

PROJECT REPORT

On

“CHAT APP”

Department of Computer Engineering & Applications
GLA UNIVERSITY



GLA University
Mathura- 281406, INDIA
2022-2023

SUBMITTED BY:-

Akansh Jain (201500055)
Kaustubh Yadav (201500333)
Sourabh Thakur(201500703)

SUBMITTED TO:-

Mr. Akash Kumar Choudhary
(Technical Trainer)

Declaration

We hereby declare that the work which is being presented in the mini project “**Chat - App**”, in partial fulfilment of the requirements for project viva voce, is an authentic record of our own work carried by the team members under the supervision of our mentor Mr. Akash Kumar Choudhary.

Group Members:

Akansh Jain (201500055)

Kaustubh Yadav (201500333)

Sourabh Thakur(201500703)

Course: B.Tech (Computer
Science and Engineering)

Year:3rd

Semester: 6th

Supervised by –

Mr. Akash Kumar Choudhary.

(Technical Trainer)

GLA University, Department of Computer Engineering & Application.

OBJECTIVE

The main objective of the Full stack Chat project is to create a platform where people can easily communicate with each other in real time. The project provide an easy way for users to sign up and login, and should also allow them to create and manage their own chat rooms. The application will be built using a React front-end that can be used by anyone, anywhere. The project allows users to send and receive messages in real-time and will give a user-friendly interface for users to interact with.

Development Tools

Languages:

- React
- Nodejs
- Expressjs.
- We used VS Code IDE as our editor.
- Connection Built with Socket.io.
- Data Stored with MongoDB and MongoDB Compass.

Software and Hardware Requirements-

- An internet connection.
- A Mac, Linux, or Windows 10 or Windows 11 computer.
- A web browser like Chrome or Microsoft Edge.

Features

1. Secure Log in-out (password encryption)
2. Secure Chats (End to End Encrypted)
3. Simple and User-Friendly UI
4. Free to use.
5. Not restricted to any one browser.

Introduction

A chat app, also known as a messaging app, is a software application that allows users to exchange text messages, voice messages, images, videos, and other types of media in real-time over the internet. Chat apps have become increasingly popular as a means of communication, both for personal and business use. They are typically designed to be user-friendly and accessible on a variety of devices, including smartphones, tablets, and computers.

Chat apps can be used for a variety of purposes, such as casual conversations with friends and family, staying in touch with coworkers or clients, or connecting with customers through customer service channels. Some chat apps also offer additional features such as group chats, voice and video calls, and file sharing.

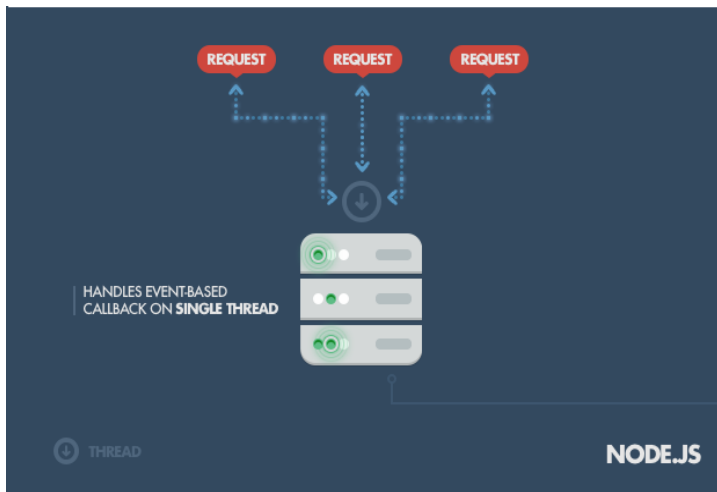
The popularity of chat apps has led to the development of numerous different options, ranging from simple, basic apps to more advanced platforms with a wide range of features. Some of the most popular chat apps include WhatsApp, Facebook Messenger, iMessage, Slack, and Microsoft Teams, among many others.

Hypothesis

Node.js –

Node JS is a JavaScript platform for building fast, scalable, network applications built on Google's V8 Engine. Node is single threaded and built around the paradigm of nonblocking IO. With Node.js each incoming request by the user is handled by one single thread in opposition to the multithreaded techniques used by PHP to scale the operations. Each request handled by this thread is coupled with a callback function that is called upon completion of the task. This is possible due to the fundamental support of JavaScript for events, Asynchronous operations, and callbacks; and Node.js puts JavaScript on the server side.





Express.js.

Although Node is capable of independently act as a web server, there are frameworks designed to make it more powerful and efficient the most popular being Express.js. It is fast, minimal, and has several applications built for it. Express was chosen for several reasons among them are:

- MongoDB Minimalistic. Express makes it possible to build an HTTP server very easily by wrapping the backend code of Node.js, and it makes the building of a RESTful API very simple.
- Express supports middlewares which

are functions called in a sequential order on a request or response. For example, a middleware I am using is body parser, which parses the body of incoming HTTP requests with a form submission. This allows me to access the parameters of the request directly. Middlewares can do anything, and they end with `next()`, which calls the next middleware.

MONGO DB

Additionally on the backend, there should be a database that stores user data. It would be preferable to use a JSON NoSQLbased database to leverage the benefits of using JavaScript across the stack when the same objects stored in the DB can be processed by the server and the frontend without any additional conversion. I started my testing with a pioneer of NoSQL databases, and a member of the MEAN stack, MongoDB. Mongo is just the database software itself, and a driver is needed to connect node with the database instance and provide a layer

for I/O with the database. Preferable also, there should be defined schemas for the database documents.

There are several advantages for using MongoDB such as:

- On its own, MongoDB is schema less, which provides the users with an easier way to append objects into the database. It also certain documents within the collection may have different values from others. For example, a user can have as many number of phone numbers' fields in their document as they own without any need to change the other documents in the same collection.
- Indexing. This is a feature of MongoDB in which any field that is 'required' in a collection, if indexed, would be added to a sorted array with the values of this field in all documents. This way searching the collection can be made very quickly.

React

React is a declarative, efficient, and flexible JavaScript library for building user interfaces. 'V' denotes the view in MVC. ReactJS is an opensource, componentbased front end library responsible only for the view layer of the application. It is maintained by Facebook.

React uses a declarative paradigm that makes it easier to reason about your application and aims to be both efficient and flexible. It designs simple views for each state in your application, and React will efficiently update and render just the right component when your data changes. The declarative view makes your code more predictable and easier to debug.

A React application is made of multiple components, each responsible for rendering a small, reusable piece of HTML. Components can be nested within other components to allow complex applications to be built out of simple building blocks. A component may also maintain an internal state for example, a TabList

component may store a variable corresponding to the currently open tab.

COMPONENTS

React code is made of entities called Components.

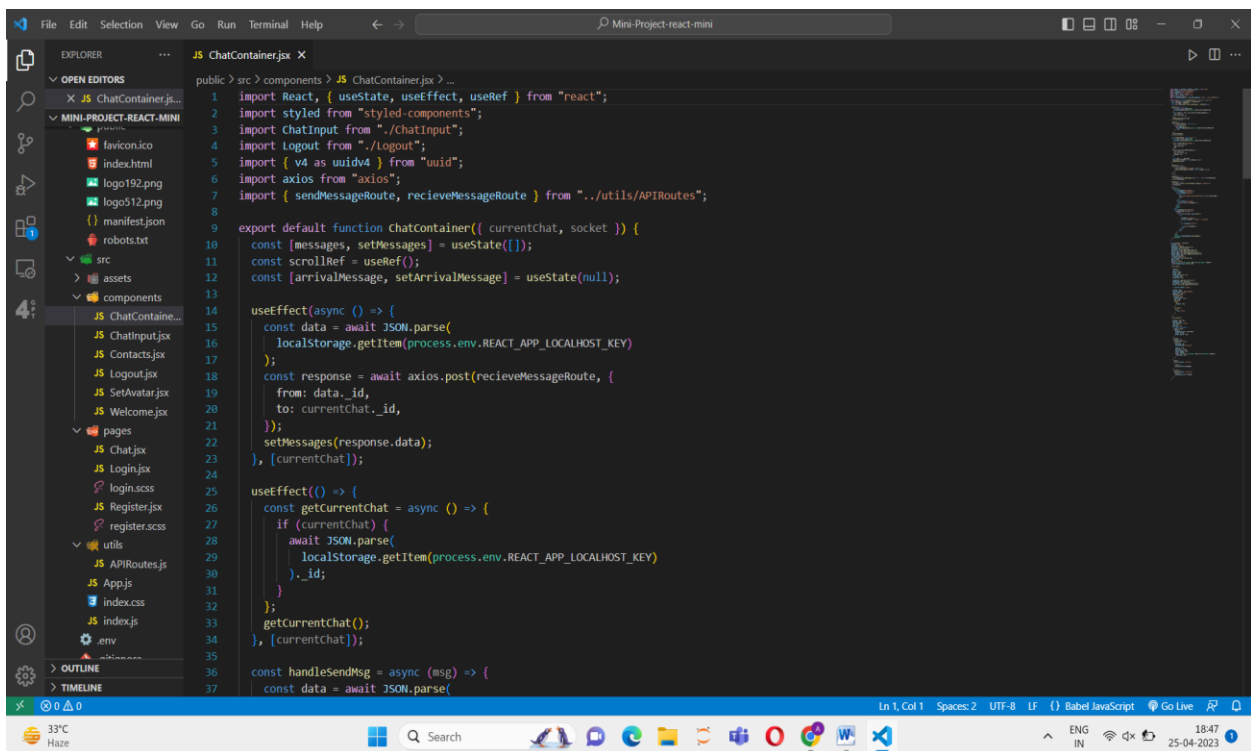
These components are reusable and must be formed in the SRC folder following the Pascal Case as its naming convention (capitalize camelCase).

Components can be rendered to a particular element in the DOM using the React DOM library. When rendering a component, one can pass the values between components through "props"

Components Used-

1.Chat Container-

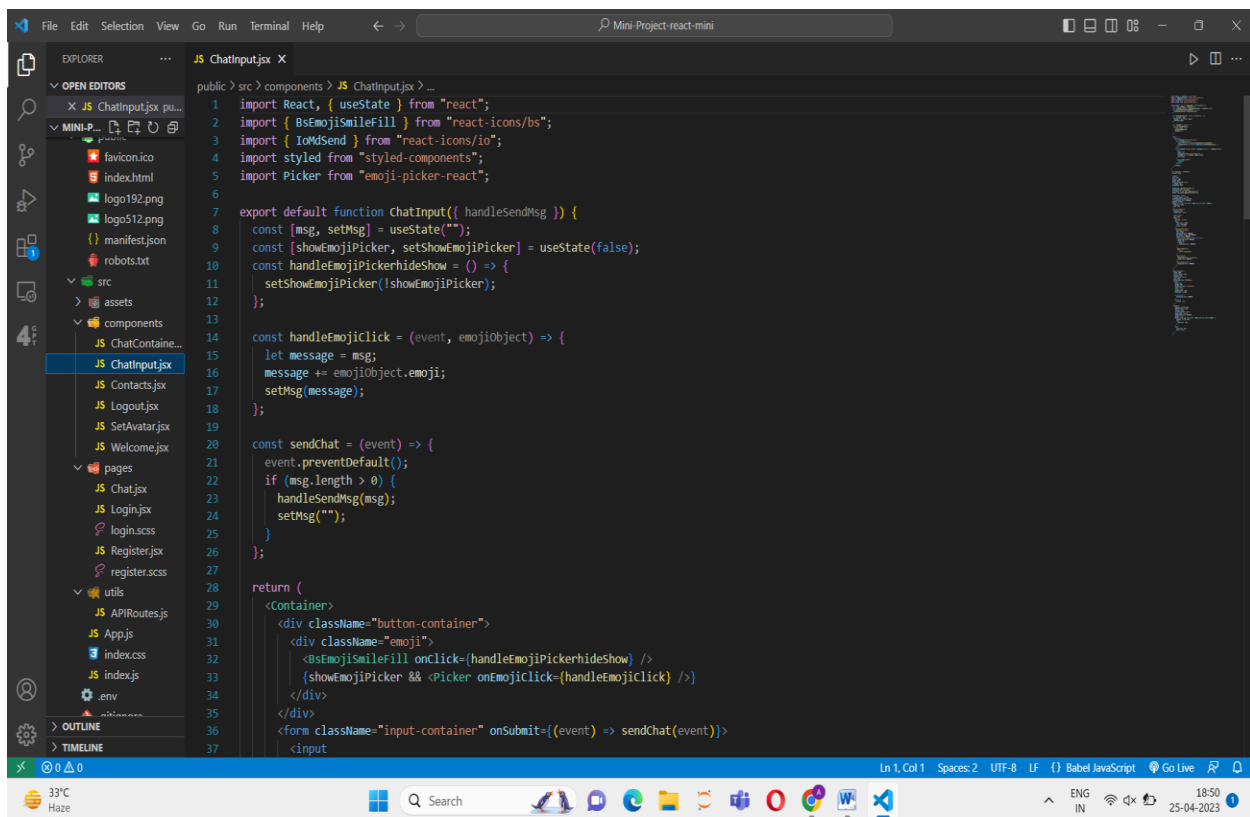
A chat container component is a user interface element used to display a conversation between two or more people in a chat application. The component typically consists of a message input area, a message history area, and other controls that allow users to interact with the conversation.



```
public > src > components > JS ChatContainer.jsx > ...
1  import React, { useState, useEffect, useRef } from "react";
2  import styled from "styled-components";
3  import ChatInput from "../chatInput";
4  import Logout from "../logout";
5  import { v4 as uuidv4 } from "uuid";
6  import axios from "axios";
7  import { sendMessageRoute, recieveMessageRoute } from "../utils/APIRoutes";
8
9  export default function ChatContainer({ currentChat, socket }) {
10     const [messages, setMessages] = useState([]);
11     const scrollRef = useRef();
12     const [arrivalMessage, setArrivalMessage] = useState(null);
13
14     useEffect(async () => {
15         const data = await JSON.parse(
16             localStorage.getItem(process.env.REACT_APP_LOCALHOST_KEY)
17         );
18         const response = await axios.post(recieveMessageRoute, {
19             from: data_id,
20             to: currentChat_id,
21         });
22         setMessages(response.data);
23     }, [currentChat]);
24
25     useEffect(() => {
26         const getCurrentChat = async () => {
27             if (currentChat) {
28                 await JSON.parse(
29                     localStorage.getItem(process.env.REACT_APP_LOCALHOST_KEY)
30                 )._id;
31             }
32             getCurrentChat();
33         }, [currentChat];
34     }, [currentChat]);
35
36     const handleSendMsg = async (msg) => {
37         const data = await JSON.parse(
```

2. Chat Input –

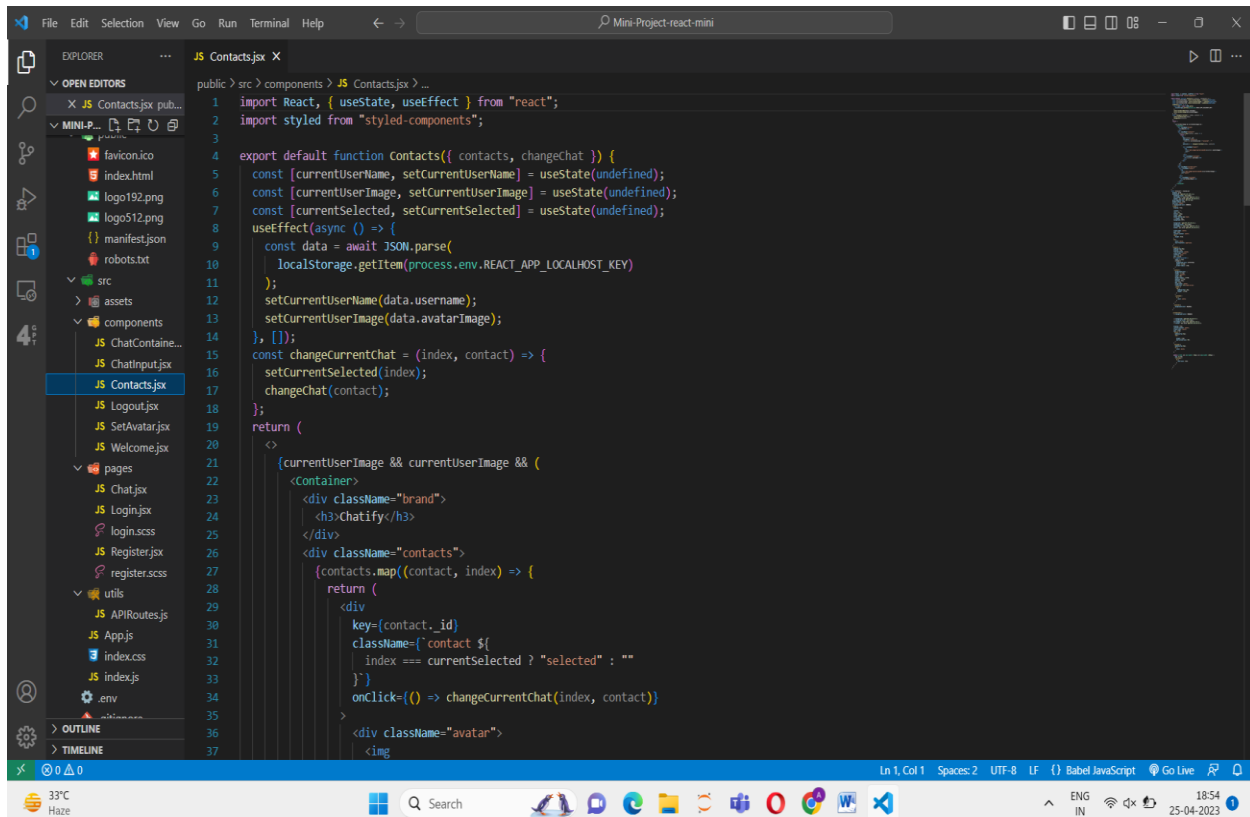
A chat input component is a user interface element that allows users to type and send messages in a chat application. It typically includes an input field where users can type their message, along with buttons or other controls that allow users to send the message or perform other actions.



```
public > src > components > JS ChatInput.jsx > ...
1  import React, { useState } from "react";
2  import { BsEmojiSmileFill } from "react-icons/bs";
3  import { IoMdSend } from "react-icons/io";
4  import styled from "styled-components";
5  import Picker from "emoji-picker-react";
6
7  export default function ChatInput({ handleSendMsg }) {
8    const [msg, setMsg] = useState("");
9    const [showEmojiPicker, setShowEmojiPicker] = useState(false);
10   const handleEmojiPickerhideShow = () => {
11     setShowEmojiPicker(!showEmojiPicker);
12   };
13
14   const handleEmojiClick = (event, emojiObject) => {
15     let message = msg;
16     message += emojiObject.emoji;
17     setMsg(message);
18   };
19
20   const sendChat = (event) => {
21     event.preventDefault();
22     if (msg.length > 0) {
23       handleSendMsg(msg);
24       setMsg("");
25     }
26   };
27
28   return (
29     <Container>
30       <div className="button-container">
31         <div className="emoji">
32           <BsEmojiSmileFill onClick={handleEmojiPickerhideShow} />
33           {showEmojiPicker && <Picker onEmojiClick={handleEmojiClick} />}
34         </div>
35       </div>
36       <form className="input-container" onSubmit={event => sendChat(event)}>
37         <input
```

3. Contacts –

In this contact components it allows users to store and organize information about their contacts and see all their contacts to chat .

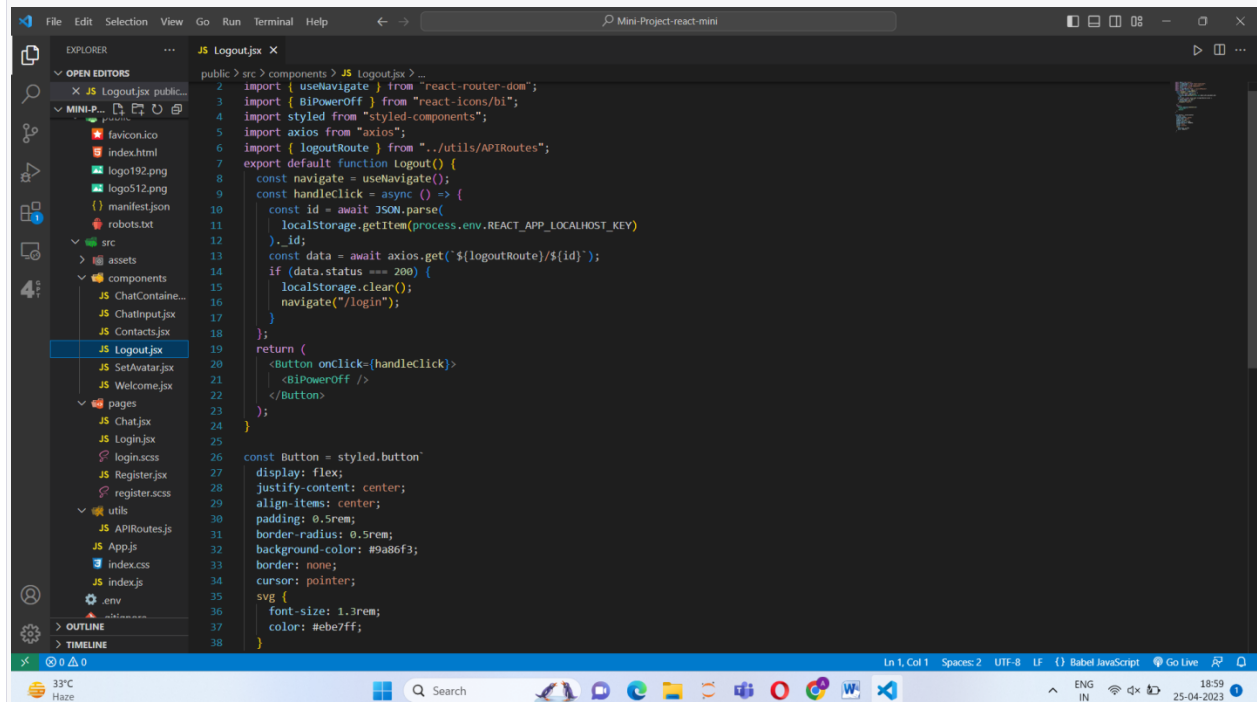


```
1 import React, { useState, useEffect } from "react";
2 import styled from "styled-components";
3
4 export default function Contacts({ contacts, changeChat }) {
5   const [currentUserName, setCurrentUserName] = useState(undefined);
6   const [currentUserImage, setCurrentUserImage] = useState(undefined);
7   const [currentSelected, setCurrentSelected] = useState(undefined);
8   useEffect(async () => {
9     const data = await JSON.parse(
10       localStorage.getItem(process.env.REACT_APP_LOCALHOST_KEY)
11     );
12     setCurrentUserName(data.username);
13     setCurrentUserImage(data.avatarImage);
14   }, []);
15   const changeCurrentChat = (index, contact) => {
16     setCurrentSelected(index);
17     changeChat(contact);
18   };
19   return (
20     <>
21     {currentUserImage && currentUserImage && (
22       <Container>
23         <div className="brand">
24           <h3>Chatify</h3>
25         </div>
26         <div className="contacts">
27           {contacts.map((contact, index) => {
28             return (
29               <div
30                 key={contact.id}
31                 className={contact.id === currentSelected ? "selected" : ""}
32               >
33                 <img
34                   className="avatar">
```

4.LOGOUT-

This component is designed to allow users to securely log out of the application when they are finished using it or when they need to switch to a different account.

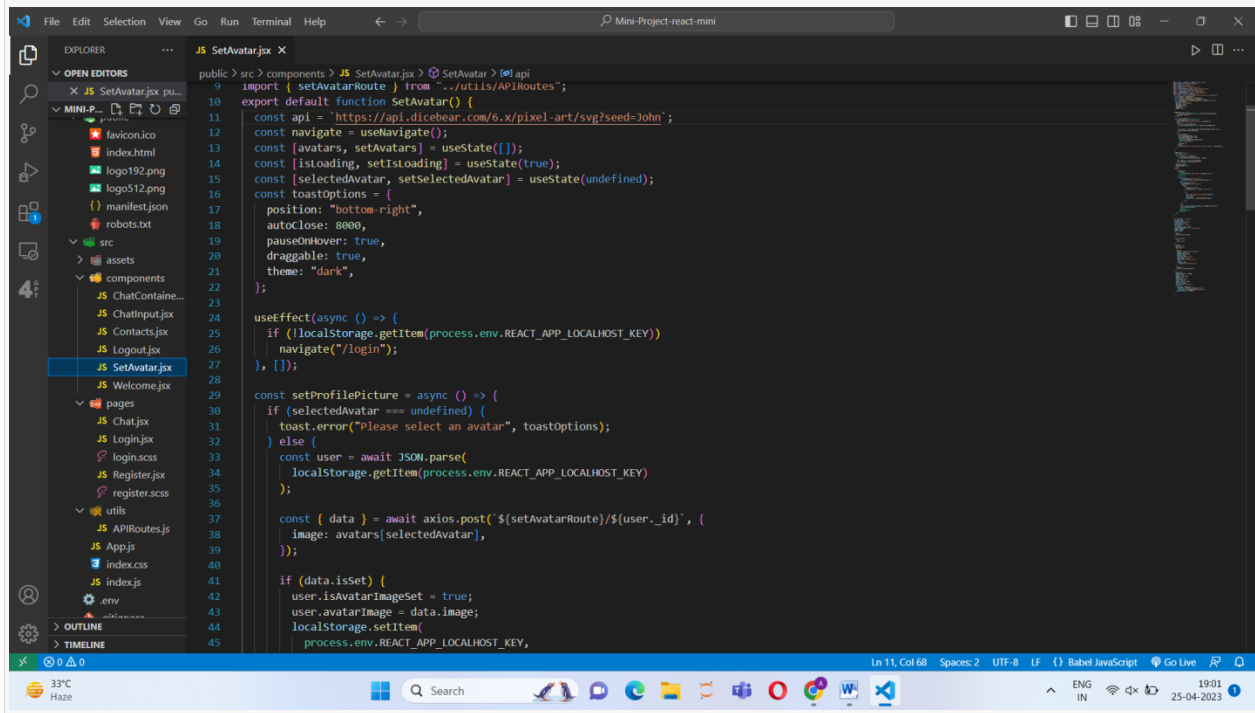
In general, a logout component is a button or link that is displayed on the application's user interface, typically in a header or navigation bar. When a user clicks on the logout button or link, the application will terminate the user's current session and log them out of the system.



```
1 import { useNavigate } from "react-router-dom";
2 import { BiPowerOff } from "react-icons/bi";
3 import styled from "styled-components";
4 import axios from "axios";
5 import { logoutRoute } from "../utils/APIRoutes";
6 export default function Logout() {
7   const navigate = useNavigate();
8   const handleClick = async () => {
9     const id = await JSON.parse(
10       localStorage.getItem(process.env.REACT_APP_LOCALHOST_KEY)
11     )._id;
12     const data = await axios.get(`${logoutRoute}/${id}`);
13     if (data.status === 200) {
14       localStorage.clear();
15       navigate("/login");
16     }
17   };
18   return (
19     <Button onClick={handleClick}>
20       <BiPowerOff />
21     </Button>
22   );
23 }
24
25 const Button = styled.button`
26   display: flex;
27   justify-content: center;
28   align-items: center;
29   padding: 0.5rem;
30   border-radius: 0.5rem;
31   background-color: #9a86f3;
32   border: none;
33   cursor: pointer;
34   font-size: 1.3rem;
35   color: #ebe7ff;
36 `;
```


5.Set Avatar-

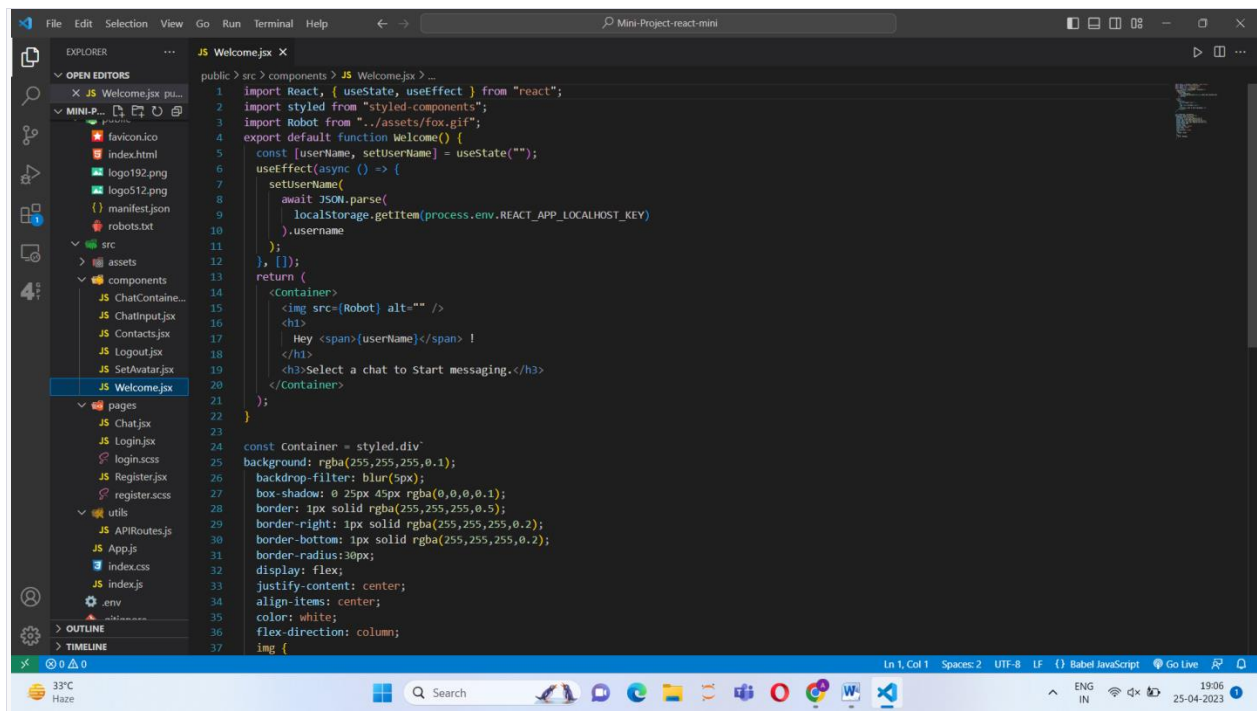
The set avatar component is a feature commonly found in applications that allow users to create profiles or accounts. This component is designed to allow users to upload or select a profile picture or avatar to represent themselves on the platform.



```
JS SetAvatar.jsx
public > src > components > JS SetAvatar.jsx > SetAvatar > @api
9 import { setAvatarRoute } from '../utils/APIRoutes';
10 export default function SetAvatar() {
11   const api = 'https://api.dicebear.com/6.x/pixel-art/svg?seed=John';
12   const navigate = useNavigate();
13   const [avatars, setAvatars] = useState([]);
14   const [isLoading, setIsLoading] = useState(true);
15   const [selectedAvatar, setSelectedAvatar] = useState(undefined);
16   const toastOptions = {
17     position: 'bottom-right',
18     autoClose: 8000,
19     pauseOnHover: true,
20     draggable: true,
21     theme: 'dark',
22   };
23   useEffect(async () => {
24     if (!localStorage.getItem(process.env.REACT_APP_LOCALHOST_KEY))
25       navigate('/login');
26   }, []);
27   const setProfilePicture = async () => {
28     if (selectedAvatar === undefined) {
29       toast.error('Please select an avatar', toastOptions);
30     } else {
31       const user = await JSON.parse(
32         localStorage.getItem(process.env.REACT_APP_LOCALHOST_KEY)
33       );
34       const { data } = await axios.post(`${setAvatarRoute}/${user._id}`, {
35         image: avatars[selectedAvatar],
36       });
37       if (data.isSet) {
38         user.isAvatarImageSet = true;
39         user.avatarImage = data.image;
40         localStorage.setItem(
41           process.env.REACT_APP_LOCALHOST_KEY,
42           JSON.stringify(user)
43         );
44       }
45     }
46   };
47 }
```

6.Welcome-

This components welcomes the user on successful login.

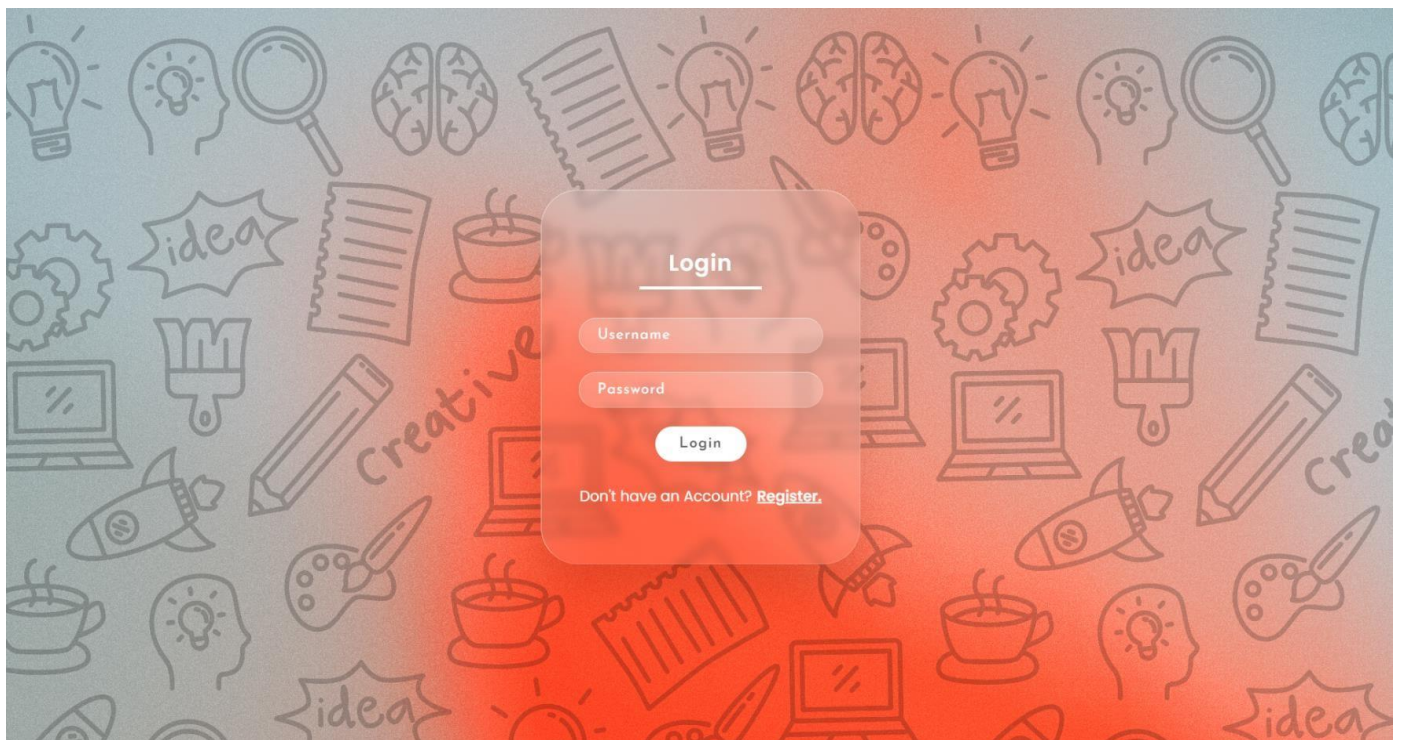


```
1 import React, { useState, useEffect } from "react";
2 import styled from "styled-components";
3 import Robot from "../assets/fox.gif";
4 export default function Welcome() {
5   const [username, setUsername] = useState("");
6   useEffect(async () => {
7     setUsername(
8       await JSON.parse(
9         localStorage.getItem(process.env.REACT_APP_LOCALHOST_KEY)
10      ), username
11    );
12   }, []);
13   return (
14     <Container>
15       <img src={Robot} alt="" />
16       <h1>
17         Hey <span>{username}</span> !
18       </h1>
19       <h3>Select a chat to Start messaging.</h3>
20     </Container>
21   );
22 }
23
24 const Container = styled.div`
25   background: rgba(255,255,255,0.1);
26   backdrop-filter: blur(5px);
27   box-shadow: 0 25px 45px rgba(0,0,0,0.1);
28   border: 1px solid rgba(255,255,255,0.5);
29   border-right: 1px solid rgba(255,255,255,0.2);
30   border-bottom: 1px solid rgba(255,255,255,0.2);
31   border-radius:30px;
32   display: flex;
33   justify-content: center;
34   align-items: center;
35   color: white;
36   flex-direction: column;
37   img {
```

SNAPSHOTS :

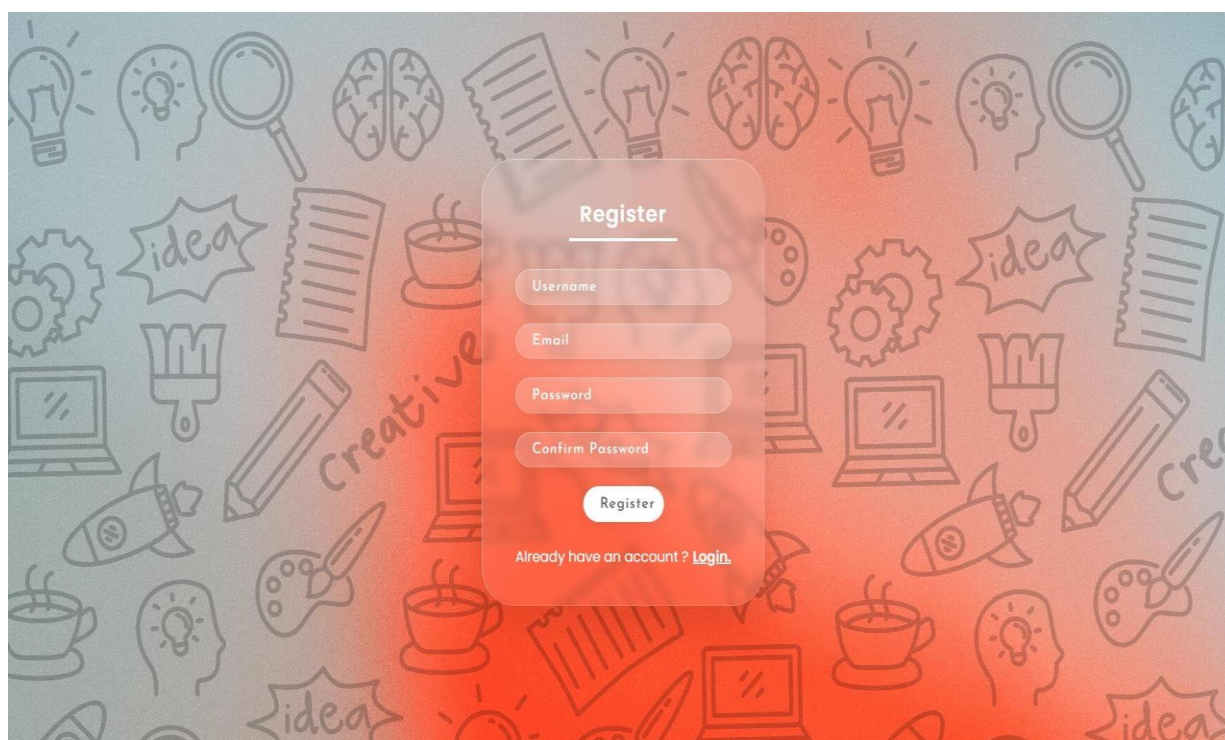
Login page –

The login page is one of the most commonly used features of a web application. It allows users to authenticate themselves before they can access the rest of the application. A login page typically has a form where the user can enter their username and password. Once the user has entered their credentials, the form is submitted to the server for authentication. If the authentication is successful, the user is redirected to the main application page. If the authentication fails, the user is typically given an error message and the opportunity to try again.



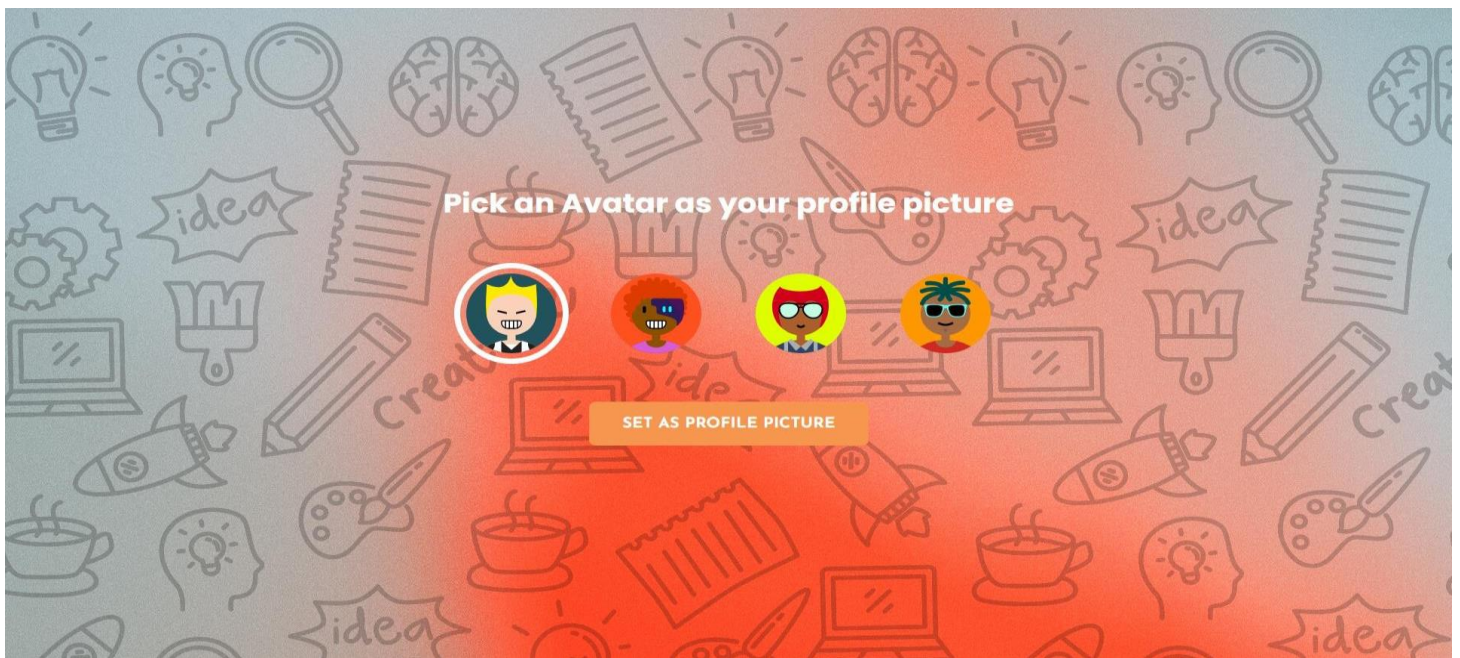
Registration Page

The registration page is a web page on a website that allows users to sign up for a user account on that website. This page typically contains a form that allows the user to enter their desired username, password, email address, and other personal information. Once the user has filled out the form and submitted it, they will usually be redirected to a confirmation page that tells them their account has been created and provides further instructions on how to activate it.



Choose Avatar

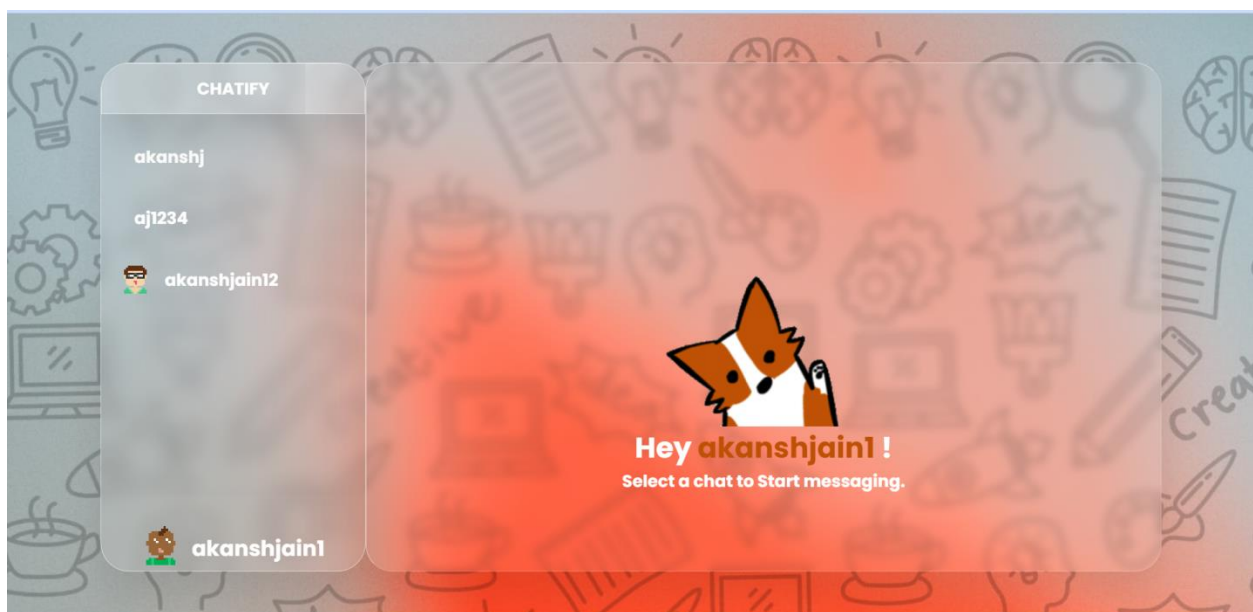
This page comes right after the registration page allows user to choose an avatar for their profile picture



Dashboard

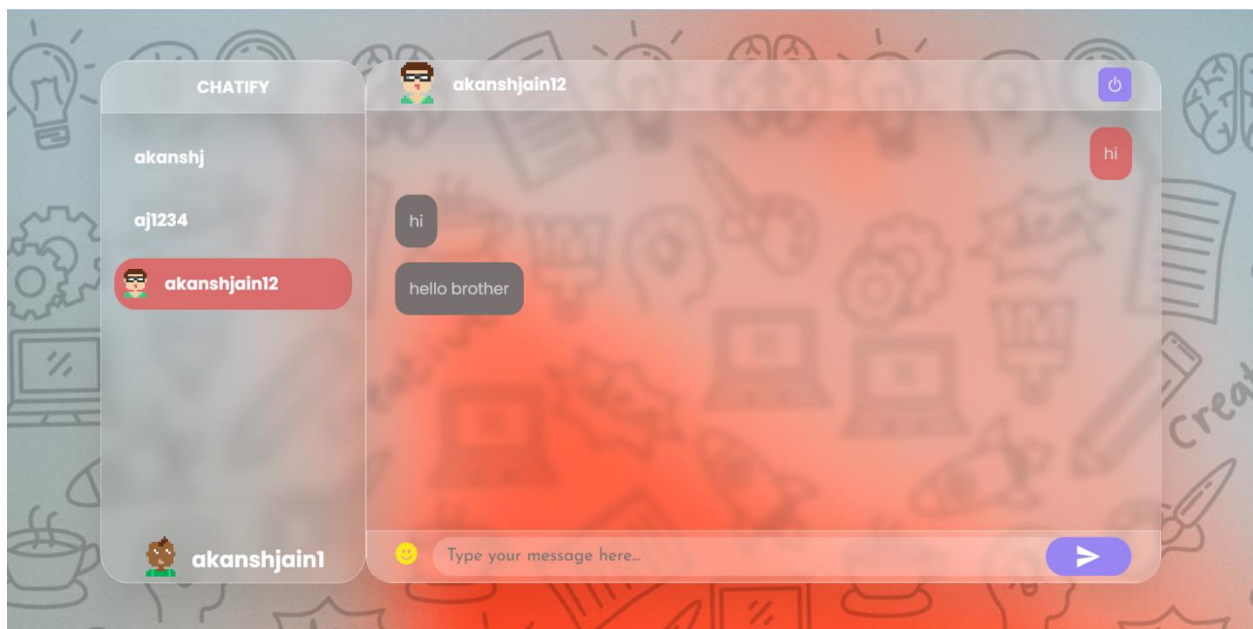
This is where user can see themselves to the very first interface of the application . Usernames are searchable and they can use them to connect with each other with their respective e mails through which users are logged in .

- Users can add friends to the ir chat app account by sending them a friend request.
- Users can view all their friends in a list on their main chat app page.
- Users can view all the conversations they have with their friends in a list on their main chat app page.



Chat Window-

This is where all the chats and contacts of an user shows .



Log out Page

This is the button where user can log out of their account.



Future Scope

- Customized Service .
- Marketing .
- Social Networking .
- Business.

Conclusion

We have completed our project within time limit with the coordination of our team members under the supervision of our mentor Mr. Akash Kumar Choudhary.

Our project repository is available at-

<https://github.com/sourabhthakur87/Real-Time-Chat-App/tree/main>

References

Books –

- Full – Stack
Modern Full- Stack Development.
Pro MERN Stack.
- React
The Road to Learn React.
React Explained.
- Full-Stack React Projects

Websites-

- <https://reactjs.org/>
- <https://www.w3schools.com/>
- <https://getbootstrap.com/>

THANK YOU