Mini Project Report On

Collaborative Editor using Socket.io

for the course

IT254: Web Technology and Applications

Submitted by

Srushti HP (181IT118)
Kotla Karthik Reddy (181IT123)
Shonali K.S (181IT244)

IV SEM B.Tech (IT)

Under the guidance of

Mrs. Priyadarshini
Dept of IT, NITK Surathkal

in partial fulfillment for the award of the degree

of

Bachelor of Technology

in

Information Technology

at

Department of Information Technology

National Institute of Technology Karnataka, Surathkal.

June 2020

ABSTRACT

In this project, we propose an interactive Web-based multi-user document editing system in which users can cooperatively create and edit documents. Real-time collaborative editing allows multiple users to edit a shared document at the same time. It received a lot of attention from both industry and academia and gained in popularity due to the wide availability of free services such as Google Docs. While these collaborative editing systems were initially used in scenarios involving only a small set of users such as for writing a research article, nowadays we notice a change in the scale from several users to communities of users. Moreover, in this system, users can not only create but also store the data of the user to a MongoDB database and download the document to a local system. Also along with an interactive editing system, we have a chat function that will help the users to communicate with each other and all this happens in real-time using sockets!we have built a collaborative text editor with node and socket.io along with angular.js and basic HTML and CSS

TABLE OF CONTENTS

Chapter 1: Introduction

A Collaborative editor allows multiple users to edit the same document simultaneously, in real-time. Anyone from around the world can edit the text editor and collaborate with you as you write your article. The aim of this project is to develop a web app implementing a simple text-based collaborative editor with some basic content management features. A widely used and popular collaborative text editor is Google docs, which runs on a method known as Operational Transformation(OT). OT is a complex algorithm based solution that can be thought of like a conflict resolver in git. Depending on the operations performed by various users on the document OT performs changes that keeps the document in the same state for every user. The demerits of OT based solution is because of its complexity in implementing the algorithm. The main point here is just to detect when a character is typed in, then send the text over the socket stream. The server socket having a message event registered sends the text stream over to all connected sockets. Then, all users registered at the message event will get the text stream from the server, and set the text area with the text stream.

A collaborative real-time editor is a type of collaborative software or a web application which enables real-time collaborative editing, simultaneous editing, or live editing of the same digital document, computer file or cloud-stored data – such as an online spreadsheet, word processing document, database or presentation – at the same time by different users on different computers or mobile devices, with the automatic and nearly instantaneous merging of their edits.

A collaborative editor is a form of collaborative software application that allows several people to edit a document or a program collaboratively over a computer network. In real-time collaborative editing users can edit the same file simultaneously, whereas, in Non-real time collaborative editing, the users do not edit the same file at the same time. Since there is a very small number of open source solutions to achieve collaborative editing in any web application, we intend to develop a framework in JavaScript through which anyone can achieve this functionality easily. Real-time operation is an important aspect to be considered in the design of collaborative editors as users should be able to see the effects of their own actions immediately and those of other users as soon as possible. Together with this, a chat system is integrated to ease the communication between the collaborators. We will achieve this through socket.io which is a javascript library. Coming to the features and application,

A real-time collaborative integrated development environment can provide developers with the facility to collaborate over software projects over a network even when developers are thousands of miles away. Real-time Collaborative IDE provides developers with the ability to collaboratively write code, build and test it as well as share their projects with other developers. Chatting with other fellow developers over a project is also possible. Besides several other useful features of a complete IDE including saving snapshots, project management is also provided to ease the entire project development process.

Chapter 2: Literature Survey

Real-time collaborative editing has a long history of research in the computer-supported work community, going back to Engelbart's famous "mother of all demos" where he had demonstrated remote collaboration on a shared screen.

In a system for real-time collaboration, a protocol is required that allows multiple users to make changes concurrently. Operational transform or OT is one of the techniques used for maintaining a consistent view of the shared document. Previous work in collaborative editing systems has examined the problem of maintaining semantic consistency, for example in graphic design and hierarchical documents.

"Collaborative editing of a document" by Bharat V Bedi and Lucas W Patridge were among the first people to bring in the idea of a collaborative environment. They have briefly explained the idea of collaborative working in the initial days and how to improvise it. It has an in-depth description of how it works inside the computer system internally and the methods to improvise the usage of computer memory without having multiple companies of the same document.

In "Software Development with Real-Time Collaborative Editing" by Max Goldman, he has given a brief analysis of the necessity of collaborative editors and their impact on the users. He has given an analytical comparison of how effective it is when people are far away and working separately on the same project and merge it, people sit together and work on the same project on different systems and then merge it and finally when people use the same document and work together even by staying far away.

Chapter 3: Requirements Analysis

In order to ensure the fast transfer of operation from one client to another, we used socket.io between clients and servers. Below are the following minimum tasks that our application would need to perform:

- Get the user details.
- Have a text area to write the content.
- Make sure all the text-area is bound to each other so that when one changes it reflects in the other too.
- Option to download the data in the local system.
- Saves user data.
- Good UI for the user and easy to use.
- Chat system to let the users communicate.

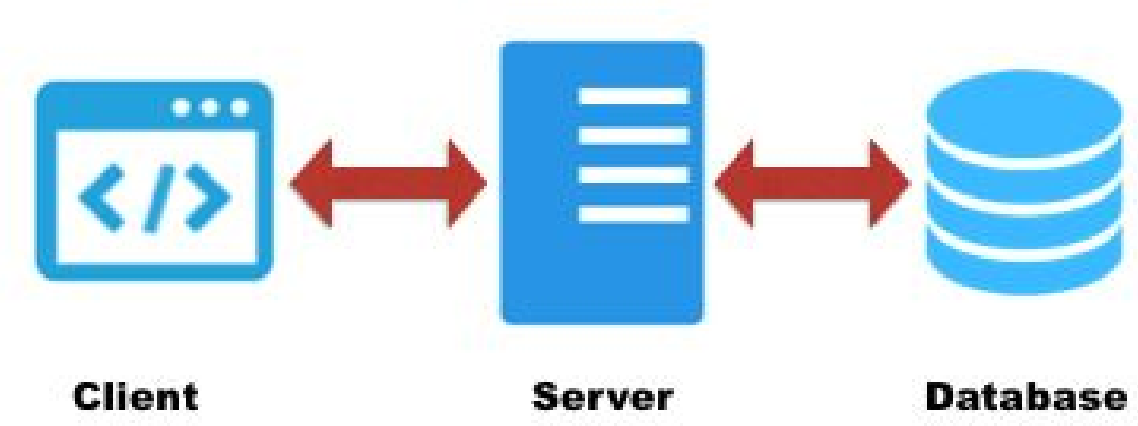
Hardware Requirements

- i3/i5/i5 core , 4/8/16/32 GB RAM ,500/1000/2000 GB memory.
- Windows /Linux/MAC OS system.

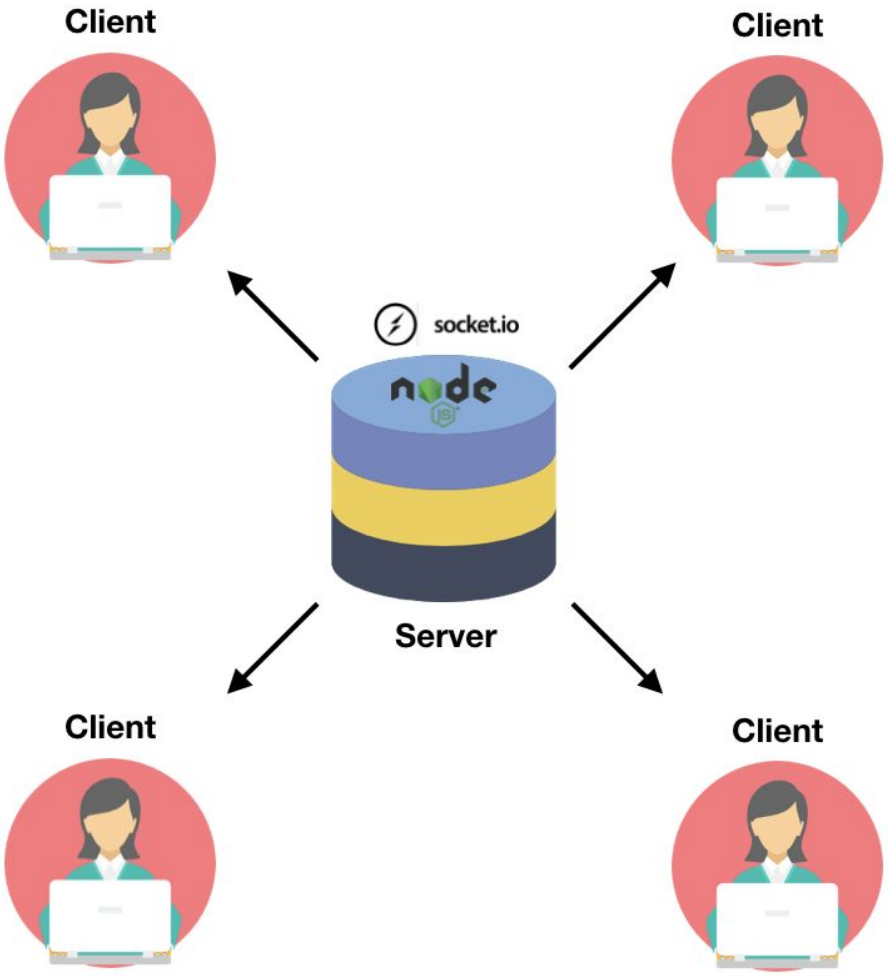
Software Requirements

- Knowledge of Javascript programming
- Libraries of Nodejs and dependencies
- Knowledge of Angularjs.
- MongoDB database.
- Basic HTML CSS Bootstrap.

Chapter 4: System Design/Architecture



Our design is dependent on a three-tier architecture. multiple clients are connected to the server and the server details are stored in the database. The server will be responsible for routing our socket streams to all other sockets connected to the server socket. The server will listen to the connection on a port, and establish io connections. The multiple clients are connected on the browser using the URL and the port number and now can communicate with each other and fulfill their tasks. The client has certain features like it allows you to browse a local file in your system and append it to the text area and it is reflected in all the clients connected to the server. This way we can share the file in one's local system to the team which is a very useful feature. there is an option on the client-side where we can download the written document. The download button downloads the file in the text area.



We have another feature that is the chat system. Along with letting the team collaborate we let them communicate using the inbuilt chat application. This chat application is also built on the socket.io and is a very useful feature for our application in order to communicate when we are parallelly working on the document. The server will be handled by node.js to make all the engineering (launch the packages and the website). This code will not be seen by the client. The client part will be loaded on the computer of the client. He will have direct access to the files (HTML/CSS and js).

Chapter 5: Methodology

In order to ensure high responsiveness, we will be using a replicated architecture where users will work on their own copy of the shared document. Only changes done are transmitted to other users.

Here is the control flow of such architecture:

- Each collaborator will hold a copy of the shared document along with the server.
- Whenever any of the shared copies will be edited, operations are calculated and transmitted to the server. To ensure real-time transmission, we will leverage the power of the socket.io library.
- In case different users are updating different parts of the same document, the server applies the changes to its copy and forwards the changes to other collaborators.

Running through the Socket.io library:

Socket.IO is a JavaScript library for realtime web applications. It enables real time, bi-directional communication between web clients and servers. What's really amazing is that you can connect to socket.io from an entirely different server.

The client requests the server to share a file, now the server will share the file basically by parsing it. It then adds a user whenever a client joins the server. All the collaborators will be notified about new collaborators in their chat box, the chat box also displays the status when a collaborator leaves the server.

- Client-server styled architecture platform.
- Chat system between the clients.
- A client can either view the file or edit which will be reflected in all the client's text area.

Together with this,the document can be saved to the database.we have used MongoDB for our database.It is a NoSQL database which stores the data in the form of JSON documents. Hence,makes it a wiser choice to store our files using the MongoDB database.

Chapter 6: Implementation

We started building a very basic collaborative editor with a simple UI using Socket.io.It has the features of creating a file,choosing any file from the system,on editing we can save back the file to the local system.

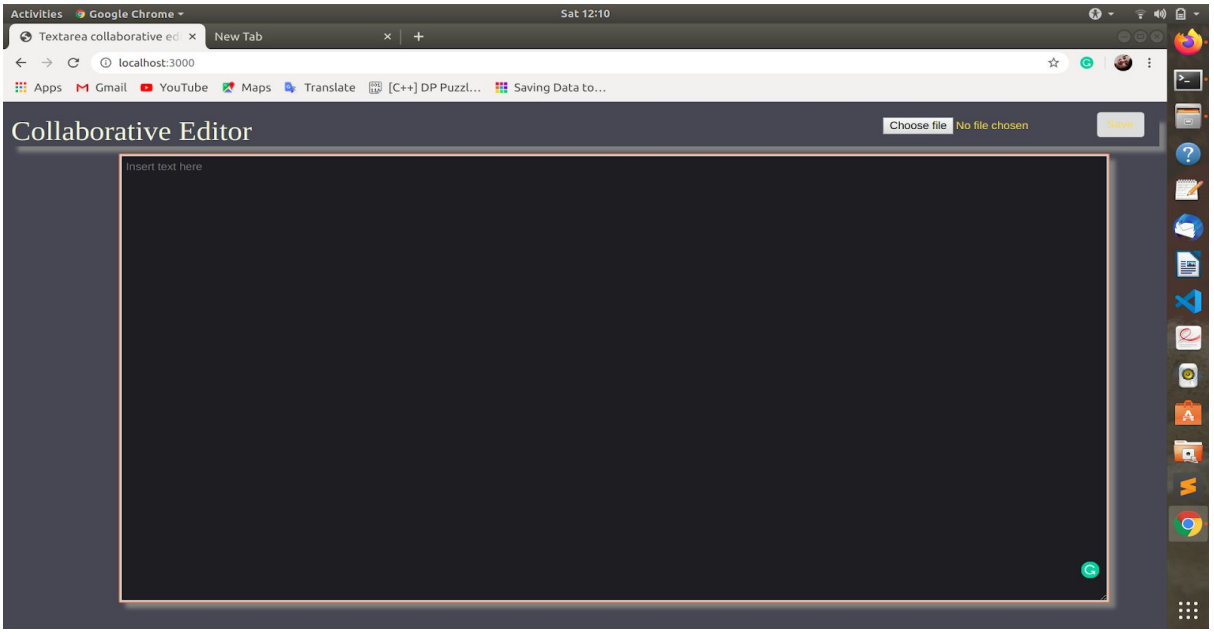


Fig 6.1:The initial UI of our application.we can create a file or choose any file from the system, edit it in our editor and save the file back to the local system.

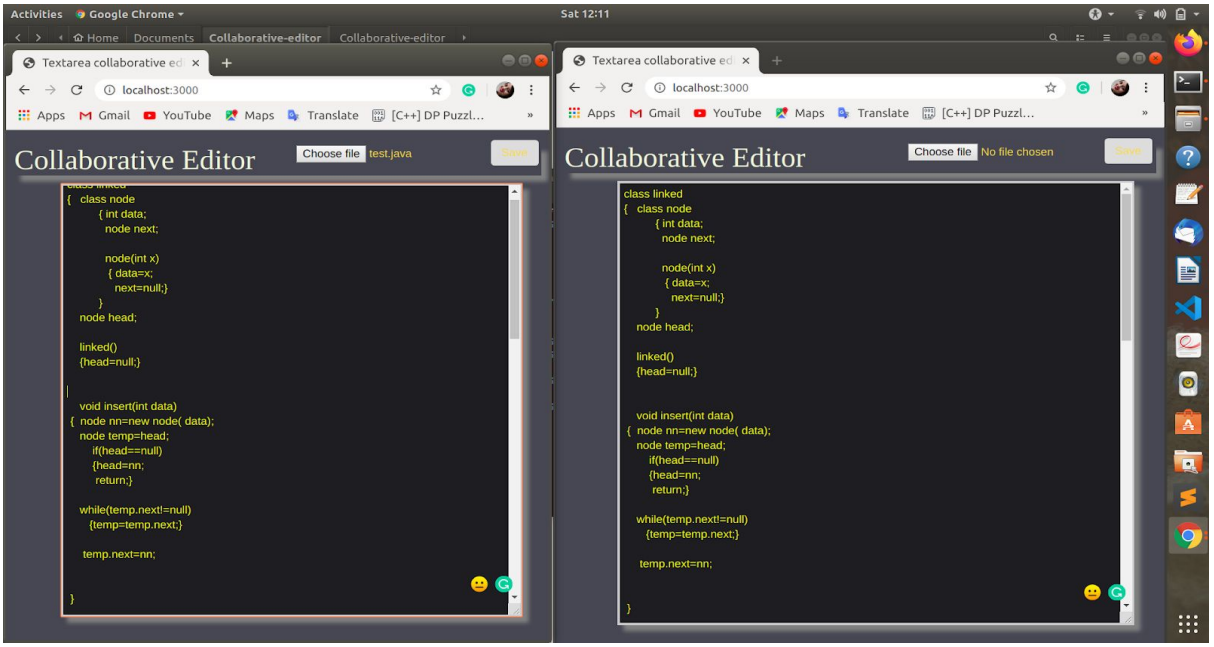


Fig 6.2: A file “test.java” is chosen from the system in the first tab.It shows the name of the file chosen.In another tab ,the same file is visible.Now ,both the editors can edit on the same file.

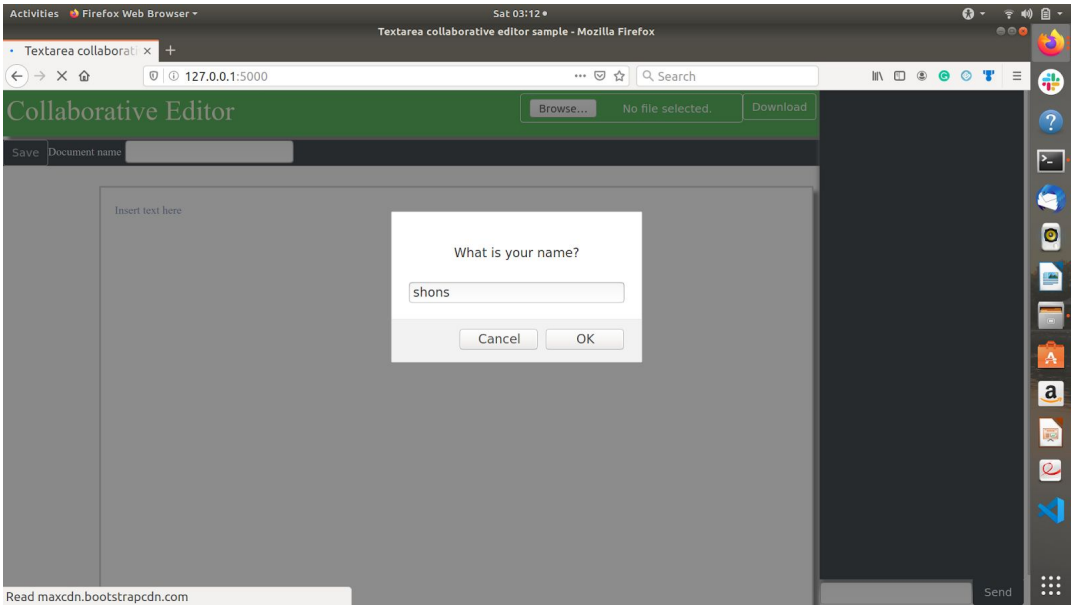


Fig 6.3: Improved UI asks for the name of the user.

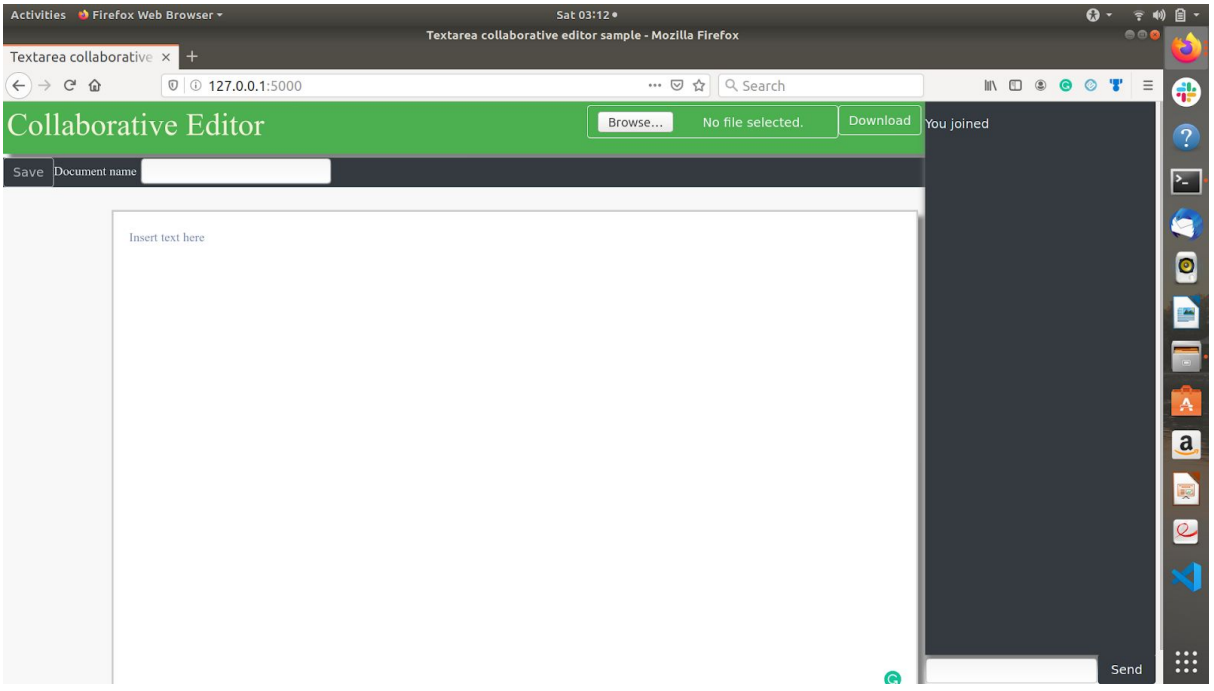


Fig 6.4: The landing page of the Improved UI where we have features like a chat option on the right side to make the communication between collaborators easier, and download option to save it to the local system and save option to save it to database.

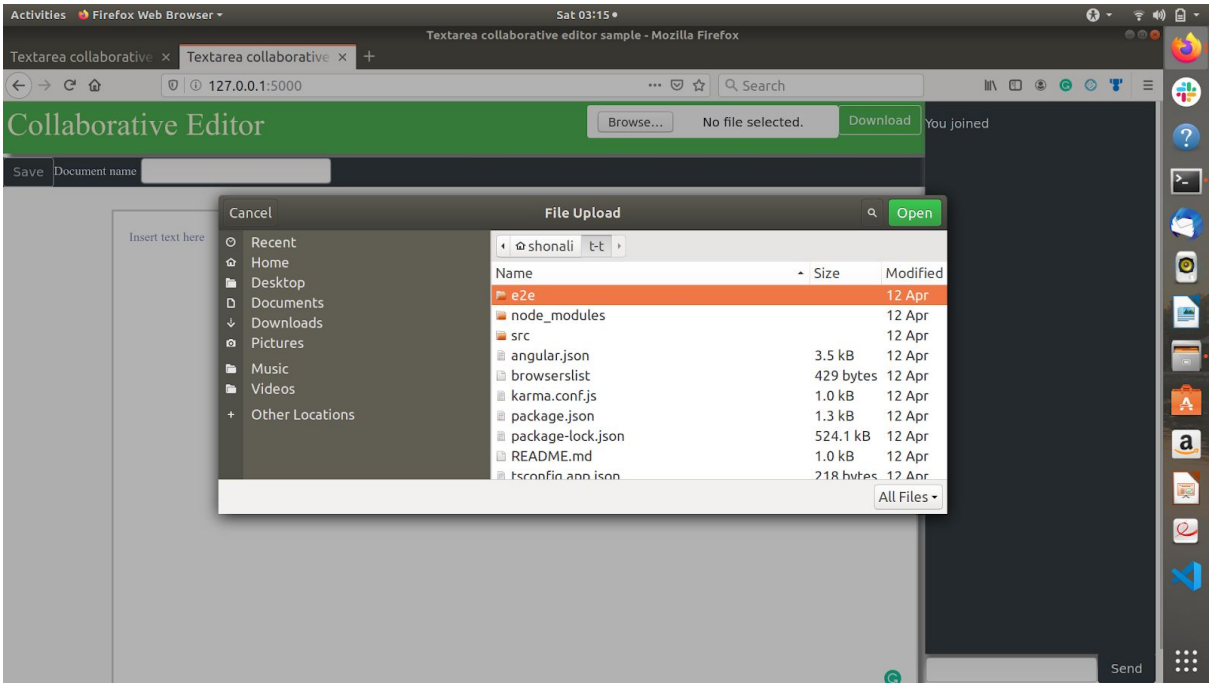


Fig 6.5: Browse option lets you choose a file from the local system.

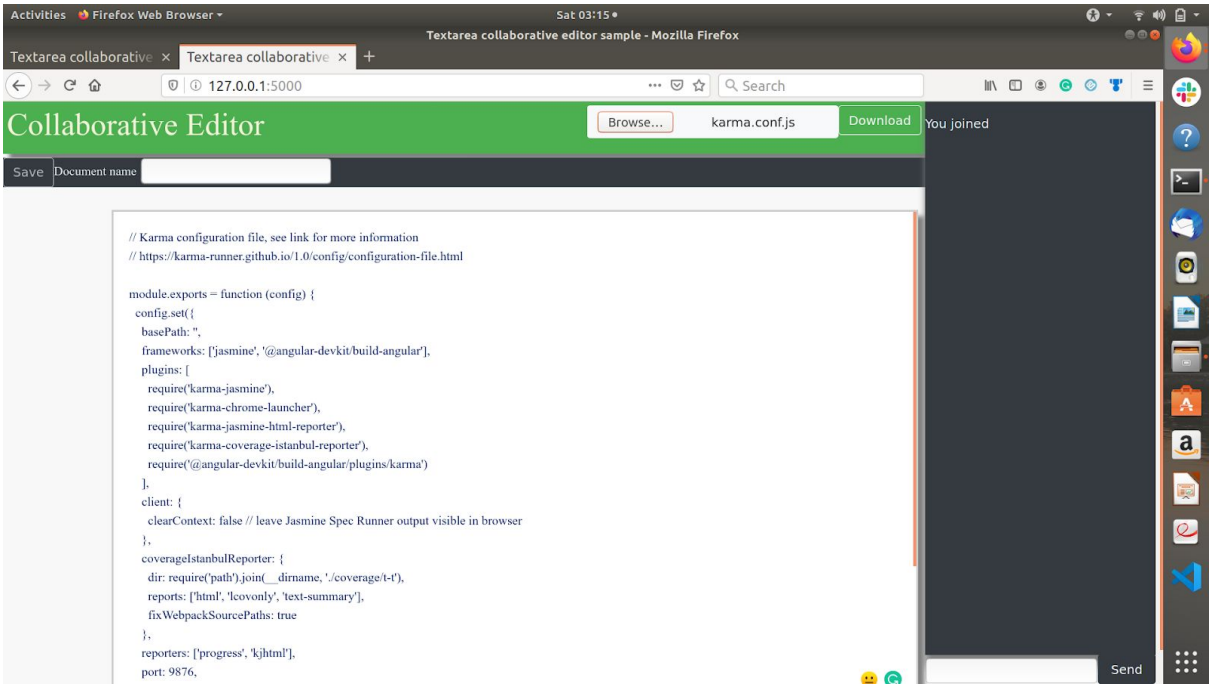


Fig 6.6 : After loading a file from the system

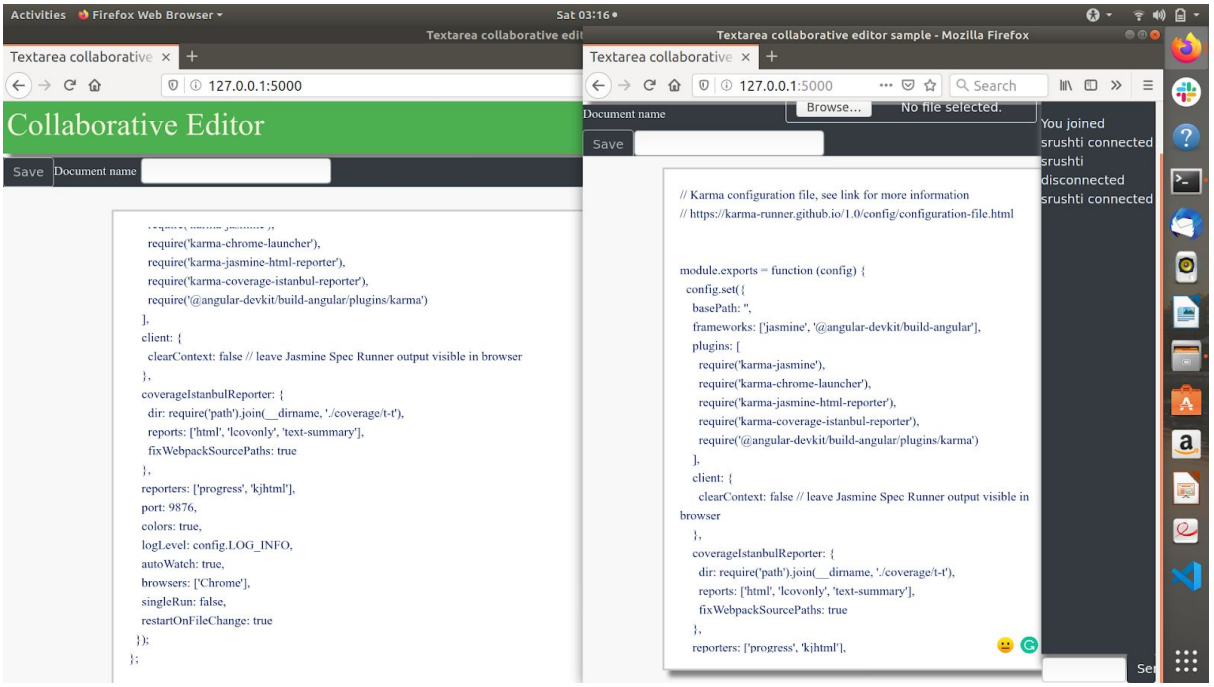


Fig 6.7: We can see the change has been reflected on both the clients interface in real time.

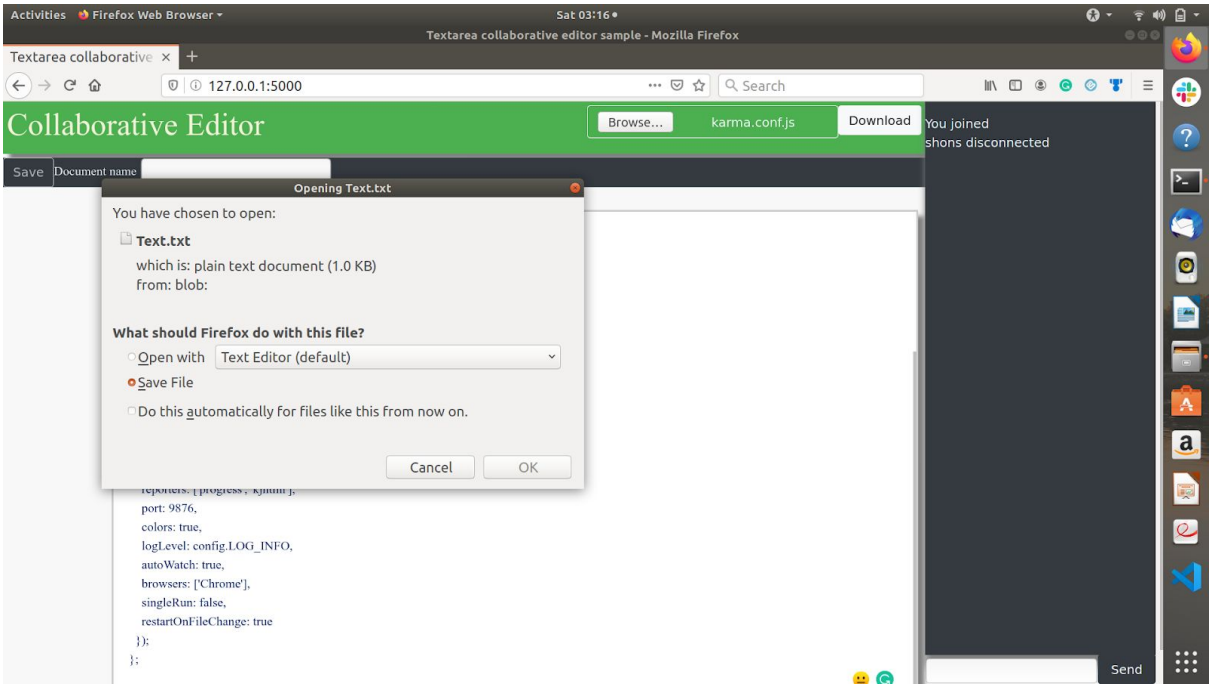


Fig 6.8: We can download the file to the local system using the download button and it will be stored in .txt format.

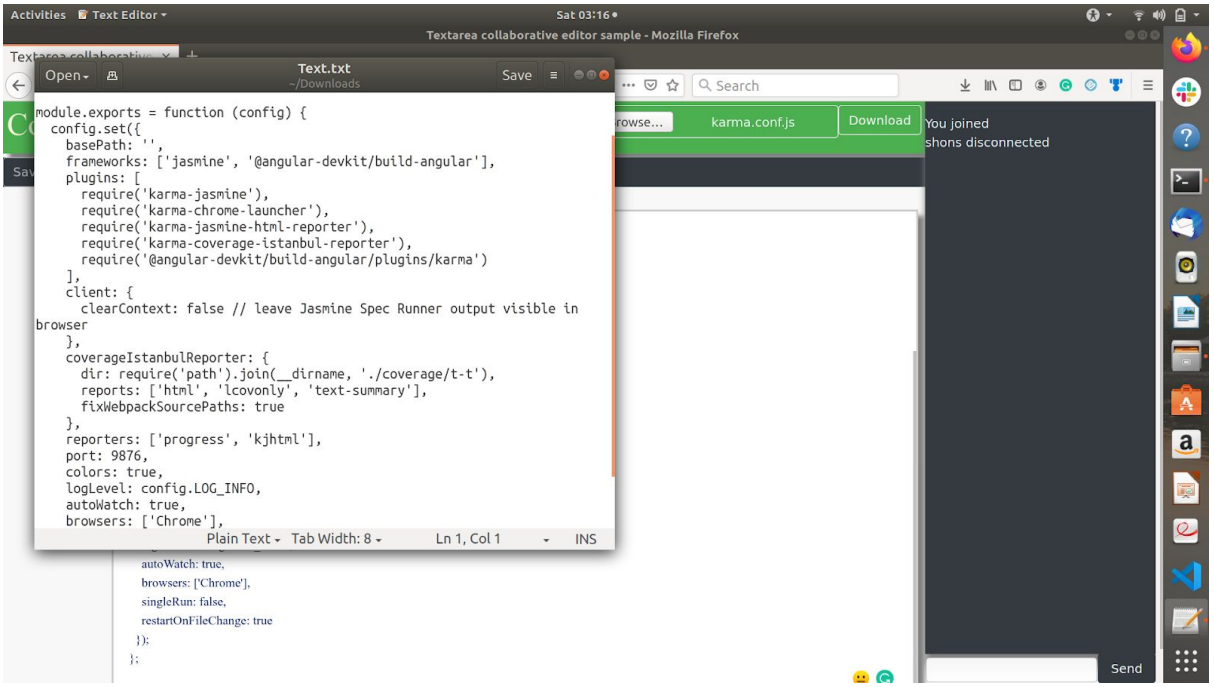


Fig 6.9: We can see it has stored it in the download directory of the local system in .txt format.

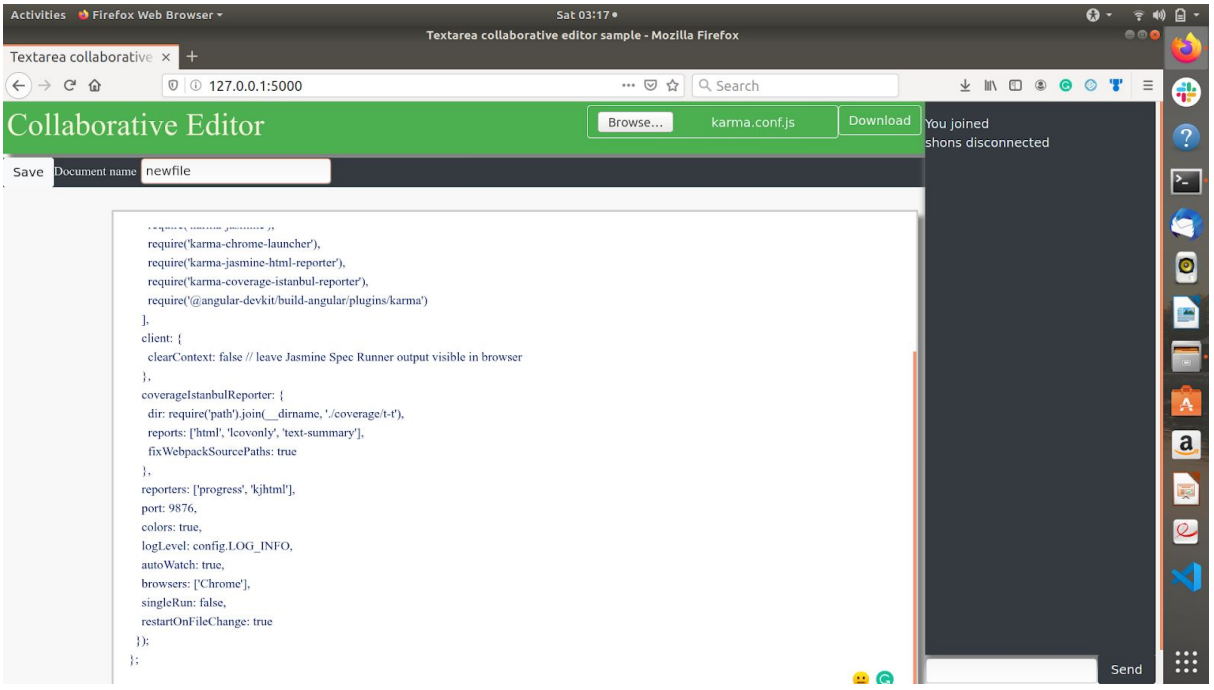


Fig 6.10: We can save it in the database too by clicking on save , the save button will be enabled only when we have filled the field of the document name , in the above picture the document name is “newfile” and we hit save.

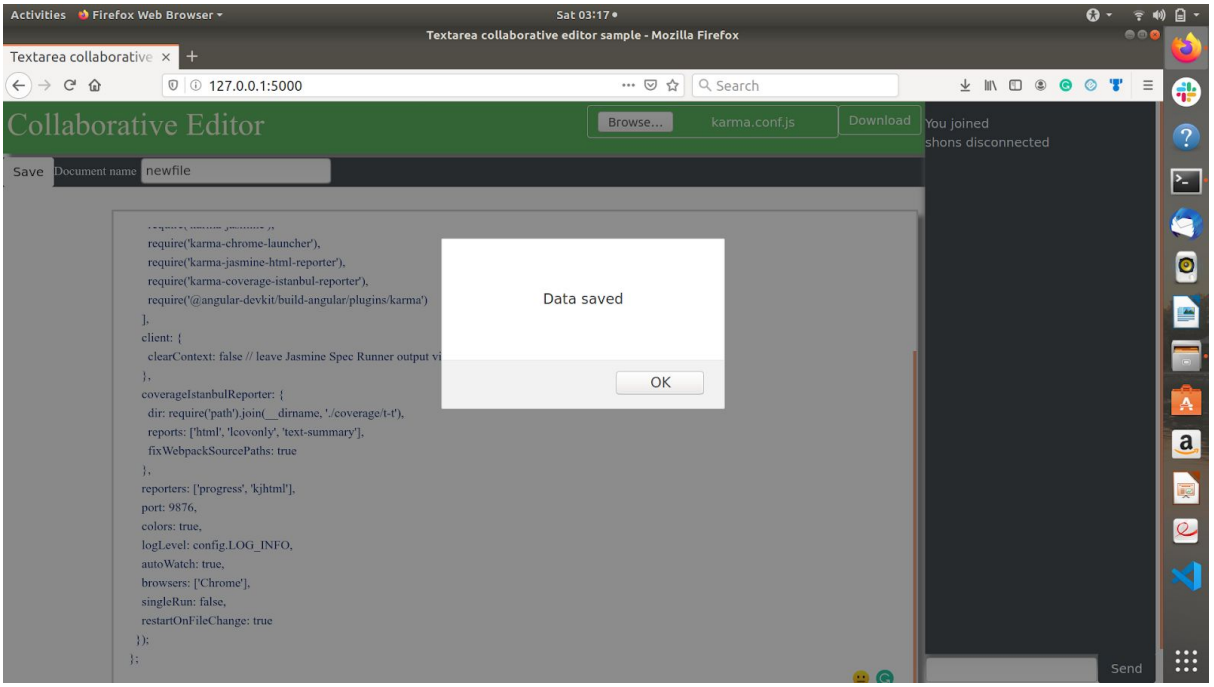


Fig 6.11: After clicking save we get a pop up message saying data saved.

Fig 6.12: We can see the post request made by our html , we store id, the document in textarea, the document name and version number of MongoDB.

Chapter 7:Results and Analysis

The final result of the web application have come up with the following result:-

- Quick collab: Any user can just hop on the website, enter a name and start collaborating on the document .The changes are reflected in all the clients connected to the same server . You can type out your article or import from the local system to share it to everyone , all you need to do is click on browse option and choose any file from your local system and it will be reflected on all the clients text area.
- Saving option: We have provided two options of saving, one to directly to your local system and second to the MongoDB database . To save it to your local system the option “download” is provided which downloads it to the local system in the .txt format. To be precise it will be stored as Text.txt in the downloads directory of the local system. “Save” option is used for saving the file into the database. It stores the document name, document file and _id of the object the changes are shown in the post request.
- Chat System: Web application also provides us with a real time chat application which allows us to communicate between each other during collaboration. This chat app is also connected through socket.io. The names of the collaborators who are joined are shown as the first person who has opened the server and started editing. This application can be very useful while working parallely on the editor.

From the above option we can see we have created a mini version of the google docs which was our inspiration for making this project.

Chapter 8: Conclusion

The project “Collaborative editor using socket.io” mainly focuses on making the collaborative environment more feasible and provides an effective method for collaborations. As noted by C. Sun and C. Ellis, the is complexity of real time text editing lies in concurrency control and that this continues to be a field in need of further research.A lot more improvements can be brought in like video calls integrated with the editors.we have integrated chat system ,which is also an effective way to communicate with the collaborators.The files are stored in the database and can also be stored in local system for further use.we can improve the system by providing documents with more features and authentication to each individual files where only the authors would be able to access the file.A lot of research is being done in this topic to improve the existing technology and hence,is a topic to be learnt and explored.

References

[1]“Max Goldman”,Software Development withReal-Time Collaborative Editing ,MIT

[2]https://medium.com/front-end-weekly/build-a-collaborative-rich-text-editor-with-node-js-and-socket-io-38ee25b6e315

[3] https://www.geeksforgeeks.org/project-idea-collaborative-editor-framework-real-time/

[4] https://www.pluralsight.com/guides/building-a-realtime-collaborative-editor-with-rethinkdb

[5] https://nodejs.org/en/

[6] https://socket.io/

[7] https://getbootstrap.com/

[8] https://socket.io/get-started/chat/