Major Project 1 (End Term Report).docx

by Vilish Kumar

Submission date: 05-Dec-2024 02:45PM (UTC+0530)
Submission ID: 2541562821
File name: Major_Project_1_End_Term_Report_.docx (595.28K)
Word count: 853
Character count: 5419

MAJOR PROJECT-1

End Term Report

ON

Cloud Based Text Editing System with Live Collaboration

Submitted By

Ashish Kukreti	Rohan Bakshi	Vilish Kumar	Amritanshu Shukla
R2142211009	R2142211066	R2142211034	R2142210099

Under the guidance of

Mr. Alok Jhaldiyal

Assistant Professor- SG

Department of Virtualization

Cloud Computing & Virtualization Technology

Department of CSO

School of Computer Science

UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

Dehradun-248007

Aug-Dec, 2024

Index

1

Contents

Chapter 1: Introduction

A Cloud-Based Text Editing System with Live Tracking is a cutting-edge solution designed to revolutionize how individuals and teams collaborate on documents in real time. With the increasing need for remote work and digital collaboration, this system provides a seamless and efficient platform for multiple users to co-edit documents from anywhere in the world. By leveraging cloud infrastructure, the system ensures that all changes are instantly saved and synchronized across all users' devices, eliminating the risks associated with version conflicts and data loss.

The live tracking feature is a core component of the system, allowing users to monitor who is editing the document. The system also integrates robust security measures, including encryption and access controls, to protect sensitive information and ensure compliance with data protection regulations.

Moreover, the cloud-based nature of the system provides unparalleled flexibility, enabling users to access and edit documents on various devices, including desktops, tablets, and smartphones. Whether it's for writing reports, drafting contracts, or managing collaborative projects, this Cloud-Based Text Editing System with Live Tracking is designed to enhance productivity, streamline workflows, and support dynamic, real-time collaboration in today's fast-paced digital environment.

Chapter 2 : Objectives

Text Editing

- **Rich Formatting Options**:
  - Users can enhance the appearance of text with **bold**, *italics*, and a variety of **fonts**.
  - Other features include adjusting **font size**, **color**, **alignment**, and applying **styles** (headings, subheadings, etc.).
- **Standard Functions**:
  - **Cut, copy, paste, and undo** functionalities are seamlessly integrated, allowing users to easily manage their text.
  - **Find and replace** text options enable quick content modifications.
  - Support for **bullet points**, **numbered lists**, and **block quotes** for organizing information.

Collaboration Tools

- **User Presence**:
  - Indicators show when other users are online and actively working on the document, improving communication.
- **Commenting and Suggestions**:
  - A built-in **commenting feature** lets users leave feedback or suggestions without altering the actual text.
  - Comments can be resolved or marked as complete, streamlining collaborative discussions.

Cross-Platform Compatibility

- **Device Accessibility**:
  - The platform is designed to work smoothly on **desktops, tablets, and smartphones**, ensuring that 5ers can contribute from anywhere.
  - The interface adapts to different screen sizes, providing an optimal user experience across devices.
- **Cloud Sync**:
  - Real-time synchronization ensures that documents remain **updated across platforms**, eliminating the need for manual uploads or transfers.

Chapter 3: Technology Stack

Frontend Technologies:

- **HTML:** Provides the structure of the web application.
- **CSS:** Styles the application, ensuring a responsive and modern design.
- **JavaScript:** Enables interactive features and real-time updates (using frameworks like React or Vue.js).

Backend Technologies:

- **Node.js:** Handles server-side logic and manages client requests efficiently.
- **Express.js:** Framework for building the web server, providing a robust set of features.

Chapter 4: System Workflow

User Authentication

- **Sign-Up and Log-In**:
  - Users can create an account with **email, social media, or single sign-on (SSO)** options, ensuring convenience and security.

Document Creation/Selection

- **New Document Creation**:
  - Users can easily start a new document using a **simple interface**, which may include templates for specific document types (e.g., reports, essays, proposals).
  - Metadata (title, description, tags) can be added during creation for easy retrieval later.

Document Editing

- **Real-Time Editing**:
  - Users can instantly begin editing after document creation or selection, with their changes visible to all collaborators in real-time.
  - Rich editing options such as **text formatting (bold, italics, underline, font color, size)**, **paragraph alignment**, and **inserting multimedia elements (images, links, tables, etc.)** enhance document structure.

Autosave and Undo/Redo:

- The system continuously **autosaves** changes, reducing the risk of losing data.

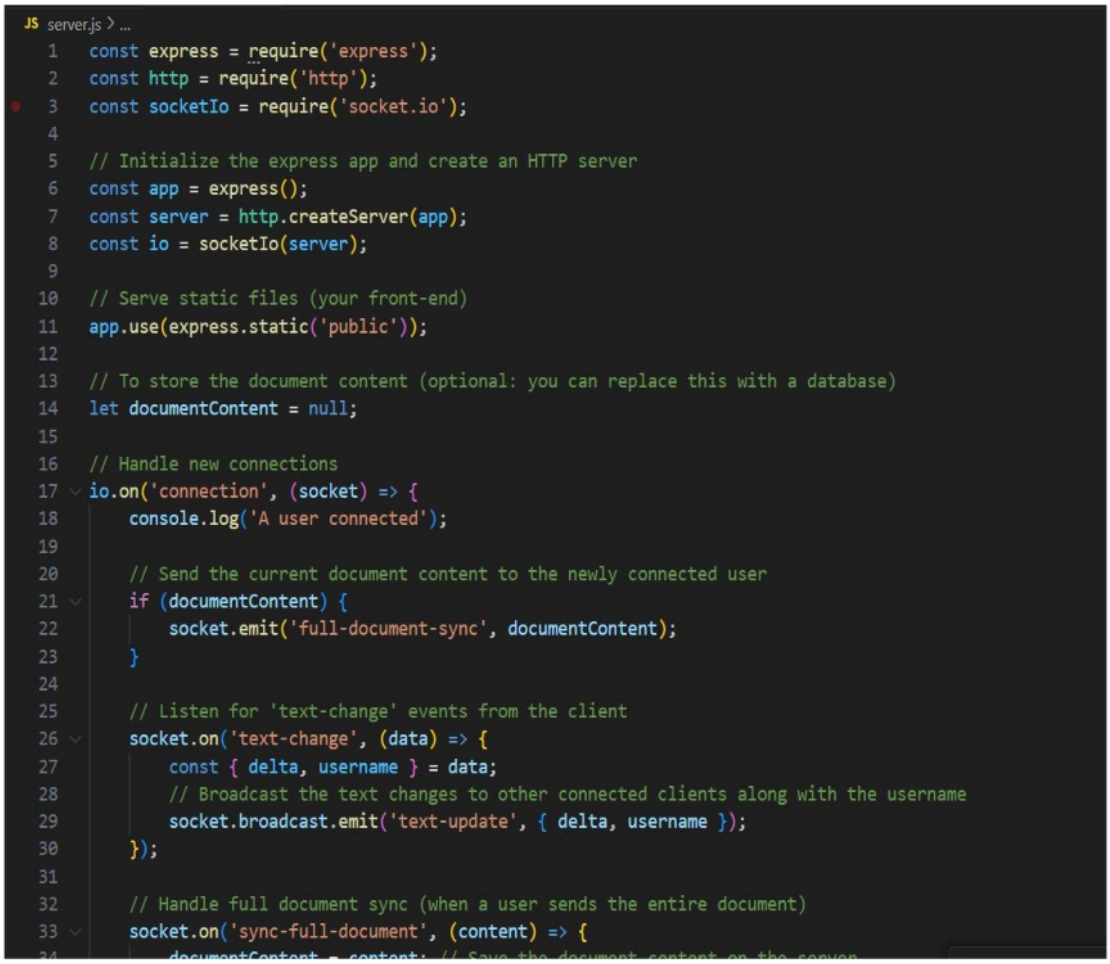
Undo/redo functions ensure that mistakes can be quickly corrected, even during collaborative session

Chapter 5: Methodology

1. Agile Methodology

- **Sprint Planning:** Break down the project into smaller tasks and plan sprints (typically 2-4 weeks) where each sprint focuses on a specific set of features (e.g., real-time collaboration, user authentication).
- **Daily Standups:** Conduct daily standup meetings to discuss progress, challenges, and next steps, ensuring the team stays aligned and any issues are addressed promptly.
- **Incremental Development:** Develop the system incrementally, with each sprint delivering a potentially shippable product increment that adds value.
- **User Feedback Loop:** Involve end-users early in the process by conducting user acceptance testing (UAT) after each sprint, gathering feedback to refine and improve the system iteratively.

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) => {
```

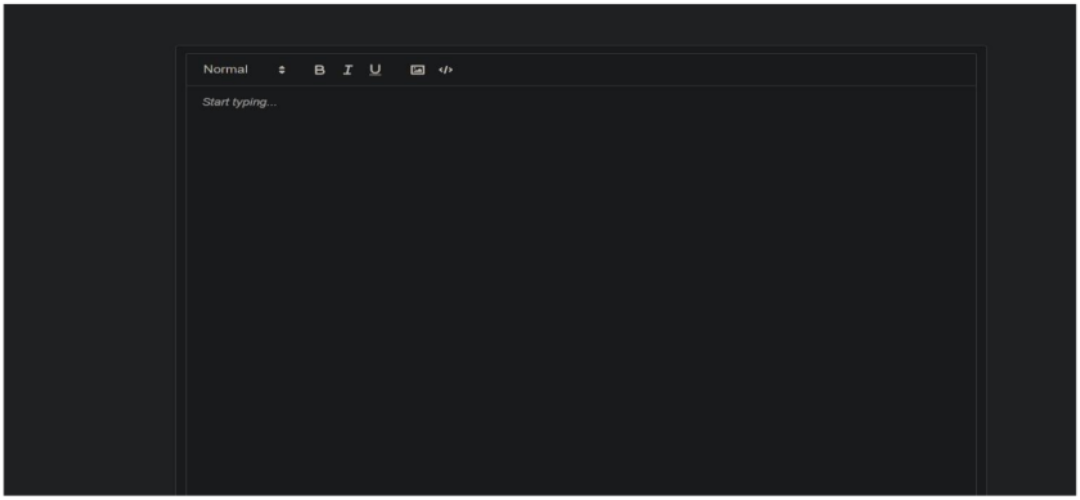
Script.js

```
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(() => {
```

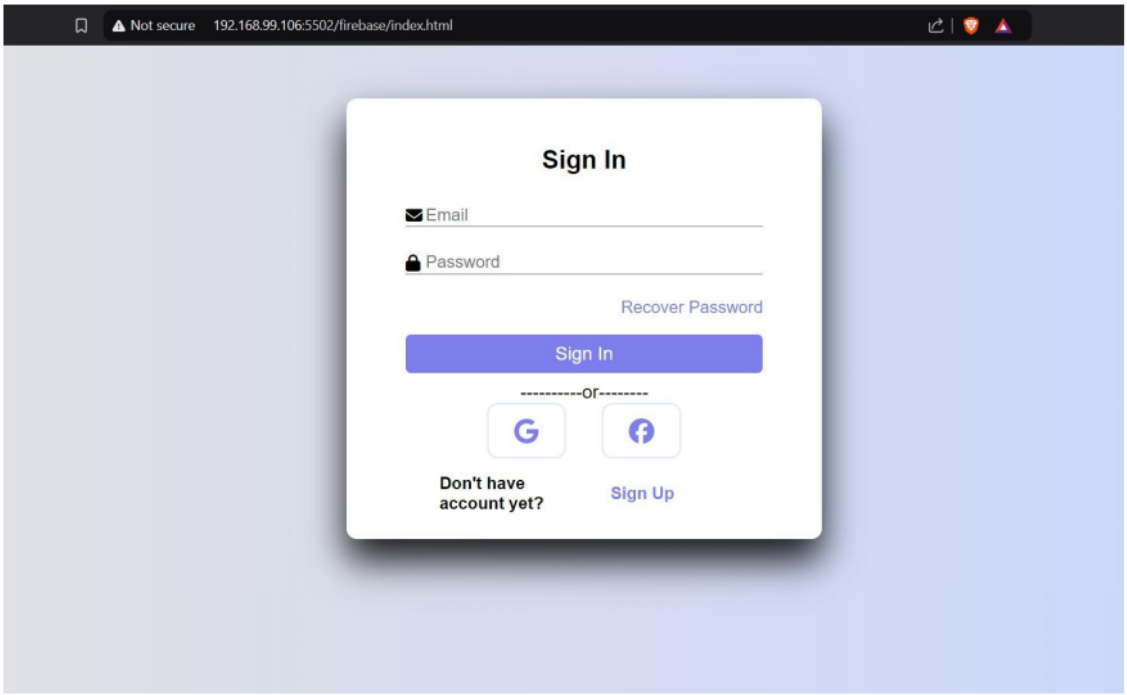
This is our file format:

```
project/
│
├── public/
│   ├── index.html
│   ├── style.css
│   └── script.js
├── server.js
├── package.json
```

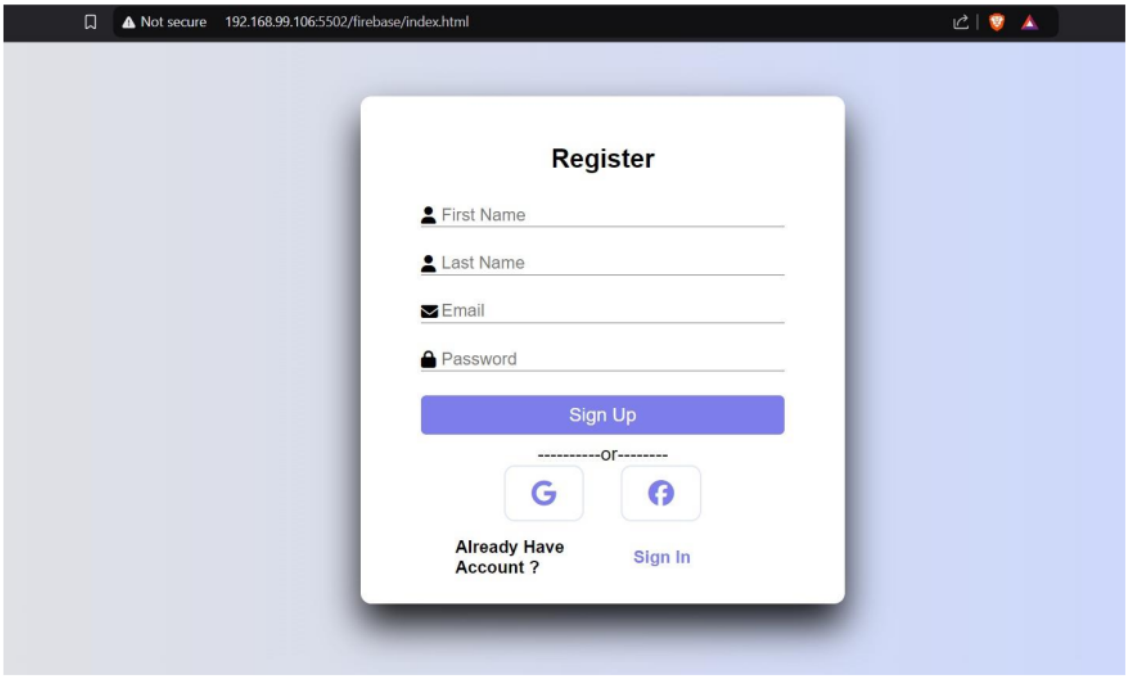
This is our website / software looks like:



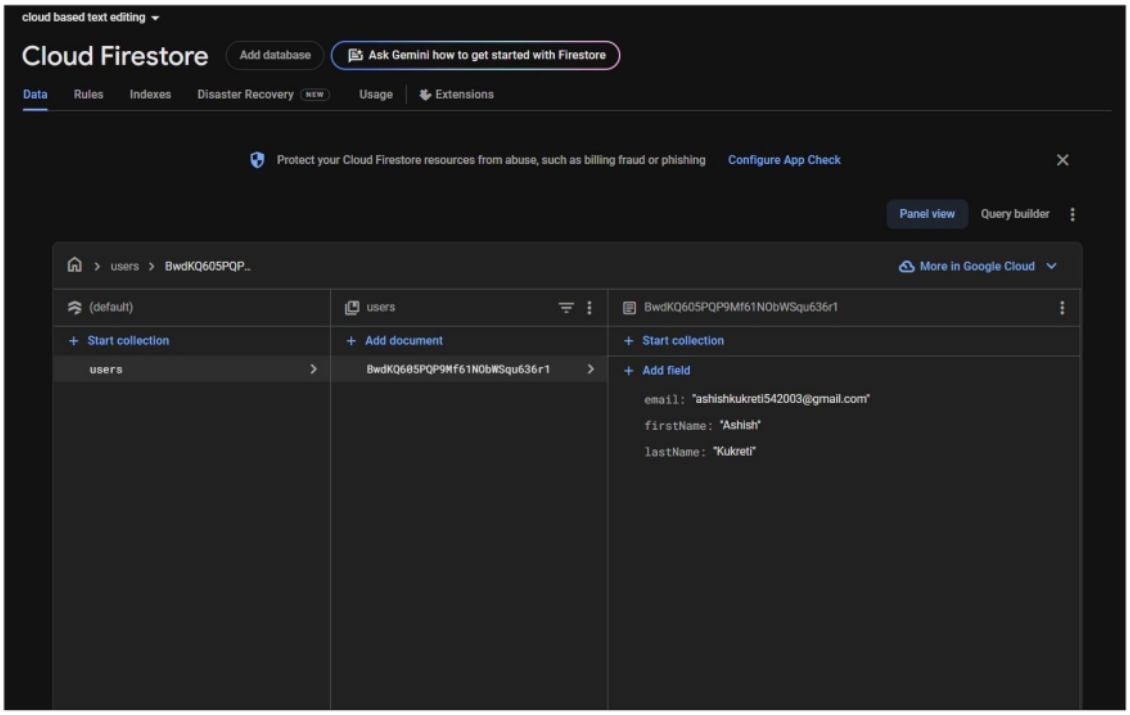
Sign-in page



Register page



Id store in firebase Console



References:

1. **Express (v4.21.1)**:
    - A fast, minimalist web framework for Node.js.
    - **Documentation**: https://expressjs.com/
2. **Firebase (v11.0.2)**:
    - Google's platform for building apps with features like authentication, database, and real-time storage.
    - **Documentation**: https://firebase.google.com/docs
3. **Socket.IO (v4.8.1)**:
    - A library for real-time, bi-directional communication between web clients and servers.
    - **Documentation**: https://socket.io/docs/v4/

Major Project 1 (End Term Report).docx

ORIGINALITY REPORT

10%	8%	0%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.coursehero.com Internet Source	2%
2	Submitted to Arab Open University Student Paper	2%
3	hackerella.com Internet Source	2%
4	ideas.repec.org Internet Source	2%
5	fastercapital.com Internet Source	1%
6	www.interfaceplanet.com Internet Source	1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		

Major Project 1 (End Term Report).docx