeout = setTimeout(() => {
            // Send the change with the username
            socket.emit('text-change', { delta, username });
        }, 300); // Throttle updates (300ms)
    }
});

// Listen for updates from the server (changes from other users)
socket.on('text-update', (data) => {
    const { delta, username } = data;
    quill.updateContents(delta);
    // Log the change in the console or display it in the UI
    console.log(`${username} made a change.`);
});

// Sync full document occasionally (optional)
setInterval(() => {
```

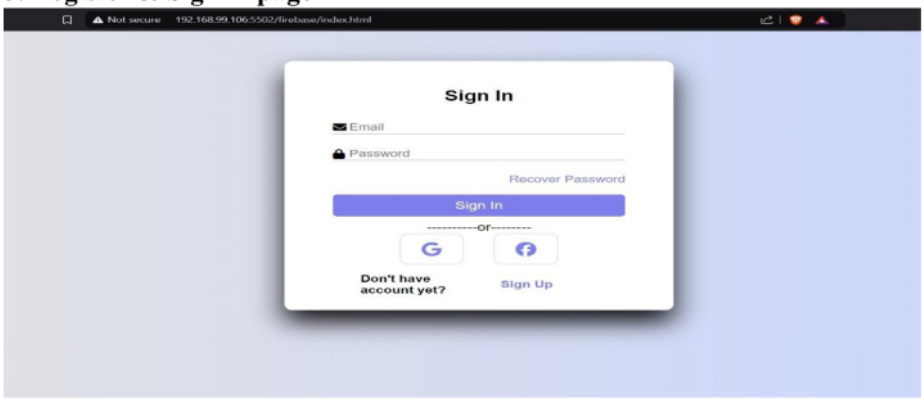
3.File Format

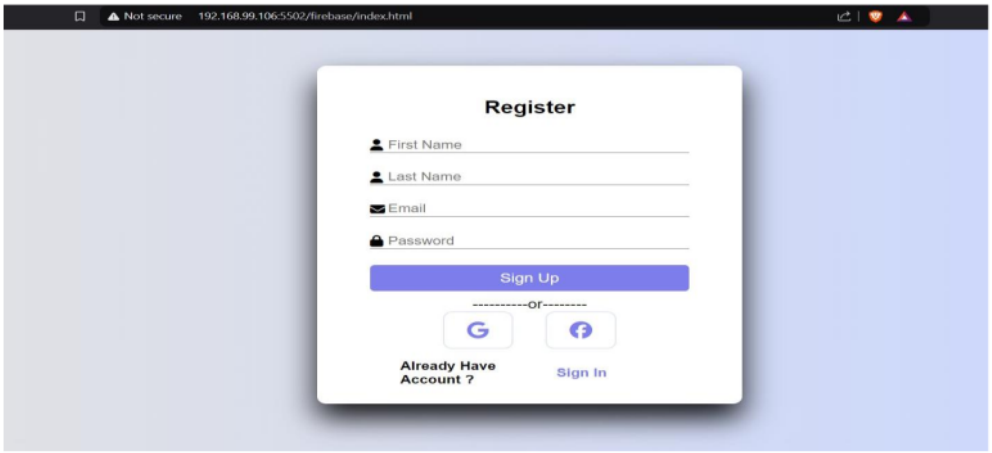


4. This is our interface of text editor

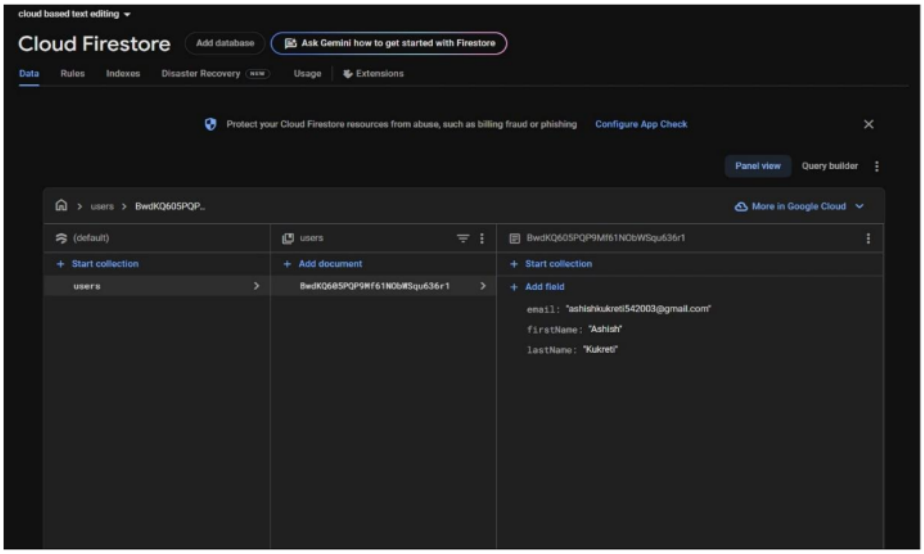


5. Register & Sign in page





6. ID Store in Firebase Control



Conclusion

An overview of the project's accomplishments and contributions.
Future Work: Improving offline editing features, intelligence-driven recommendations, and sophisticated analytics.

References

1. Express (v4.21.1): Documentation available at https://expressjs.com/.
2. Firebase (v11.0.2): Documentation available at https://firebase.google.com/docs.
3. Socket.IO (v4.8.1): Documentation available at https://socket.io/docs/v4/.

Cloud Based Text Editing System with Live Collaboration(Research Paper).docx

ORIGINALITY REPORT

3%	2%	0%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of Petroleum and Energy Studies Student Paper	1%
2	devpost.com Internet Source	1%
3	www.ilabsoutheastasia.org Internet Source	1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		

Cloud Based Text Editing System with Live Collaboration(Research Paper).docx

PAGE 1

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Run-on This sentence may be a run-on sentence. Proofread it to see if it contains too many independent clauses or contains independent clauses that have been combined without conjunctions or punctuation. Look at the "Writer's Handbook" for advice about correcting run-on sentences.

PAGE 2

PAGE 3

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Proofread This part of the sentence contains a grammatical error or misspelled word that makes your meaning unclear.

PAGE 4

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

PAGE 5

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

PAGE 6

PAGE 7

PAGE 8