[CSE] Distributed Computing Systems

Final Project

Online Text Editor (QM Text Editor)

Team 4:

Mahmoud Maged Mahmoud Mohamed Hafez	18P2847
Ahmed Khaled Saad Ali	1809799
Moataz Ahmed Mohamed Refaat	1601442
Mohamed Khaled Talaat	1802559

1 Introduction

A shared document editor program is very used nowadays by various job fields, its even recommended to have expertise in using one like Google Docs; that's why programmers and networking engineers must put their hands together and implement such a system. But we will stumble in many communication, networking, security, fault tolerance and fast response problems, which we will solve in this project naming it "QM Editor".

2 Table of Contents

3 Project Description

At first, we specified the projects requirements which are the following:

- System should have the ability of supporting multiple, autonomous agents/computers (human or automated) and managing shared resources between the agents.
- Any agent/computer should just have a link to access the application.
- System's state to be distributed upon several client or server nodes.
- System should be strong enough to be reliable for not crashing.
- System resumes its operations in its last reached state if one node crashes.
- System recovers the state of the crashing node in order for it to continue its operation.
- System managing real-time editing and viewing by several participants.
- An agent can see how many agents are editing the document he/she is editing.
- Maintaining replicas for fault tolerance.
- Caching/Copy migration to minimize application response time.

These requirements will be discussed on how they are reached in "Detailed Analysis" section of the report and proving that they are met successfully in "Testing scenarios and results" section of the report.

For designing and implementing our QM Editor, it is a Web Application with a Three-Tier System Architecture, Front-End, Back-End and Database.
The Front-End is coded in React-Js and is running and hosted on a separate server from the Back-End. The Back-End is coded is JavaScript (Node-Js server) and is hosted on a separate server from the Front-End. Front-End and Back-End are connected with sockets written using JavaScript library called "socket.io", this library manages the real-time editing feature.
MongoDB is used as a database for storing changes in form of JSON. Database is hosted on a separate server on MongoDB Atlas cloud. We use a library called mongoose library that has RPCs (remote procedure calls) to communicate with the mongoDB server.

QM Editor has operations like, adding a new document, deleting, and modifying an existing document.

4 Beneficiaries Of System

Our target groups who will benefit from such a system, one of them is a newspaper writers' team who could share in writing multiple newspaper reports.
Another targeted group is in a software development company, when a client and a software engineer want to write the software requirements of the client's dream software application dividing the requirements on multiple documents each document is being edited by both of them so that the client types what he desires, and the software engineer rephrases that to more understandable and applicable requirements.

5 Detailed Analysis:

To achieve that multiple computers/users all access at the same time the Web Application we need to deploy/host the Web Application on a PAAS Cloud, so we hosted our QM Editor on Heroku which is a platform as a service (PAAS) Cloud and it provides an OS and an environment for running the application where in our case, OS is Linux and environment installed is Node-JS.

6 Tasks Breakdown

After we studied the requirements of the system and connecting them to our Beneficiaries, we intended to work on the project in a Water-Fall Model and break the requirements to the following tasks:

- Project Code

1. What platform will be the application?
   Web Application
2. System Architecture
   3-Tier (Front-End, Back-End and Database)
3. Website Styling.
4. Adding, editing and deleting documents.
5. Making client and server nodes/sockets.
6. Supporting multiple, autonomous agents/computers (human or automated) and managing shared resources between the agents.
7. System's state to be distributed upon several client or server nodes.
8. System resumes its operations in its last reached state if one node crashes.
9. System recovers the state of the crashing node in order for it to continue its operation.
10. System managing real-time editing and viewing by several users/computers.
11. An agent can see how many agents are editing the document he/she is editing.
12. Maintaining replicas for fault tolerance.
13. Caching/Copy migration to minimize application response time.
14. System should be strong enough to be reliable for not crashing.
15. Deploying the application on PAAS Cloud so that multiple computers/users can access same the website at the same time editing documents together using **ONLY ONE LINK**.

16. Backup server
17. Testing Manually each feature implemented in the application.
18. Documenting the project code to an organized report showing:

1. Introduction
2. Detailed project description
3. Beneficiaries of the project
4. Detailed analysis explaining Code structure and logic
5. Task breakdown structure
6. System architecture and design
7. Testing scenarios and results
8. End-user guide
9. Conclusion

- YouTube Video illustrating Test case scenarios for the application.
- ReadMe file
- Uploading project code on GitHub

7 Team Members Roles

Mahmoud Maged	-Most of Project Code (Front End) -YouTube Video
Moataz Ahmed	-Contributed in Project Code (Second server) -Report Formatting and gathering -Documentation point No. 4
Ahmed Khaled	-Documentation points No. 2,3,4,5,7 -ReadMe file -Contributed in Project Code (Data Base)
Mohammed Khaled	-Documentation points No. 1,8,9 -Contributed in Project Code (Back End)

8 System architecture and design

The system used the MVC layered Architecture where the Model is the Data base containing the files we are using, the View is the front-end development, and the controller is the back-end logic.

The system is designed to have two servers and multiple clients.
The client communicates with the available two servers and deals with the first response of them then the client continues with him by this we are working on (Active-Active) Servers

If one of the servers are down to any reason:

- Hardware problems
- Electricity cut off
- Malicious Attack

- Huge Traffic

The Another servers continue with the client

9 Testing Scenarios and results

- Creating new document
  In Home window, click “new file” component.

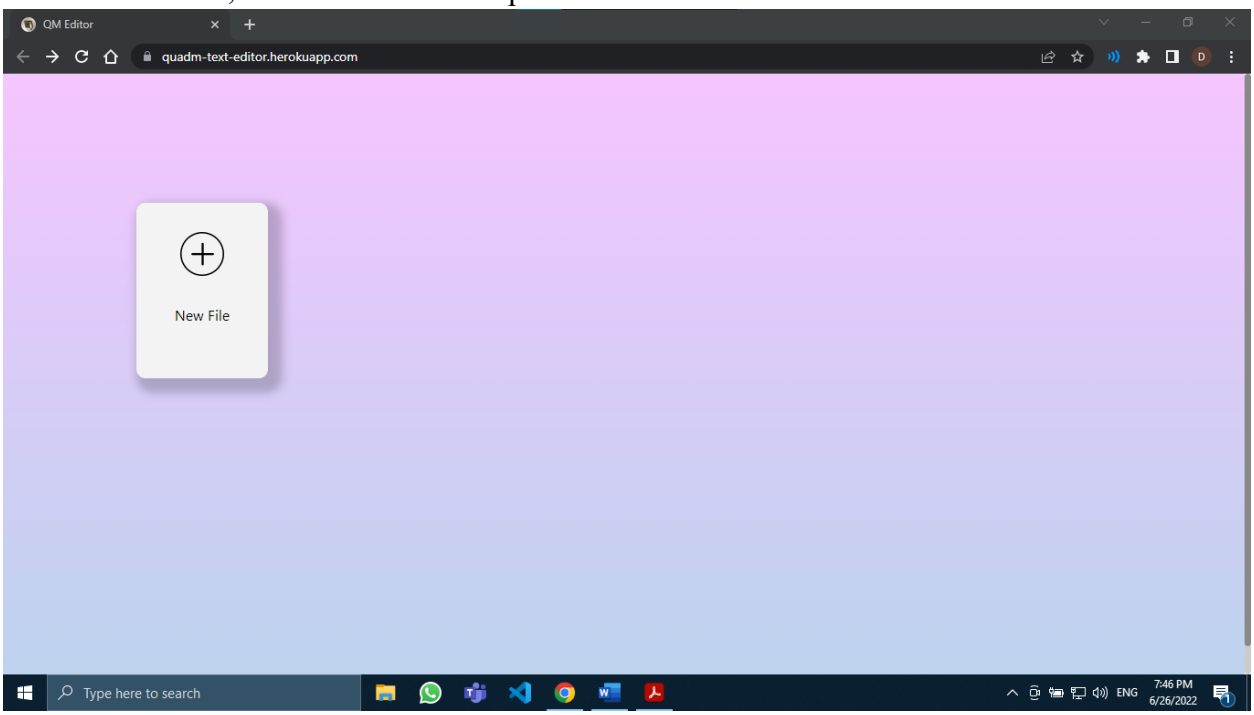

Fig.1

Now you are in the window of document created.

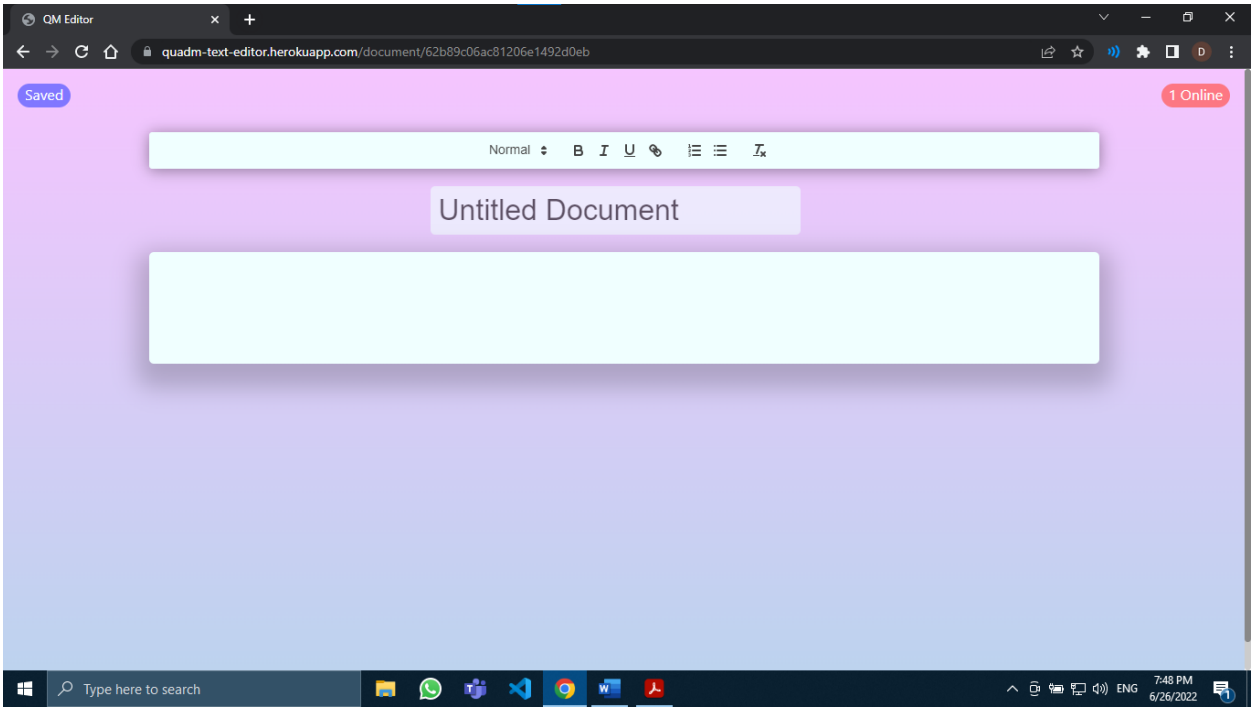

Fig.2

You can edit the document's title, text content, and the style of your text. Also, document is auto-saving as you type in the document and in top left you can see that you are the only one online opening the document.

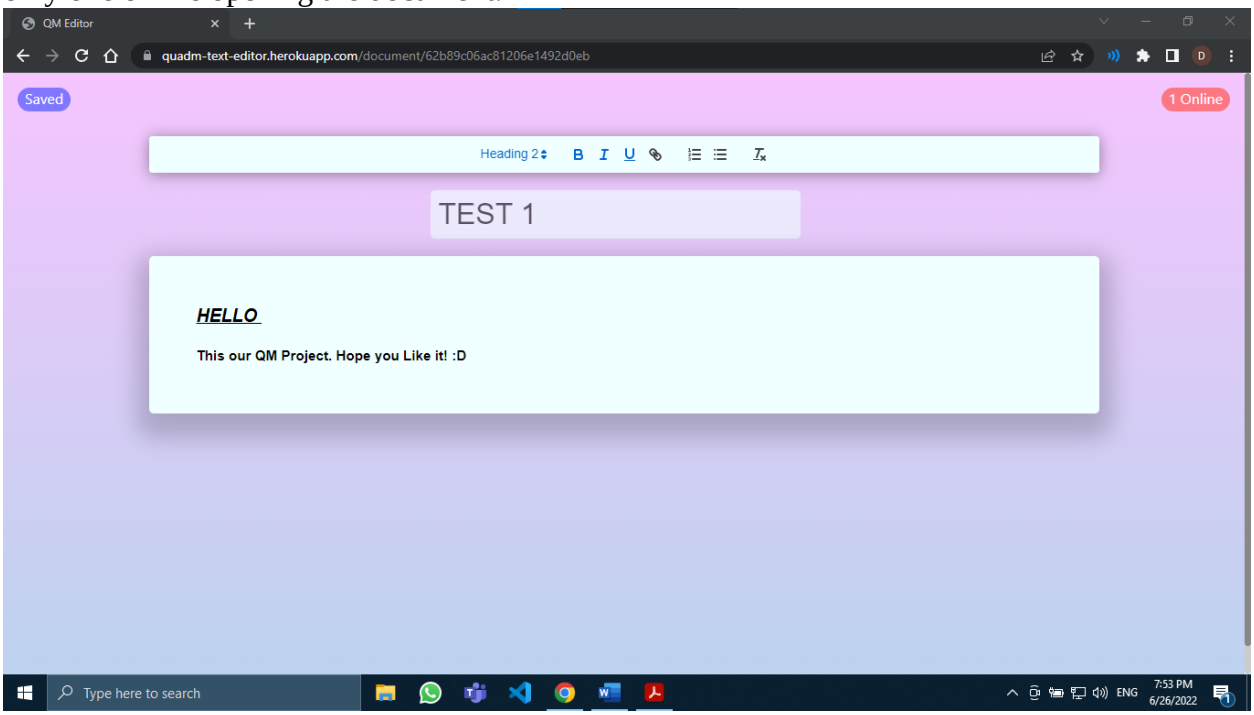

Fig.3

Click back from browser to get back to home page. No you can see the created document, you can click on it to edit it.

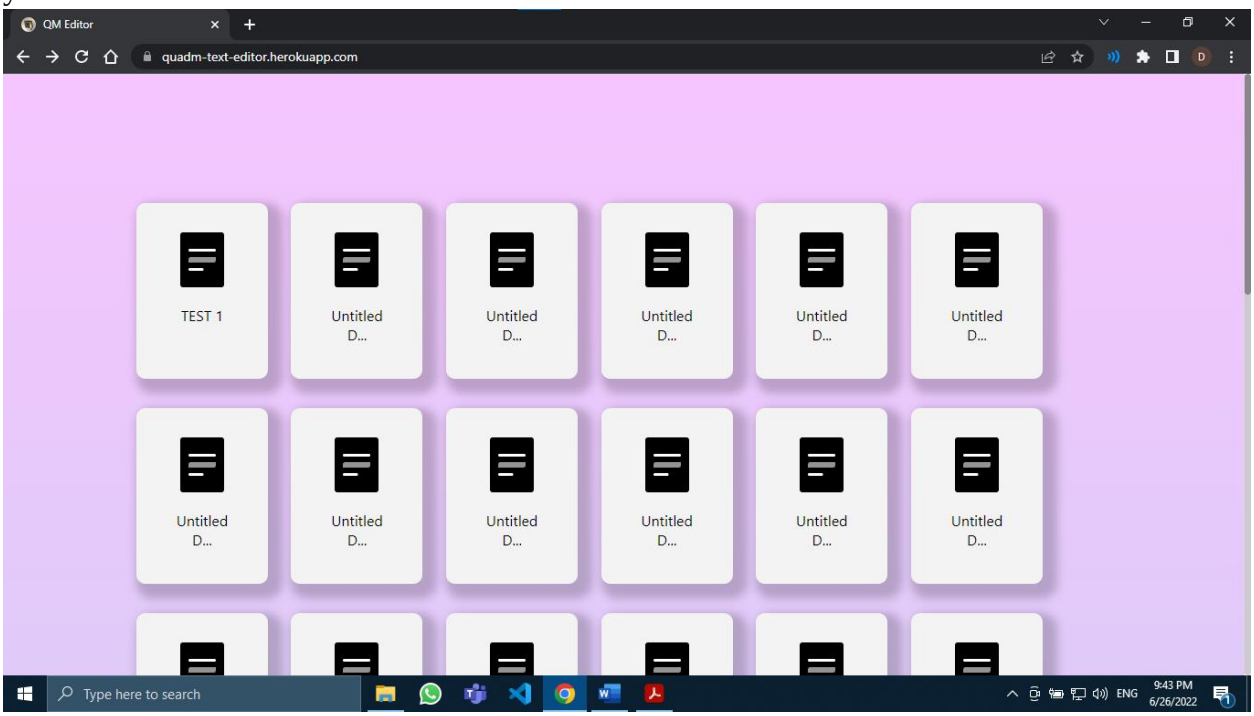

Fig.4

- Deleting a document

Right Click a document, to show delete option.

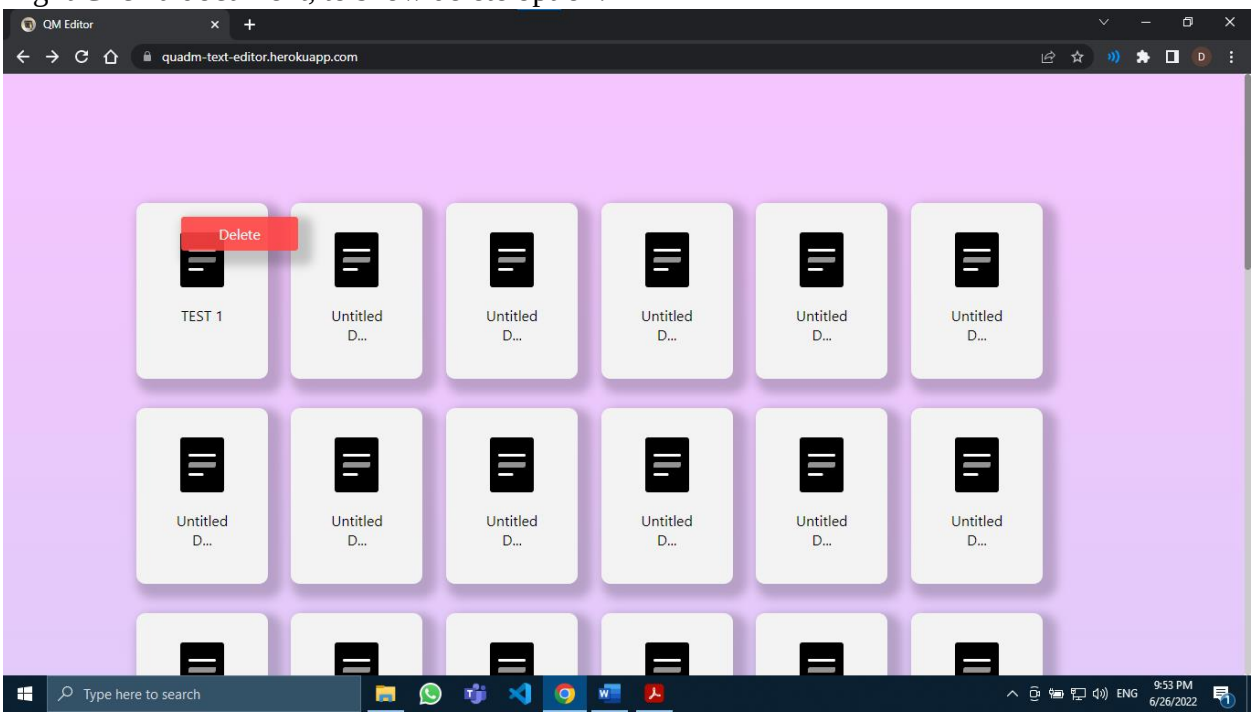

Fig.5

Click on delete and an alert will pop up top middle

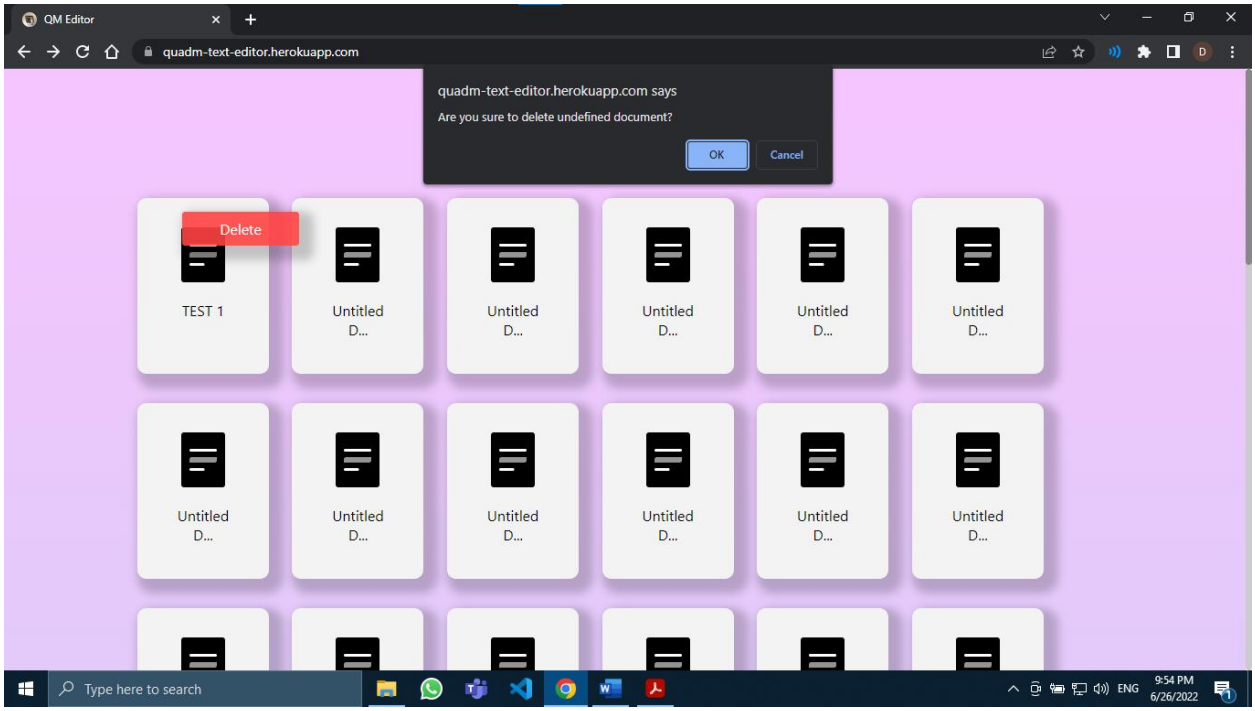

Fig.6

Click “OK” to confirm deletion or “Cancel” to cancel the deletion. We pressed “OK” so “TEST1” document is actually deleted as shown below.

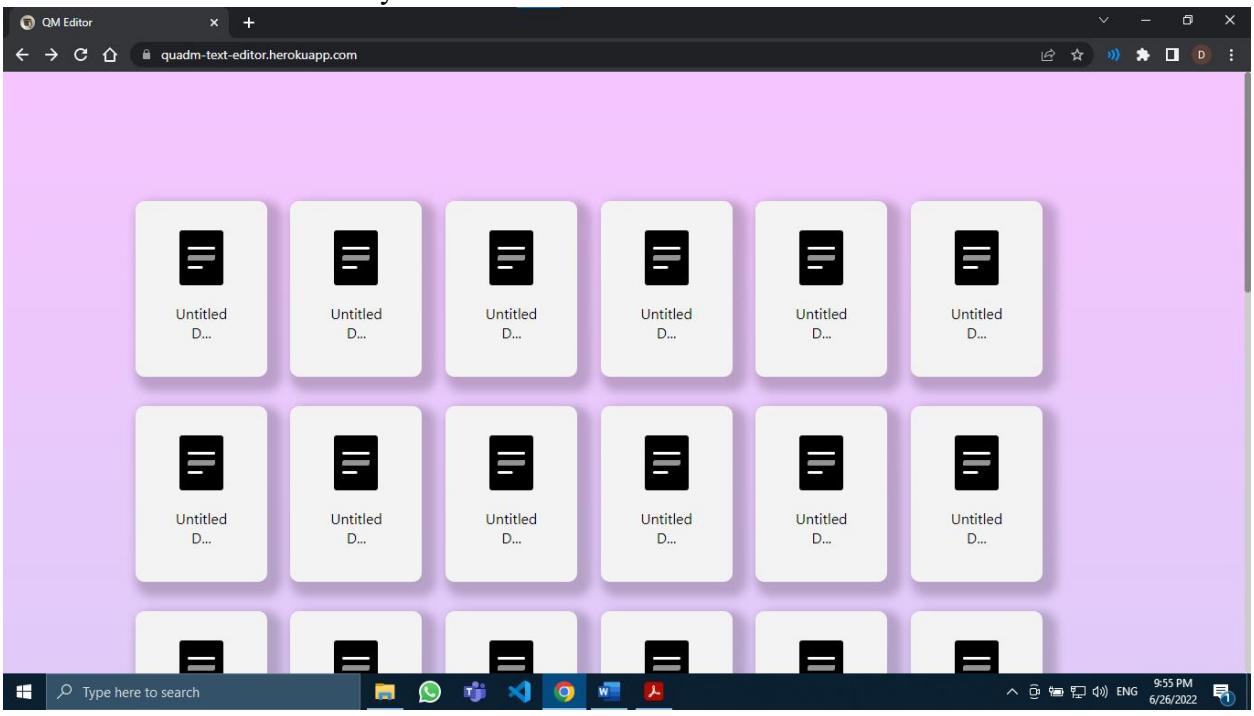

Fig.7

- Multiple tabs editing same document

Document named “Client1” is observed by both tabs

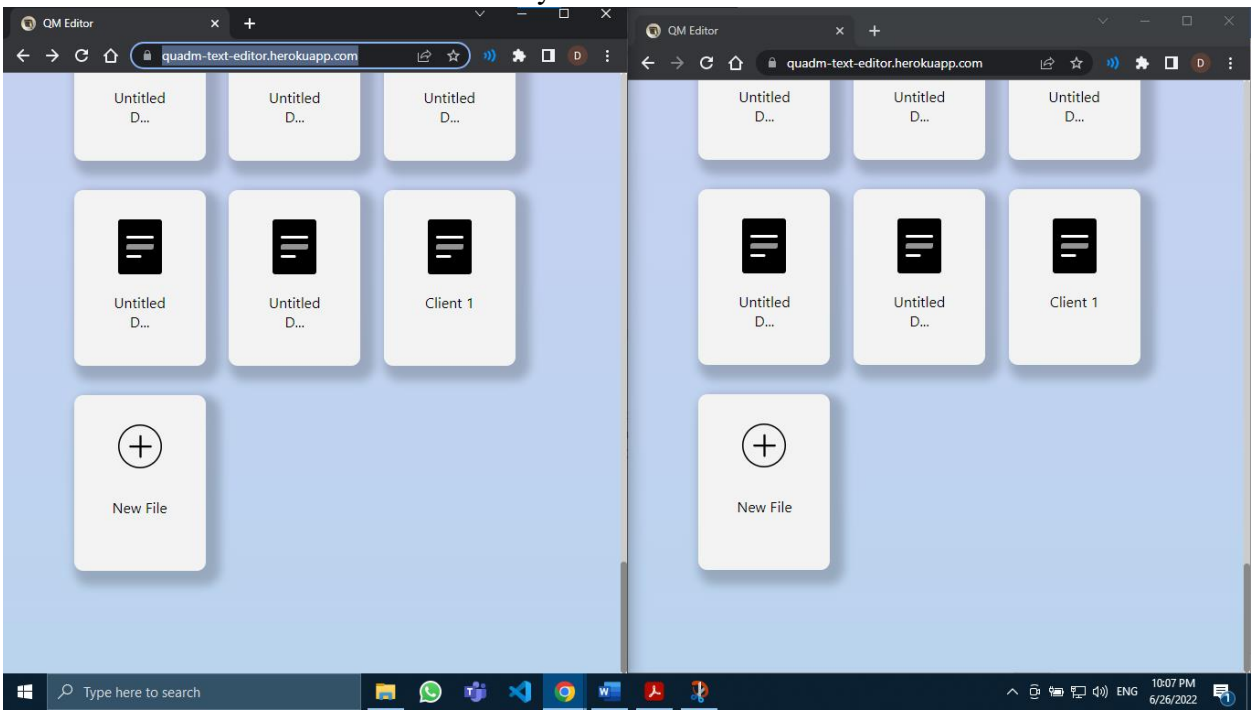

Fig.8

Now both tabs are opening “Client1” document. Very good feature top right that there are 2 online users opening this document which proves our case.

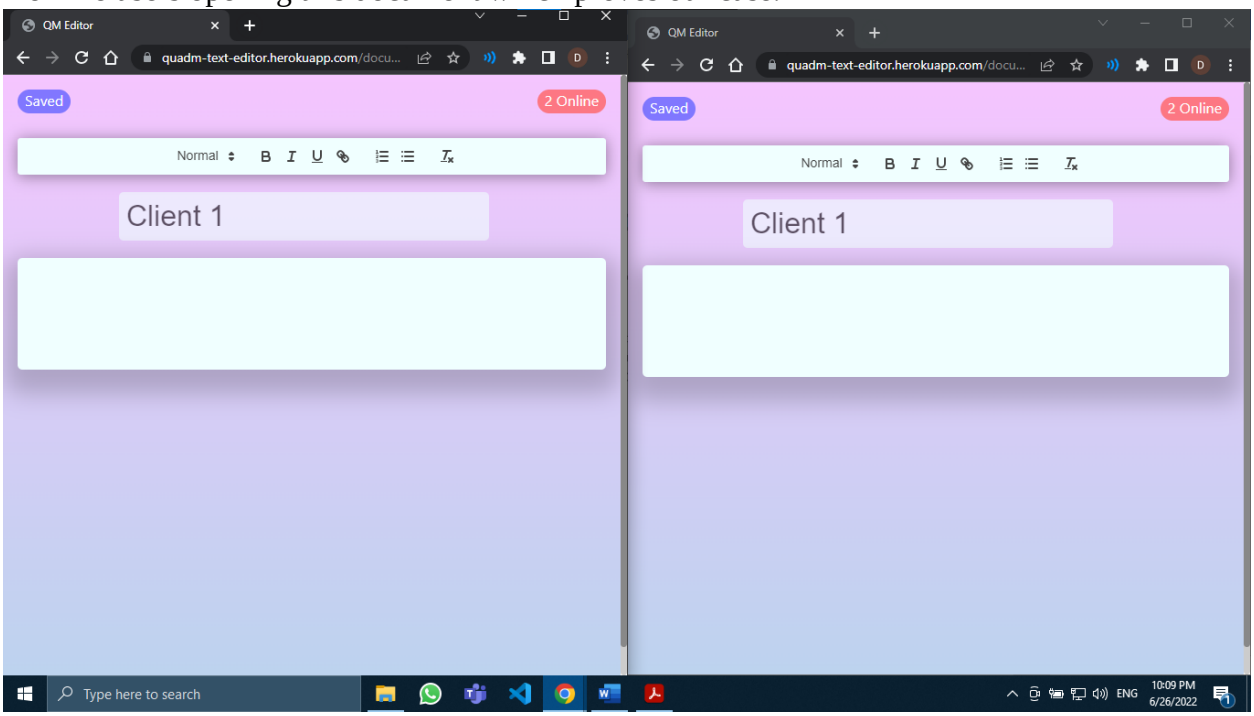

Fig.9

This GIF shows that if one tab edits the document the other tab is responding to its changes, and vice versa.

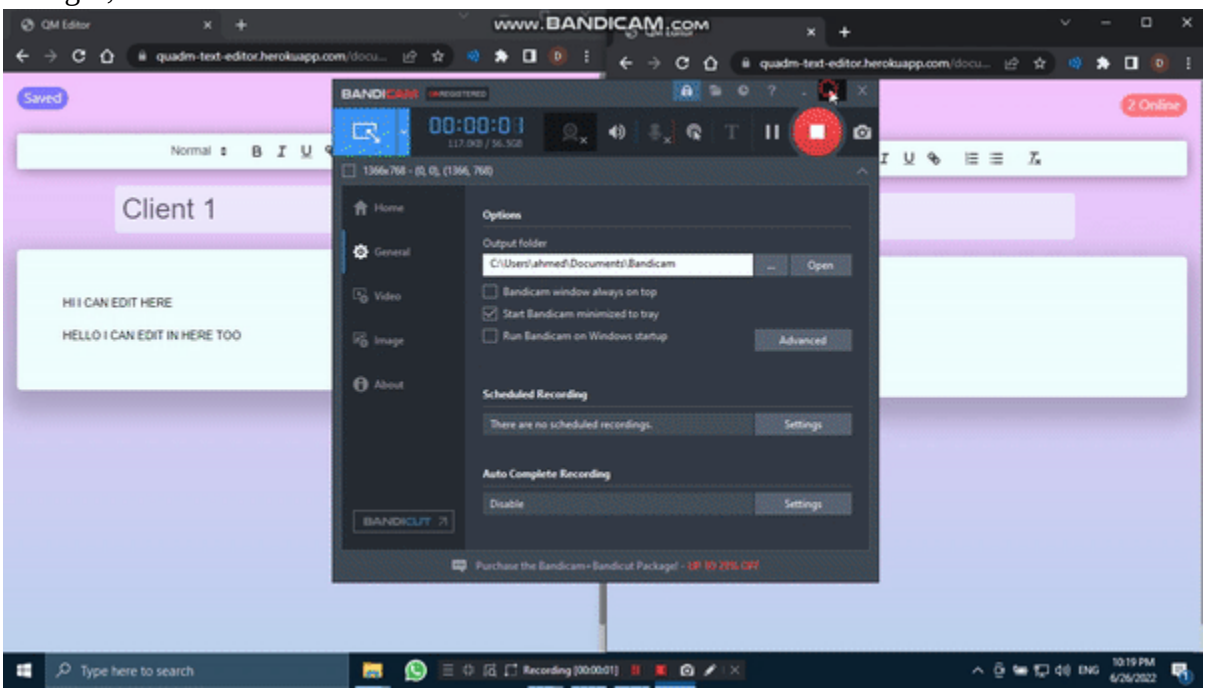

Fig.10

- Multiple users/computers editing same document
  This test case is best illustrated in the Video Recorded for the Project Showcase.

10 End-User Guide

- Click the following link: https://quadm-text-editor.herokuapp.com/
- You will see all documents saved and a “new file” component, press it to make a new document.
- Now you are in the new document window you created.
- Top middle you have several text decoration options such as font size, bold, underline, numbered/un-numbered lists and other options.
- Under this you can edit the document’s title.
- Under this you can see the text area you can type in which will be the document’s text data.
- Top left you will see when the document autosaves as you write in the document.
- Top right you can see how many users are opening the same document you are opening.
- Press back button top right to go back to home window showing that the document has been created successfully and saved.

- You can delete the document by right clicking on the document and a delete button will appear click it, then an alert message at the window's top will appear to confirm the deletion process.

Click "OK" to confirm deletion or "Cancel" to cancel the deletion

11 Conclusion

We observe that maintaining a sustainable, reliable, organized-communication, fault tolerant, real-time editing, crash-ready, multiple user supporter, good response-time Distributed System requires Networks, Distributed Computing and Programming skills.

12 Drive Link To The Code + Video

Press Here

13 GitHub Link

Press Here