CHAT APPLICATION USING SOCKET IO AND REACT JS

ABSTRACT

Most often we see that nowadays with the increasing demands of efficient communication, there have been many applications on the internet that facilitates **chatting** amongst peers. A similar application with quite a **different** approach is what we have tried to implement with our Project.

The **CHAT-EASY** application that we have made focuses on an **anonymous** chatting prototype where people will not have to know each other's social media handles or their personal phone number but they can still talk to each other with just a unique id of each user logged into the application. This serves the purpose of **uniqueness** and also **confidentiality** amongst official conversations.

The project deals with the implementation of **Web Sockets** that provide negligible latency and fast transfer of messages, be it in a group or single conversations. The below mentioned report focuses on the technicalities and working of the app.

TABLE OF CONTENTS

1. INTRODUCTION

A user friendly designed application **CHAT EASY** focuses on peer to peer conversation, in groups as well as single person with **negligible latency** and fast transfer. With the help of React framework, every user has a 128 bit character generated **unique id** which makes it possible for the viewer to maintain his/her uniqueness in the app and all of his/her **contacts** and **conversation** history remains **saved** when he logs in to his/her account again with the same id.

Apart from this, we **do not store any other personal detail** of the user and hence provide a well-managed and confidential system of communication. There is **no limit** to the contacts one wants to add, all that is needed is the Unique ID of the person and then it can be added to the contact history with a name so as to make it easy to converse in the future.

1.1 RELATED WORK:

A lot of chatting applications are available on the internet that are used pretty much by everyone nowadays. Some of them include:

- WhatsApp by Facebook
- Messenger by Facebook
- Microsoft Teams by Microsoft
- Google Hnagouts by Google

The way in which our chat application differ from all of the above mentioned one's is on the basis of confidentiality and no personal details of the user. As we know that data is stored by all the big tech giants like Facebook, Google, our application retraces all such affirmations and just allows the user to communicate with minimum risk factors.

1.2 OBJECTIVES AND GOALS

To implement Web Sockets and integrate it with the React Frameworks to make a single page application that allows secure peer to peer communication and reduces the chances of data breaches.

1.3 FEATURES

The CHAT-EASY application provides many essential feature that we do not find in most of the popular chatting applications nowadays. Some of the following are:

1. **User Friendly Front End:** The front end design is simple subtle and user friendly.

2. **Confidential and Secure:** To create a new contact we need not know any personal phone number, the random generated id would work.

3. **Simple and Efficient:** The group texts are simple to create and can be created even if the other person does not have our contact added to theirs. Only the id must be known to us.

1.4 APPLICATIONS

The application can be used in almost all work fields and even amongst friends and families. Some of the major places it can be implemented is:

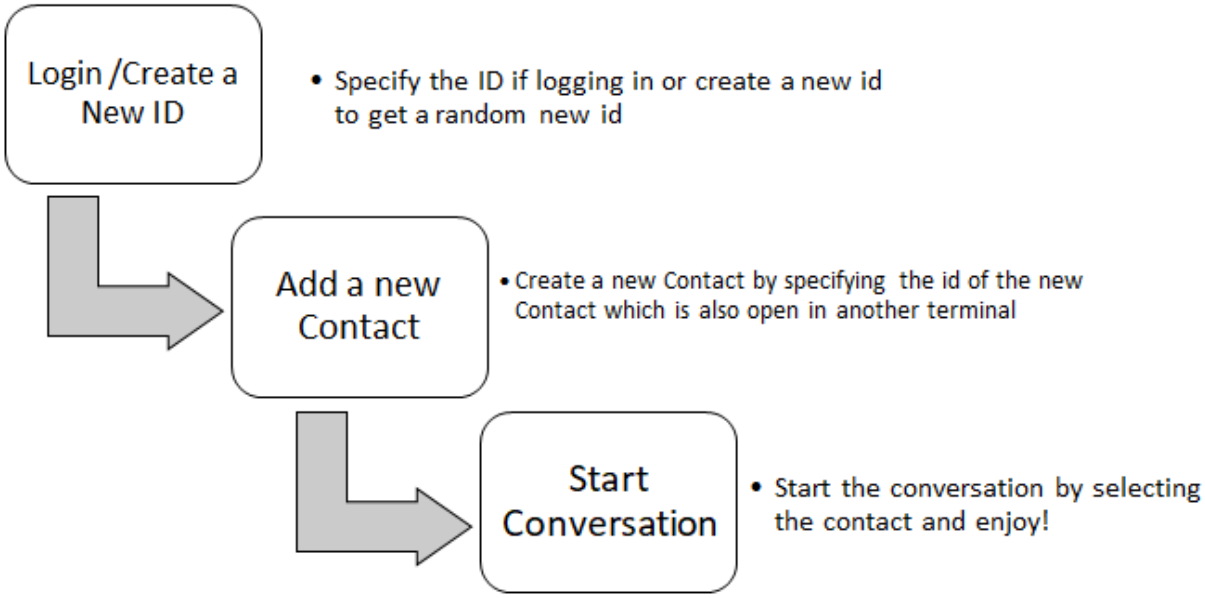
1. By army personnel to talk to their family when they are on their duty. Since usage of mobile phones with the same phone numbers are restricted for the heroes of the nation, with the use of internet, they can talk to their friends and family even when at work field as it ensures safe and secure way of communication.

2. Discussion of extremely secure project ideas and information that cannot be done on applications that store the message history and personal details like location and so on on their giant databases.

3. To plan surprise meetings and celebrations amongst family and friends without storing mobile numbers of so many new people.

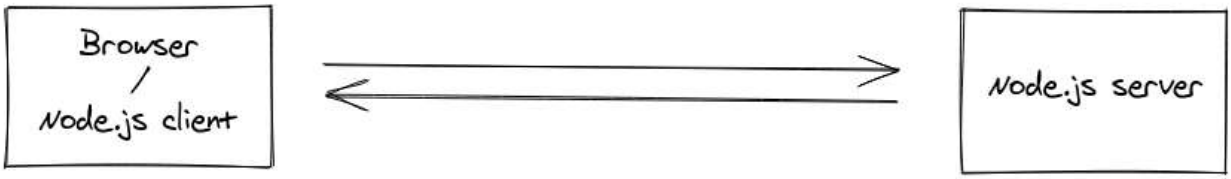
2 PROPOSED DESIGN

2.1 FLOW CHART



The basic prototype of the working of the chat easy app is the simplest of all. We just need to follow three steps and the work is done.

In the backend, the client-server communication takes place with the help of web socket integration.



The application works on a client-server model. The client will try to establish a Web Socket connection if possible, and will fall back on HTTP long polling if not.

2.2 SOFTWARE REQUIREMENTS

Front End Framework used:

- **React.Js:** React is a JavaScript library for building user interfaces.

Back End:

- **Node.js:** As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications.

Libraries used:

- **Socket.io:** Socket.IO is a library that enables real-time, bidirectional and event-based communication between the browser and the server.

2.3 IMPLEMENTATION METHODOLOGY

The application is built on a Client-Server model and runs on a common server for all the clients logged into the system. The two models contain different files that work on the intricacies of the application as shown below:

Client:

- **Login.js:** Frames a react rendered login page which has a login and a create new id button. The new id is randomly created by the uuid class available in react packages.

- **ConversationModal.js:** Whenever a new conversation is to be created, a dialog box appears that asks the user about whom he/she wants to chat with, the user can select the person(s) and start the new conversation.

- **ContactModal.js:** Determines the contact panel and has the feature of new contact addition which can be added just with the knowledge of the id of the other person and the name can be given by the user as he/she wants it to display.

- **NewConversation.js:** Determines the user interface when a new conversation is started.

- **OpenConversation.js:** When a Conversation is selected from the conversation panel, the format of the conversation is framed here.

- **useLocalStorage.js:** The local storage library of react is implemented to keep the conversation and contact history on even being refreshed.

- **ConversationsProvider.js:** It implements the client side socket implementation of messages storage and transfer over the server.

- **ContactsProvider.js:** Storage of the contacts pertinent to a particular id and also has the feature of actions to create new contact.

- **Index.js:** Binds all the react components and makes the app.js ready for implementation.

- **Dashboard.js and Sidebar.js:** Design of the Dashboard and the sidebar

- **Conversation.js and Contact.js:** Design of the Contact and Conversation feature.

Server Side:

- The client communication happens over the server: localhost:5000. All the socket.io implementation occurs over the server.

- Web Socket is a communication protocol which provides a full-duplex and low-latency channel between the server and the browser.

- Server.js: Implements the server side logic and the code of socket.io running on localhost:5000 and it has the feature to handle send and receive message.

2.4 RESULTS OF IMPLEMENTAION

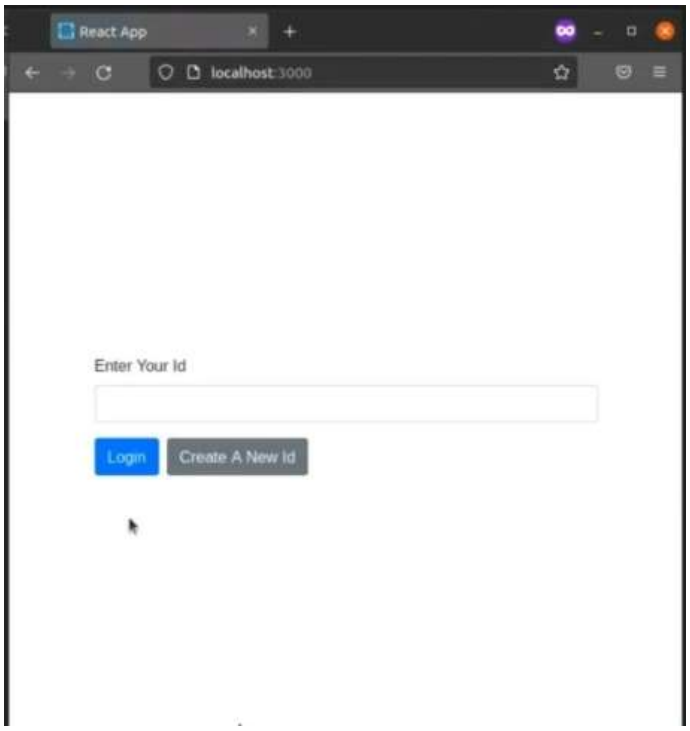
To the client terminal write:

- npm start: this starts the client side of the application on localhost:3000 and then create the id and start adding contacts and start the conversation

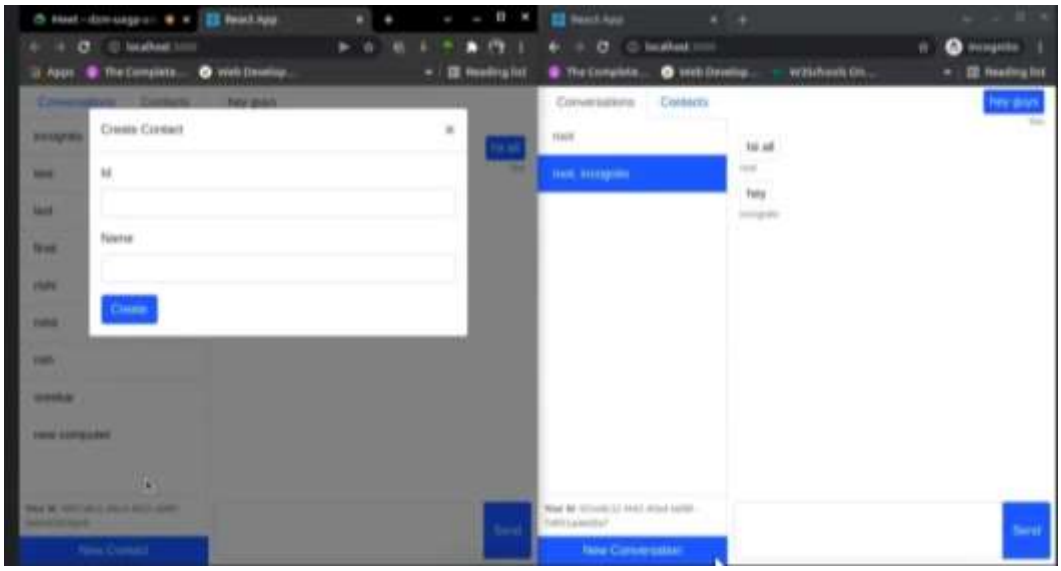
To the server terminal write:

- nodemon server.js: it starts the server on localhost: 5000 and implements the features of sending and receiving texts from one client to other over one common server.

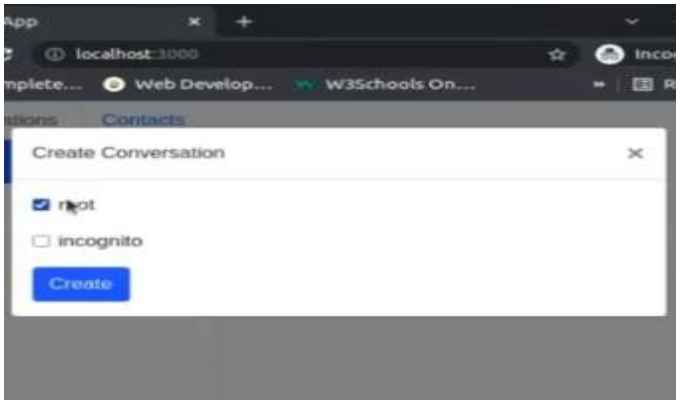
The application was successfully implemented and was running perfectly fine with each functionality.



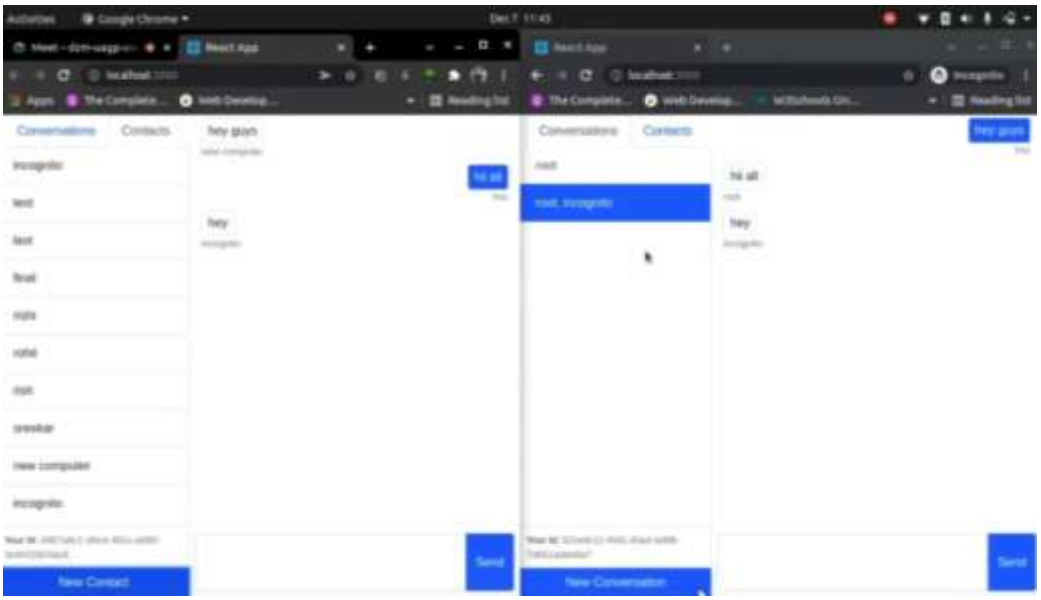
the login page that lets you login with the old id or create a new id



creating a new contact feature is displayed



new conversation creation looks like this and we can choose single/group conversations



the conversation panel in a group conversation between three people

2.5 TRADE OFFS & COST ANALYSIS

Before the cost analysis and deployment of the application, there are a few related factors that we must be aware of:

1. Requirements, OS versions and demography.
2. Extra App features, USP feature, Server backed or static app.
3. Geography of development that you would want to prefer.

Apart from all such information, the probable cost estimation of the deployment of a chat application to work full time and on any location, AWS offers different storage facilities that we can choose to deploy. As per the sources, “Chat app development cost can be starting from $15,000 with the basic features and can go to any expensive price, therefore it is important to plan your budget in advance and hire app development company to meet your app development needs.” (The link is mentioned in the references panel).

3. CONCLUSION

3.1 CONCLUSION & INFERENCE

The concept behind communication over the internet using web sockets was understood and also the implementation of it was successful. Moreover, the project involved the use of Github to integrate all the files that were being made by all the group members.

4. REFERENCES

a. **https://socket.io/**

b. **https://reactjs.org/docs/getting-started.html**

c. **https://react-bootstrap.github.io/**

d. https://medium.com/flutter-community/how-much-does-it-cost-to-develop-a-messaging-app-like-whatsapp-or-telegram-in-2020-760c9f58d71f