

INTERNSHIP REPORT on

FULL STACK DEVELOPMENT WITH MERN

A REPORT SUBMITTED TO JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY,
KAKINADA IN PARTIAL FULFILLMENT OF AWARD OF DEGREE

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEER



SUBMITTED BY

BATTU ANUSHA	(21HU5A0402)
KOMMURU PRAVALLIKA	(20HU1A0405)
KOTAPATI INDRANI	(20HU1A0406)
SHAIK SANJARI MEHTAB	(21HU5A0414)

UNDER ESTEEMED GUIDENCE OF

Mr. K. MANOJ PAVAN KUMAR M.TECH , Asst.professor

CHEBROLU ENGINEERING COLLEGE

(Approved by AICTE New Delhi, Affiliated to JNTU-Kakinada)

ACCREDITED NAAC 'A' GRADE,UGC-AUTONOMOUS

Door No:2B, Main Road, Chebrolu, Chebrolu, Guntur

522212

CHEBROLU ENGINEERING COLLEGE

(Approved by AICTE, New Delhi & Affiliated to JNTU Kakinada, A.P)

CHEBROLU (Village & Mandal), GUNTUR-522212



CERTIFICATE

This is to certify that the Internship report entitled **“FULL STACK DEVELOPMENT (MERN)”** is a piece of project work done by **BATTU ANUSHA (21HU5A0402)**, **KOMMURU PRAVALLIKA (20HU1A0405)**, **KOTAPATI INDRANI (20HU1A0406)** and **SHAIK SANJARI MEHTAB (21HU5A0414)**. Under the guidance of **Mr. K. MANOJ PAVAN KUMAR** M.Tech, Asst.professor. at the **DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING** for the degree of Bachelor of Technology of JNTU- KAKINADA, AP,INDIA.

The project viva-voice Exam is held on _____ of _____, 2023/2024.

Project Guide

Head of thDepartment

External Exam

DECLARATION

We hereby declare that the project report entitled “**FULL STACK DEVELOPMENT (MERN)**” with reference to JNTU KAKINADA is carried out by us under the guidance of **Mr. K. MANOJ PAVAN KUMAR** M. Tech , **Asst.professor**. We also declare that this project report is a result of our own effort and were not submitted to any other completion of bachelor of technology in **ELECTRONICS AND COMMUNICATION ENGINEERING**.

SIGNATURES

BATTU ANUSHA	(21HU5A0402)
KOMMURU PRAVALLIKA	(20HU1A0405)
KOTAPATI INDRANI	(20HU1A0406)
SHAIK SANJARI MEHTAB	(21HU5A0414)

ACKNOWLEDGEMENT

We express our profound gratitude to the principal **Dr.R.V. KRISHNAIAHM.Tech**, ^{PhD} & to the Director **M. SRINIVAS, MSC**, ^{M.Phil.} for holistic support in project work.

We would like to express our sincere thanks to **CH. HARI BABU**^{M.Tech}, Head of the department of **ELECTRONICS AND COMMUNICATION ENGINEERING** for support in project work.

We express our gratitude and acknowledgement our deep indebtedness to our project guide **Mr. K. MANOJ PAVAN KUMAR** ^{M. Tech, Asst.professor}. Her valuable suggestions and guidance helped us a lot in completing and presenting on “**FULL STACK DEVELOPMENT (MERN)**”.

Finally, we wish to express our whole gratitude to our **PARENTS** and all our **FRIENDS** without whose encouragement would not have been able to complete this work.

WITH REGARDS

BATTU ANUSHA	(21HU5A0402)
KOMMURU PRAVALLIKA	(20HU1A0405)
KOTAPATI INDRANI	(20HU1A0406)
SHAIK SANJARI MEHTAB	(21HU5A0414)

ABSTRACT

This abstract explores the role of full stack developers in the dynamic landscape of social media applications. With the exponential growth of social media platforms, the demand for versatile developers capable of handling both front-end and back-end development has surged. Full stack developers are integral in crafting seamless user experiences while ensuring robust server-side functionalities.

This paper delves into the essential skills and technologies required for full stack development, including proficiency in languages such as JavaScript, Python, and SQL, as well as frameworks like React and Django. Additionally, it examines the unique challenges faced by full stack developers in the realm of social media, such as scalability, security, and real-time data processing.

By employing agile methodologies and continuous integration practices, full stack developers contribute to the rapid iteration and evolution of social media applications, enabling them to stay ahead in a competitive market. Through case studies and industry insights, this abstract sheds light on the pivotal role of full stack developers in shaping the future of social media platforms.

Social Media App

Introduction:

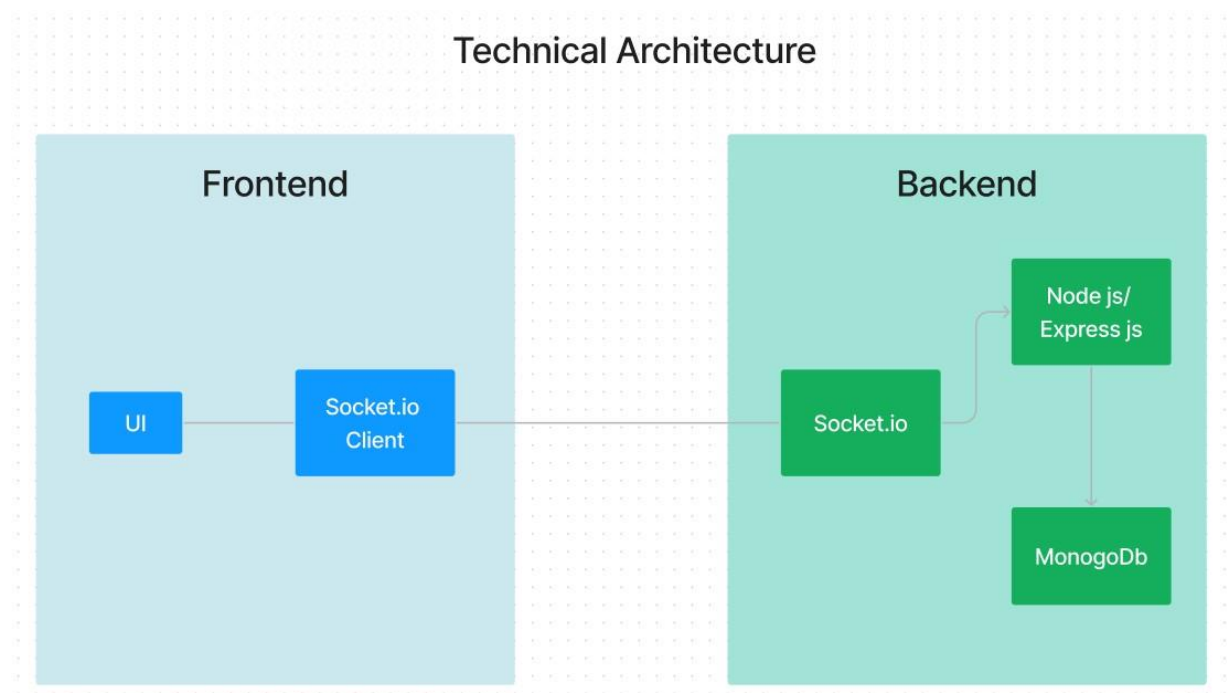
Introducing our revolutionary social media app! Designed to redefine online connections, our platform offers an intuitive and innovative space for seamless communication and engagement.

Break down barriers and enhance collaboration with real-time messaging. Whether you're sharing ideas, collaborating on projects, or simply having fun conversations, our messaging feature allows for seamless interaction and understanding.

Never miss a moment with our convenient post saving functionality. Capture important posts, articles, or inspiring content for later access and sharing with your followers. Your valuable discoveries are preserved, ensuring nothing gets lost in the vast social media landscape.

Your privacy and security are paramount. Our app employs robust encryption to safeguard your data, ensuring all communications remain confidential and protected from unauthorized access.

Step into a new era of online connections and communication. Discover the unmatched convenience of seamless messaging, in-app notifications, stories— all within our gamechanging social media app. Connect, engage, and explore more together!



The technical architecture of our social media app follows a client-server model, with a REST API used for the initial client-server connection. The frontend serves as the client and incorporates `socket.io-client` for establishing real-time communication with the backend server. The backend utilizes `socket.io` and `Express.js` frameworks to handle server-side logic and facilitate real-time messaging, post uploading, story uploading, and more.

The frontend includes the user interface and presentation layer, as well as the `socket.io-client` for establishing a persistent socket connection with the server. This enables real-time bidirectional communication, allowing for instant updates and seamless interaction between users.

Authentication is handled through the REST API, which securely verifies user credentials and provides access tokens or session cookies for subsequent requests. Once authenticated, the client establishes a socket connection with the backend to enable real-time features.

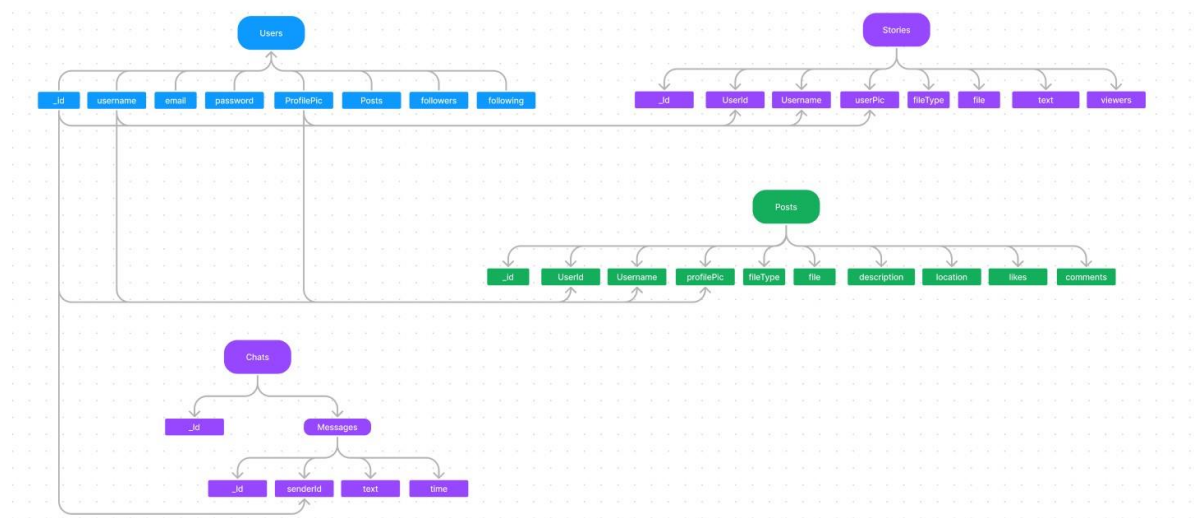
Real-time messaging is facilitated through the `socket.io` library, enabling instant exchange of messages between users. Users can engage in chats with their friends, sharing text, emojis, images, etc., in real-time.

Uploading posts and stories is also supported through the socket connection. Users can create and upload posts or stories, including text, images, videos, or a combination of media. The server receives and processes these uploads, ensuring they are associated with the correct user and made available for other users to view and interact with in real-time.

The backend utilizes `Express.js` to handle the REST API endpoints, routing requests to the appropriate controllers and services. User data, including profiles, posts, stories, and other relevant information, is stored and retrieved from a database such as `MongoDB`, ensuring efficient storage and retrieval of data.

Together, the frontend, backend, REST API, `socket.io`, `Express.js`, and database (e.g., `MongoDB`) form a comprehensive technical architecture for our social media app. This architecture enables real-time messaging, seamless post and story uploading, authentication, and secure data storage, providing users with a dynamic and interactive social media experience.

ER Diagram:



In our social media app, the ER diagram showcases entities such as users, posts, and interactions. It illustrates how these entities relate to each other, helping us understand the underlying database structure and the flow of information within the app.

The ER diagram represents the relationship between users and posts, highlighting how users can create, share, and interact with posts within the app. It also captures the relationships between users and their followers, indicating the ability for users to follow and be followed by others.

Additionally, the diagram represents interactions such as likes, comments, etc., showcasing how users can engage with posts and interact with each other's content. These interactions contribute to the overall user experience and social engagement within the app.

By visualizing the relationships between entities, the ER diagram helps us understand the overall structure of the database and the interconnectedness of different components within the social media app. It provides a valuable tool for designing and optimizing the app's functionality and data management.

Key features:

- ❑ **Real-time Updates:** Stay up to date with the latest activities and posts from your connections. Receive instant notifications for likes, comments, and mentions, ensuring you never miss out on important interactions.
- ❑ **Explore & Discover:** Explore a vast world of content and discover new ideas, trends, and communities. Engage with trending posts, discover new accounts, and connect with like-minded individuals.

- **Messaging and Chat:** Engage in private conversations and group chats with friends and followers. Share messages, emojis, photos, and videos, fostering real-time communication and connection.
- **Interactive Features:** Interact with posts through likes, comments, and shares. Express your thoughts, provide feedback, and engage in lively discussions with your network.
- **Follow and Connect:** Follow your favourite accounts and connect with influencers, brands, and individuals who inspire you. Build a vibrant network of connections and discover new opportunities.
- **Data privacy and Security:** We prioritize the protection of your personal information and data. Our app employs robust security measures, ensuring that your interactions, posts, and personal details remain secure and confidential.

These key features collectively enhance your social media experience, providing a dynamic and interactive platform for real-time communication, discovery, and connection with others.

PRE-REQUISITES:

Here are the key prerequisites for developing a full-stack application using Node.js, Express.js, MongoDB, React.js, Socket.io:

- **Node.js and npm:**

Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the server-side. It provides a scalable and efficient platform for building network applications.

Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side.

- Download: <https://nodejs.org/en/download/>
- Installation instructions: <https://nodejs.org/en/download/package-manager/>

- **Express.js:**

Express.js is a fast and minimalist web application framework for Node.js. It simplifies the process of creating robust APIs and web applications, offering features like routing, middleware support, and modular architecture. Install Express.js, a web application framework for Node.js, which handles server-side routing, middleware, and API development.

Installation: Open your command prompt or terminal and run the following command:

```
npm install express
```

□ MongoDB:

MongoDB is a flexible and scalable NoSQL database that stores data in a JSONlike format. It provides high performance, horizontal scalability, and seamless integration with Node.js, making it ideal for handling large amounts of structured and unstructured data.

Set up a MongoDB database to store your application's data.

- Download: <https://www.mongodb.com/try/download/community>
- Installation instructions: <https://docs.mongodb.com/manual/installation/>

□ React.js:

React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications.

Install React.js, a JavaScript library for building user interfaces.

Follow the installation guide: <https://reactjs.org/docs/create-a-new-react-app.html>

□ Socket.io:

Socket.io is a real-time bidirectional communication library that enables seamless communication between the server and clients. It allows for real-time data exchange, event-based messaging, and facilitates the development of real-time applications such as chat, collaboration, and gaming platforms.

Install Socket.io, a real-time bidirectional communication library for web applications.

Installation:

- Open your command prompt or terminal of server and run the following command: **npm install socket.io**
- Open your command prompt or terminal of client and run the following command: **npm install socket.io-client**

□ HTML, CSS, and JavaScript: Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

- **Database Connectivity:** Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Node.js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations. To Connect the Database with Node JS go through the below provided link:
 - <https://www.section.io/engineering-education/nodejs-mongoosejs-mongodb/>
- **Front-end Framework:** Utilize Angular to build the user-facing part of the application, including products listings, booking forms, and user interfaces for the admin dashboard.
- **Version Control:** Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.
 - Git: Download and installation instructions can be found at: <https://git-scm.com/downloads>
- **Development Environment:** Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.
 - Visual Studio Code: Download from <https://code.visualstudio.com/download>
 - Sublime Text: Download from <https://www.sublimetext.com/download>
 - WebStorm: Download from <https://www.jetbrains.com/webstorm/download>

To run the existing Video Conference App project downloaded from GitHub:

Follow below steps:

- **Clone the Repository:**
 - Open your terminal or command prompt.
 - Navigate to the directory where you want to store the e-commerce app.
 - Execute the following command to clone the repository:

```
git clone https://github.com/harsha-varadhan-reddy-07/SocialeX.git
```
- Install Dependencies:**
- Navigate into the cloned repository directory:

```
cd SocialeX
```

- Install the required dependencies by running the following commands:

```
cd client
```

```
npm install
```

```
cd ../server
```

```
npm install
```

□ **Start the Development Server:**

- To start the development server, execute the following command:

```
npm start
```

- The video conference app will be accessible at <http://localhost:3000>

□ **Access the App:**

- Open your web browser and navigate to <http://localhost:3000>.
- You should see the video conference app's homepage, indicating that the installation and setup were successful.

You have successfully installed and set up the e-commerce app on your local machine. You can now proceed with further customization, development, and testing as needed.

Roles & Responsibilities:

User:

- Create and manage a personal profile.
- Share posts, photos, videos, and stories with their network.
- Engage in conversations through comments, likes, and shares.
- Follow other users and discover new accounts, topics, and trends.
- Explore and discover new content, communities, and opportunities.
- Interact with notifications and stay updated with the activities of their connections.
- Utilize messaging and chat features to communicate with friends and followers.

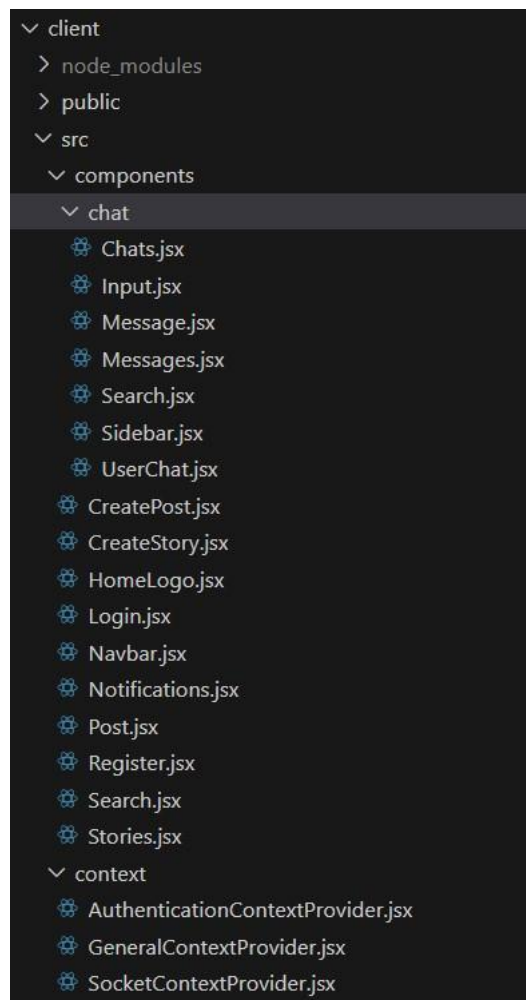
Project structure:

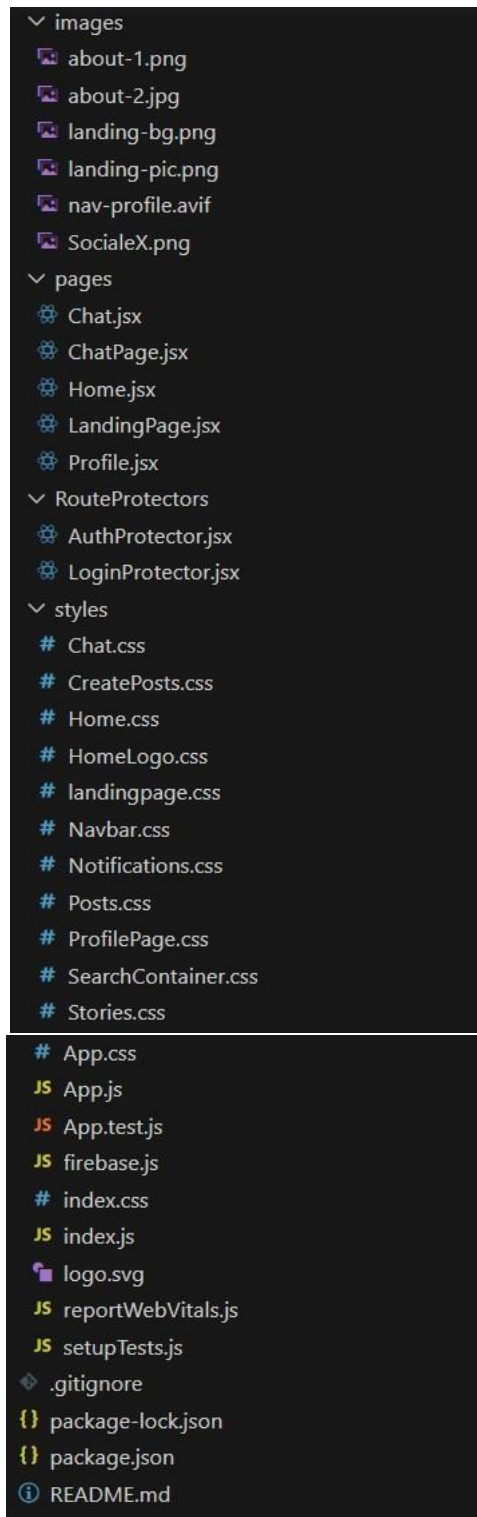
- Inside the SocialeX (social media app) directory, we have the following folders



- **Client directory:**

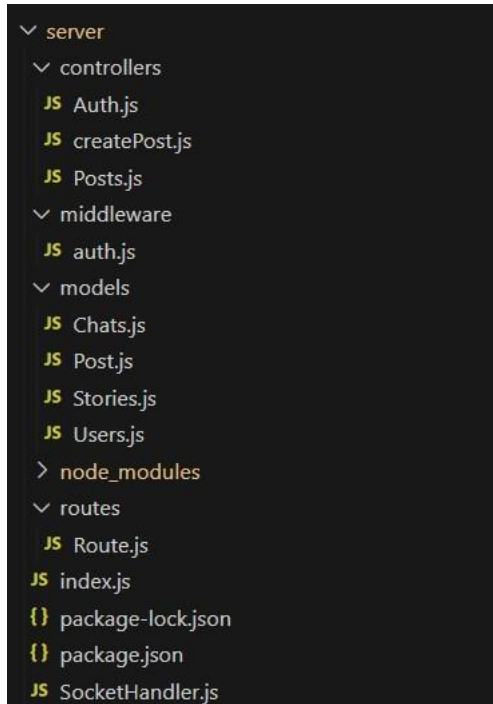
The below directory structure represents the directories and files in the client folder (front end) where, react Js is used along with Api's such as socket.io.





- **Server directory:**

The below directory structure represents the directories and files in the server folder (back end) where, node js, express js and mongodb are used along with socket.io.



Project Flow:

Project demo:

Before starting to work on this project, let's see the demo.

Demo link:

<https://drive.google.com/file/d/15JCHvQTNjOBfsxkzomK6s4YkhQBHE18o/view?usp=sharing>

Use the code in: <https://github.com/harsha-vardhan-reddy-07/SocialeX>

Milestone 1: Project setup and configuration.

- **Folder setup:**

1. Create frontend and backend folders

- ✦ client
- ✦ server

-

Installation of required tools:

Open the frontend folder to install necessary tools. For frontend (client), we use:

Tools/libraries	Installation command
React Js	<code>npx create-react-app .</code>
Socket.io-client	<code>npm install socket.io-client</code>
Bootstrap	<code>npm install bootstrap</code>
Axios	<code>npm install axios</code>
Firebase	<code>npm install firebase</code>
uuid	<code>npm install uuid</code>

Open the backend folder to install necessary tools. For backend (server), we use:

Tools/libraries	Installation command
Express Js	<code>npm install express</code>
Mongoose	<code>npm install mongoose</code>
Bcrypt	<code>npm install bcrypt</code>
Body-parser	<code>npm install body-parser</code>
Cors	<code>npm install cors</code>
Dotenv	<code>npm install dotenv</code>
Http	<code>npm install http</code>
Socket.io	<code>npm install socket.io</code>

Milestone 2: User Authentication & landing page

- **Setup express server**

1. Create index.js file in the server (backend folder).
2. Create a .env file and define port number to access it globally.
3. Configure the server by adding cors, body-parser.

- **Configure MongoDB**

-

1. Import mongoose.
2. Add Database URL to the .env file.
3. Connect the database to the server.
4. Create a 'models' folder in the server to store all the DB models.

Develop Ui (landing & login)

1. Develop the UI for landing page of the application.
2. Add the login & registration components to it or create new pages for the.
3. Collect the data from forms and send a request to backend along with that data.
4. Use axios to communicate with the server.

- **Add authentication in the server**

1. Create the "User" model for the MongoDB.
2. Create auth controller file to control the authentication actions.
3. Import "bcrypt" – used to hash(encode) the password to make it secure.
4. Define registration & login activities in the server.
5. Using Axios library, make request from the frontend.
6. Configure frontend & backend for authentication and store authenticated data in Context API in frontend.
7. On successful authentication, redirect to the home page.

Milestone 3: Web application development

- **Create socket.io connection**

1. After the successful authentication, establish a socket connection between the client and the server.
2. Use socket.io connection to update data on user events seamlessly.
3. Use socket.io in chat feature as it helps to retrieve data in real-time.

- **Add create post feature**

1. Allow the user to create a post (photo/video).

- 2. Upload the media file to firebase storage or any other cloud platform and store the data and file link in the MongoDB.
 3. Retrieve the posts and display to all the users.

- **Add profile management**

1. Add a profile page for every individual user.
2. Display the user details and posts created by the user.
3. Allow user to update details such as profile pic, username and about.

Add chat feature

1. Create an in-app chat feature.
2. Use socket.io for real-time updates.
3. Allow users to share media files in the chat.

- **Add stories/feed**

1. Stories became one of the popular features of a social media app nowadays.
2. Create the UI to display the stories.
3. Allow users to add stories.
4. Delete stories automatically when the uploaded time reaches 24hrs.
5. Display stories to the followers.

- **Add notifications**

1. Use notifications feature to notify about new followers or new chats.

Code Explanation:

1. Server setup:

Firstly, let's setup the server (backend). In the index.js file, import the required libraries and tools.

```
JS index.js M X
server > JS index.js > ...
1  import express from 'express';
2  import bodyParser from 'body-parser';
3  import mongoose from 'mongoose';
4  import cors from 'cors';
5  import { Server } from 'socket.io';
6  import http from 'http';
7  import path from 'path';
8  import { fileURLToPath } from 'url';
9
10 import authRoutes from './routes/Route.js';
11 import SocketHandler from './SocketHandler.js';
12
13
```

After importing all the libraries, setup the server with express.js and add “cors” as the middleware. Also define the socket connection for the future use. Here, the routes in the code below will be defined later. Also it’s important to connect the mongo database.

JS index.js M X

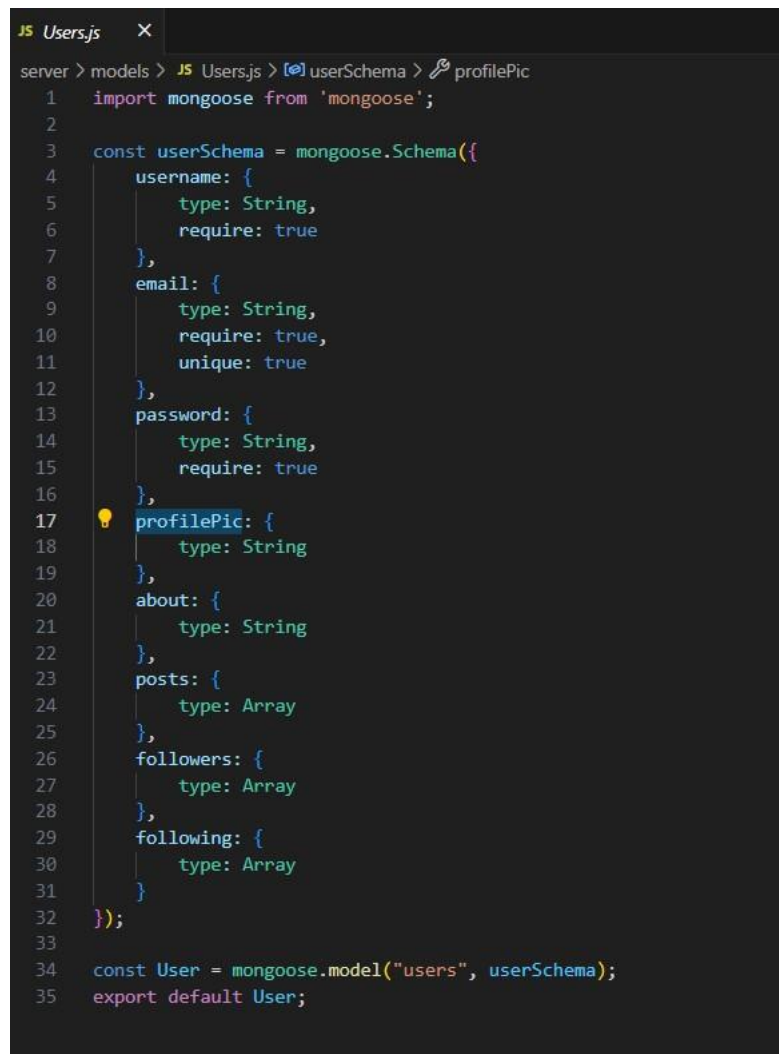
server > JS index.js > ...

```
13
14 // config
15 const __filename = fileURLToPath(import.meta.url);
16 const __dirname = path.dirname(__filename);
17
18 const app = express();
19
20 app.use(express.json());
21
22 app.use(bodyParser.json({limit: "30mb", extended: true}))
23 app.use(bodyParser.urlencoded({limit: "30mb", extended: true}));
24 app.use(cors());
25
26
27 app.use('', authRoutes);
28
29 const server = http.createServer(app);
30
31 const io = new Server(server, {
32   cors: {
33     origin: '*',
34     methods: ['GET', 'POST', 'PUT', 'DELETE']
35   }
36 });
37
38 io.on("connection", (socket) =>{
39   console.log("User connected");
40
41   SocketHandler(socket);
42 })
43
44
45 // mongoose setup
46
47 const PORT = 6001;
48
49 mongoose.connect('mongodb://localhost:27017/socialX', {
50   useNewUrlParser: true,
51   useUnifiedTopology: true,
52 })
53 ).then(()=>{
54
55   server.listen(PORT, ()=>{
56     console.log(`Running @ ${PORT}`);
57   });
58
59 }).catch((e)=> console.log(`Error in db connection ${e}`));
60
```

2. Create Database models:

Now let's define all the required models for database. Initially, let's create a models folder and add separate files for each model.

- Users model



```
JS Users.js X
server > models > JS Users.js > userSchema > profilePic
1  import mongoose from 'mongoose';
2
3  const userSchema = mongoose.Schema({
4    username: {
5      type: String,
6      require: true
7    },
8    email: {
9      type: String,
10     require: true,
11     unique: true
12   },
13   password: {
14     type: String,
15     require: true
16   },
17   profilePic: {
18     type: String
19   },
20   about: {
21     type: String
22   },
23   posts: {
24     type: Array
25   },
26   followers: {
27     type: Array
28   },
29   following: {
30     type: Array
31   }
32 });
33
34 const User = mongoose.model("users", userSchema);
35 export default User;
```

- Chats model

```
JS Chats.js X
server > models > JS Chats.js > ...
1  import mongoose from "mongoose";
2
3  const chatSchema = mongoose.Schema({
4    // _id = u1 _id + u2 _id (u1 < u2 - compare both and arrange in order)
5    _id: {
6      type: String,
7      require: true
8    },
9    messages: {
10     type: Array
11   }
12 });
13
14 const Chats = mongoose.model("chats", chatSchema);
15 export default Chats;
```

- Stories model

```
JS Stories.js X
server > models > JS Stories.js > [0] storySchema
1  import mongoose from 'mongoose';
2
3  const storySchema = new mongoose.Schema({
4    userId:{
5      type: String
6    },
7    username: {
8      type: String
9    },
10   userPic:{
11     type: String
12   },
13   fileType: {
14     type: String
15   },
16   file: {
17     type: String
18   },
19   text:{
20     type: String
21   },
22   viewers: {
23     type: Array
24   }
25 }, {timestamps: true});
26
27 const Stories = mongoose.model('stories', storySchema);
28 export default Stories;
```

- Posts model

```
JS Post.js X
server > models > JS Post.js > [e] postSchema
1  import mongoose from "mongoose";
2
3  const postSchema = mongoose.Schema({
4    userId: {
5      type: String
6    },
7    userName: {
8      type: String
9    },
10   userPic: {
11     type: String
12   },
13   fileType: {
14     type: String
15   },
16   file : {
17     type: String
18   },
19   description: {
20     type: String
21   },
22   location: {
23     type: String
24   },
25   likes: {
26     type: Array
27   },
28   comments: {
29     type: Array
30   }
31 }, {timestamps: true});
32
33 const Post = mongoose.model("posts", postSchema);
34 export default Post;
```

3. Authentication:

- Backend:

In the backend(server), let's define functions for registration and login

Register:

```
JS Authjs M X
server > controllers > JS Authjs > login
1  import bcrypt from 'bcrypt';
2  import jwt from 'jsonwebtoken';
3  import Users from '../models/Users.js';
4
5
6
7
8  const generateToken = (id) =>{
9
10     const jwtSecret = 'thisIsTheSecretCodeForTheJWTToken';
11
12     return jwt.sign({id}, jwtSecret, {
13       expiresIn: '30d',
14     })
15   }
16
17  export const register = async (req, res) =>{
18    try{
19
20      const {username, email, password, profilePic} = req.body;
21
22      const salt = await bcrypt.genSalt();
23      const passwordHash = await bcrypt.hash(password, salt);
24
25      const newUser = new Users({
26        username,
27        email,
28        password: passwordHash,
29        profilePic
30      });
31
32      const user = await newUser.save();
33
34      // generate jwt token using function we defined at top of the page
35      const token = generateToken(user._id);
36
37      const userData = {_id: user._id, username: user.username,
38                        email: user.email, profilePic: user.profilePic,
39                        about: user.about, posts: user.posts,
40                        followers: user.followers, following: user.following };
41
42      res.status(200).json({token, user: userData});
43
44    }catch(err){
45      res.status(500).json({error: err.message});
46    }
47  };
48
```

Login:

```
JS Authjs M X
server > controllers > JS Authjs > login
48
49  export const login = async (req, res) =>{
50    try{
51      const {email, password} = req.body;
52      const user = await Users.findOne({email: email});
53      if(!user) return res.status(400).json({msg: "User does not exist"});
54
55      const isMatch = await bcrypt.compare(password, user.password);
56      if(!isMatch) return res.status(400).json({msg: "Invalid credentials"});
57
58      // generate jwt token using function we defined at top of the page
59      const token = generateToken(user._id);
60      delete user.password;
61      const userData = {_id: user._id, username: user.username, email: user.email,
62                        profilePic: user.profilePic, about: user.about, posts: user.posts,
63                        followers: user.followers, following: user.following };
64
65      res.status(200).json({token, user: userData});
66      console.log(token, userData);
67    }catch(err){
68      res.status(500).json({error: err.message});
69    }
70  };
71
```


Frontend:

In the frontend, first we need to create UI for the authentication and the with the data collected in the auth for, we need to perform actions accordingly. In our case, we used separate components for login, register. Then we used context Api to store the authentication data

Register UI:

```
Register.jsx X
client > src > components > Register.jsx > [0] default
1 import React, { useContext } from 'react'
2 import { AuthenticationContext } from '../context/AuthenticationContextProvider'
3
4 const Register = ({ setIsLoginBox }) => {
5
6   const { setUsername, setEmail, setPassword, register } = useContext(AuthenticationContext);
7
8   const handleRegister = async (e) => {
9     e.preventDefault();
10
11     await register()
12   }
13
14   return (
15     <form className="authForm">
16       <h2>Register</h2>
17       <div className="form-floating mb-3 authFormInputs">
18         <input type="text" className="form-control" id="floatingInput" placeholder="username" onChange={(e) => setUsername(e.target.value)} />
19         <label htmlFor="floatingInput">Username</label>
20       </div>
21       <div className="form-floating mb-3 authFormInputs">
22         <input type="email" className="form-control" id="floatingEmail" placeholder="name@example.com" onChange={(e) => setEmail(e.target.value)} />
23         <label htmlFor="floatingInput">Email address</label>
24       </div>
25       <div className="form-floating mb-3 authFormInputs">
26         <input type="password" className="form-control" id="floatingPassword" placeholder="Password" onChange={(e) => setPassword(e.target.value)} />
27         <label htmlFor="floatingPassword">Password</label>
28       </div>
29       <button className="btn btn-primary" onClick={handleRegister}>Sign up</button>
30
31       <p>Already registered? <span onClick={() => setIsLoginBox(true)}>Login</span></p>
32     </form>
33   )
34 }
35
36 export default Register
```

Login UI:

```
Login.jsx X
client > src > components > Login.jsx > [0] Login
1
2 import React, { useContext } from 'react'
3 import { AuthenticationContext } from '../context/AuthenticationContextProvider';
4
5 const Login = ({ setIsLoginBox }) => {
6
7   const { setEmail, setPassword, login } = useContext(AuthenticationContext);
8
9   const handleLogin = async (e) => {
10     e.preventDefault();
11     await login();
12   }
13
14   return (
15     <form className="authForm">
16       <h2>Login</h2>
17       <div className="form-floating mb-3 authFormInputs">
18         <input type="email" className="form-control" id="floatingInput" placeholder="name@example.com" onChange={(e) => setEmail(e.target.value)} />
19         <label htmlFor="floatingInput">Email address</label>
20       </div>
21       <div className="form-floating mb-3 authFormInputs">
22         <input type="password" className="form-control" id="floatingPassword" placeholder="Password" onChange={(e) => setPassword(e.target.value)} />
23         <label htmlFor="floatingPassword">Password</label>
24       </div>
25       <button type="submit" className="btn btn-primary" onClick={handleLogin}>Sign in</button>
26
27       <p>Not registered? <span onClick={() => setIsLoginBox(false)}>Register</span></p>
28     </form>
29   )
30 }
31
32 export default Login
33
```

Context Api for Authentication:

```
AuthenticationContextProvider.jsx M X
client > src > context > AuthenticationContextProvider.jsx > AuthenticationContextProvider
1  import React, { createContext, useState } from 'react';
2  import axios from "axios";
3  import { useNavigate } from "react-router-dom";
4
5  export const AuthenticationContext = createContext();
6
7  const AuthenticationContextProvider = ({children}) => {
8
9      const [username, setUsername] = useState('');
10     const [email, setEmail] = useState('');
11     const [password, setPassword] = useState('');
12
13     // const profilePic = 'https://images.unsplash.com/photo-1593085512500-5d55148d6f0d?ixlib=rb-4.0
14
15     const profilePic = '';
16
17     const inputs = {username: username, email: email, password: password, profilePic: profilePic};
18
19
20     const navigate = useNavigate();
21
```

```
AuthenticationContextProvider.jsx M X
client > src > context > AuthenticationContextProvider.jsx > AuthenticationContextProvider
19
20     const navigate = useNavigate();
21
22     const login = async () =>{
23
24         try{
25
26             const loginInputs = {email: email, password: password}
27             await axios.post('http://localhost:6001/login', loginInputs)
28             .then( async (res)=>{
29                 console.log("holaaads",res);
30                 localStorage.setItem('userToken', res.data.token);
31                 localStorage.setItem('userId', res.data.user._id);
32                 localStorage.setItem('username', res.data.user.username);
33                 localStorage.setItem('email', res.data.user.email);
34                 localStorage.setItem('profilePic', res.data.user.profilePic);
35                 localStorage.setItem('posts', res.data.user.posts);
36                 localStorage.setItem('followers', res.data.user.followers);
37                 localStorage.setItem('following', res.data.user.following);
38                 navigate('/');
39             }).catch((err) =>{
40                 console.log(err);
41             });
42
43         }catch(err){
44             console.log(err);
45         }
46     }
47
48     const register = async () =>{
49
50         try{
51             await axios.post('http://localhost:6001/register', inputs)
52             .then( async (res)=>{
53                 localStorage.setItem('userToken', res.data.token);
54                 localStorage.setItem('userId', res.data.user._id);
55                 localStorage.setItem('username', res.data.user.username);
56                 localStorage.setItem('email', res.data.user.email);
57                 localStorage.setItem('profilePic', res.data.user.profilePic);
58                 localStorage.setItem('posts', res.data.user.posts);
59                 localStorage.setItem('followers', res.data.user.followers);
60                 localStorage.setItem('following', res.data.user.following);
61                 navigate('/');
62             }).catch((err) =>{
63                 console.log(err);
64             });
65
66         }catch(err){
67             console.log(err);
68         }
69     }
70
```

```
AuthenticationContextProvider.jsx M X
client > src > context > AuthenticationContextProvider.jsx > AuthenticationContextProvider
71
72
73 const logout = async () =>{
74   for (let key in localStorage) {
75     if (localStorage.hasOwnProperty(key)) {
76       localStorage.removeItem(key);
77     }
78   }
79 }
80
81 navigate('/landing');
82 }
83
84
85
86
87 return (
88   <AuthenticationContext.Provider value={{login, register, logout, username, setUsername, email, setEmail, password, setPassword }} >{children}</AuthenticationContext.Provider>
89 )
90
91 export default AuthenticationContextProvider
```

4. Set Socket client:

After successful authentication, redirect to the home page. Along with that, we establish a socket connection at client side. Now let's create a general context file to pass required info to all the children files.

General context:

```
GeneralContextProvider.jsx M X
client > src > context > GeneralContextProvider.jsx > GeneralContextProvider
1 import React, { createContext, useReducer, useState } from 'react'
2 import socketIoClient from 'socket.io-client';
3
4 export const GeneralContext = createContext();
5
6
7 const WS = 'http://localhost:6001';
8
9 const socket = socketIoClient(WS);
10
11 export const GeneralContextProvider = ({children}) => {
12
13   const [isCreatPostOpen, setIsCreatePostOpen] = useState(false);
14   const [isCreateStoryOpen, setIsCreateStoryOpen] = useState(false);
15   const [isNotificationsOpen, setNotificationsOpen] = useState(false);
16
17   const [notifications, setNotifications] = useState([]);
18
19   const [chatFirends, setChatFriends] = useState([]);
20
21   const INITIAL_STATE = {
22     chatId: 'null',
23     user: {},
24   };
25
26   const userId = localStorage.getItem('userId');
27
28   const chatReducer = (state, action) => {
29     switch (action.type) {
30       case "CHANGE_USER":
31         return {
32           user: action.payload,
33           chatId: userId > action.payload._id ? userId + action.payload._id : action.payload._id + userId
34         }
35       default:
36         return state;
37     }
38   };
39
40
41   const [state, dispatch] = useReducer(chatReducer, INITIAL_STATE);
42
43
44
45
46   return (
47     <GeneralContext.Provider value={{socket, isCreatPostOpen, setIsCreatePostOpen, isCreateStoryOpen,
48       setIsCreateStoryOpen, isNotificationsOpen, setNotificationsOpen, notifications,
49       setNotifications, chatFirends, setChatFriends, chatData:state, dispatch}}>
50       {children}
51     </GeneralContext.Provider>
52   )
53
54 }
```

5. Create Posts:

Frontend:

Here, on creating posts, we use firebase storage to store the files in the cloud. So, first we get the uploaded file URL from firebase and then update it to the MongoDB.

```
client > src > components > CreatePost.jsx > CreatePost > handlePostUpload > uploadTask.on('state_changed') callback > then() callback
1 import React, { useContext, useState } from 'react'
2 import './styles/CreatePosts.css'
3 import { RxCross2 } from 'react-icons/rx'
4 import { GeneralContext } from '../context/GeneralContextProvider'
5 import axios from 'axios'
6 import { ref, uploadBytesResumable, getDownloadURL } from 'firebase/storage'
7 import { storage } from '../firebase.js'
8 import { v4 as uuidv4 } from 'uuid'
9
10 const CreatePost = () => {
11
12   const { isCreatePostOpen, setIsCreatePostOpen } = useContext(GeneralContext);
13   const [postType, setPostType] = useState('photo');
14   const [postDescription, setPostDescription] = useState('');
15   const [postLocation, setPostLocation] = useState('');
16   const [postFile, setPostFile] = useState(null);
17   const [uploadProgress, setUploadProgress] = useState();
18
19   if (uploadProgress === 100) {
20     setPostDescription('');
21     setPostLocation('');
22     setPostFile(null);
23     setIsCreatePostOpen(false);
24     setUploadProgress();
25   }
26   const handlePostUpload = async (e) => {
27     e.preventDefault();
28     const storageRef = ref(storage, uuidv4());
29     const uploadTask = uploadBytesResumable(storageRef, postFile);
30     uploadTask.on('state_changed',
31       (snapshot) => {
32         setUploadProgress((snapshot.bytesTransferred / snapshot.totalBytes) * 100);
33       },
34       (error) => {
35         console.log(error);
36       },
37       () => {
38         getDownloadURL(uploadTask.snapshot.ref).then(async (downloadURL) => {
39           console.log('File available at', downloadURL);
40           try {
41             const inputs = {
42               userId: localStorage.getItem('userId'),
43               userName: localStorage.getItem('username'),
44               userPic: localStorage.getItem('profilePic'),
45               fileType: postType,
46               file: downloadURL,
47               description: postDescription,
48               location: postLocation,
49               comments: ["New user: " + "This is my first comment"]
50             };
51             await axios.post('http://localhost:6001/createPost', inputs)
52               .then(async (res) => {
53                 console.log(res);
54               })
55               .catch((err) => {
56                 console.log(err);
57               });
58           } catch (err) {
59             console.log(err);
60           }
61         });
62       }
63     );
64   };
65 }
```

```
client > src > components > CreatePost.jsx > CreatePost > handlePostUpload > uploadTask.on('state_changed') callback > then() callback
67
68 return (
69   <div className="createPostModalBg" style={isCreatePostOpen ? {display: 'contents'} : {display: 'none'}} >
70     <div className="createPostContainer">
71       <RxCross2 className="closeCreatePost" onClick={() => setIsCreatePostOpen(false)} />
72       <h2 className="createPostTitle">Create post</h2>
73       <hr className="createPostHr" />
74       <div className="createPostBody">
75         <form>
76           <select className="form-select" aria-label="Select Post Type" onChange={(e) => setPostType(e.target.value)} >
77             <option defaultValue="photo">Choose post type</option>
78             <option value="photo">Photo</option>
79             <option value="video">Video</option>
80           </select>
81           <div className="uploadBox">
82             <input type="file" name="PostFile" id="uploadPostFile" onChange={(e) => setPostFile(e.target.files[0])} />
83           </div>
84           <div className="form-floating mb-3 authFormInputs descriptionInput">
85             <input type="text" className="form-control descriptionInput" id="floatingDescription" placeholder="Description"
86               onChange={(e) => setPostDescription(e.target.value)} value={postDescription} />
87             <label htmlFor="floatingDescription">Description</label>
88           </div>
89           <div className="form-floating mb-3 authFormInputs postLocation">
90             <input type="text" className="form-control postLocation" id="floatingLocation" placeholder="Location"
91               onChange={(e) => setPostLocation(e.target.value)} value={postLocation} />
92             <label htmlFor="floatingLocation">Location</label>
93           </div>
94           <div>
95             <button disabled={uploadProgress ? (Math.round(uploadProgress))%100 : true} />
96             <button onClick={handlePostUpload}>Upload</button>
97           </div>
98         </form>
99       </div>
100     </div>
101   </div>
102 )
103
104 export default CreatePost
```


- **Backend:**

In the backend, we create a separate file in controllers folder and define a function to create new post.

```

JS createPost.js X
server > controllers > JS createPost.js > createPost
1 import Post from '../models/Post.js';
2
3 export const createPost = async (req, res) =>{
4   try{
5
6     const newPost = new Post(req.body);
7
8     const post = await newPost.save();
9
10  }catch(e){
11    res.status(500).json({error:e});
12  }
13 }

```

6. Display Posts:

Now we need to fetch all the posts from the database and display to users depending on the rules we define. In this case, posts will be displayed to all the users and a follow button will be displayed on the top of the post, if the user is not following the user of that post.

```

Post.jsx M X
client > src > components > Post.jsx > Post
1 import React, { useContext, useEffect, useState } from 'react';
2 import './styles/Posts.css';
3 import { AiOutlineHeart, AiTwotoneHeart } from "react-icons/ai";
4 import { FaGlobeAmericas } from "react-icons/fa";
5 import { IoIosPersonAdd } from 'react-icons/io';
6 import axios from 'axios';
7 import { GeneralContext } from '../context/GeneralContextProvider';
8 import { useNavigate } from 'react-router-dom';
9
10 const Post = () => {
11
12   const navigate = useNavigate();
13   const {socket} = useContext(GeneralContext);
14   const [posts, setPosts] = useState([]);
15
16   useEffect(() => {
17     fetchPosts();
18   }, []);
19   const fetchPosts = async () => {
20     try {
21       const response = await axios.get('http://localhost:6001/fetchAllPosts');
22       const fetchedPosts = response.data;
23       setPosts(fetchedPosts);
24     } catch (error) {
25       console.error(error);
26     }
27   };
28
29   // Like
30
31   const handleLike = (userId, postId) =>{
32     socket.emit('postLiked', {userId, postId});
33   }
34   const handleUnlike = (userId, postId) =>{
35     socket.emit('postUnLiked', {userId, postId});
36   }
37   useEffect(()=>{
38     socket.on("likeUpdated", ()=>{
39       // alert("likedd");
40     })
41     socket.on('userFollowed', ({following})=>{
42       localStorage.setItem('following', following);
43     })
44   },[socket])
45
46   const handleFollow = async (userId) =>{
47     socket.emit('followUser', {ownId: localStorage.getItem('userId'), followingUserId: userId});
48   }
49
50   const [comment, setComment] = useState('');
51   const handleComment = (postId, username) =>{
52     socket.emit('makeComment', {postId, username, comment});
53   }
54 }

```

```

Post.jsx M X
client > src > components > Post.jsx > Post > posts.map() callback
56
57   return (
58     <div className='postsContainer'>
59
60       {posts > posts.map((post) => {
61
62         return(
63
64           <div className="Post" key={post._id}>
65
66             <div className="postTop">
67               <div className="postTopDetails">
68                 <img src={post.userPic} alt="" className="userpic" />
69                 <h3 className="usernameTop" onClick={() => navigate(`/profile/${post.userId}`)}>{post.userName}</h3>
70               </div>
71
72               {localStorage.getItem('following').includes(post.userId) || localStorage.getItem('userId') === post.userId ?
73
74                 <></>
75                 :
76                 <IoPersonAdd style={{cursor: "pointer"}} id='addFriendInPost' onClick={() => handleFollow(post.userId)} />
77               }
78             </div>
79
80             { post.fileType === 'photo'?
81               <img src={post.file} className='posting' alt="" />
82               :
83               <video id="videoPlayer" className='posting' controls autoPlay muted>
84                 <source src={post.file} />
85               </video>
86             }
87
88             <div className="postReact">
89               <div className="supliconcol">
90
91                 {
92                   post.likes.includes(localStorage.getItem('userId')) ?
93                   <AiTwotoneHeart className='support reactbtn' onClick={() => handleUnlike(localStorage.getItem('userId'), post._id)} />
94                   :
95                   <AiOutlineHeart className='support reactbtn' onClick={() => handleLike(localStorage.getItem('userId'), post._id)} />
96                 }
97
98                 <label htmlFor="support" className='supportCount'>{post.likes.length}</label>
99               </div>
100               { /* <BiCommentDetail className='comment reactbtn' /> */ }
101               { /* <FiSend className='share reactbtn' onClick={() => {handleShare(post)}} /> */ }
102               <div className="placeiconcol">
103                 <FaGlobeAmericas className='placeicon reactbtn' name='place' />
104                 <label htmlFor="place" className='place'>{post.location}</label>
105               </div>
106             </div>
107           </div>
108

```

```

Post.jsx M X
client > src > components > Post.jsx > Post > posts.map() callback
108
109     <div className="detail">
110       <div className="descdataWithBtn">
111         <label htmlFor="username" className="desc labeldata" id="desc">
112           <span style={{fontWeight: 'bold'}}>
113             {post.userName}
114           </span>
115           &nbsp;    {post.description}
116         </label>
117       </div>
118     </div>
119     <div className="commentsContainer">
120       <div className="makeComment">
121         <input type="text" placeholder="type something..." onChange={(e) => setComment(e.target.value)} />
122         {comment.length === 0 ?
123           <button className="btn btn-primary" disabled>comment</button>
124           :
125           <button className="btn btn-primary" onClick={() => handleComment(post._id, localStorage.getItem('username'))}> >comment</button>
126         }
127       </div>
128       <div className="commentsBody">
129         <div className="comments">
130           {post.comments.map((comment) => {
131             return(
132               <p><b>{comment[0]}</b> {comment[1]}</p>
133             )
134           })}
135         </div>
136       </div>
137     </div>
138   </div>
139 </div>
140 )
141
142 }) : <></>
143
144 </div>
145 }
146
147 export default Post

```

7. Socket Handling in backend:

Let's create a file to handle the socket actions in the backend.

```
JS SocketHandler.js X
server > JS SocketHandler.js > [X] SocketHandler > [X] socket.on('story-played') callback
1
2 import Chats from './models/Chats.js';
3 import Post from './models/Post.js';
4 import Stories from './models/Stories.js';
5 import User from './models/Users.js';
6
7 const SocketHandler = (socket) => {
8
9   socket.on('postLiked', async ({userId, postId}) =>{
10     await Post.updateOne({_id: postId}, {$addToSet: {likes: userId}});
11     socket.emit("likeUpdated");
12   })
13
14   socket.on('postUnLiked', async ({userId, postId}) =>{
15     await Post.updateOne({_id: postId}, {$pull: {likes: userId}});
16     socket.emit("likeUpdated");
17   })
18
19   socket.on('fetch-profile', async({_id})=>{
20     const user = await User.findOne({_id})
21     console.log(user);
22     socket.emit("profile-fetched", {profile: user})
23   })
24
25
26
27   socket.on('updateProfile', async ({userId, profilePic, username, about})=>{
28     const user = await User.updateOne({_id: userId}, {profilePic: profilePic, username: username, about:about})
29     socket.emit("profile-fetched", {profile: user})
30   })
31
32   socket.on('user-search', async({username})=>{
33     const user = await User.findOne({username:username});
34     socket.emit('searched-user', {user});
35   })
36
37   socket.on('followUser', async({ownId, followingUserId})=>{
38     await User.updateOne({_id: ownId}, {$addToSet: {following: followingUserId}});
39     await User.updateOne({_id: followingUserId}, {$addToSet: {followers: ownId}});
40
41     const user1 = await User.findOne({_id: ownId});
42     const user2 = await User.findOne({_id: followingUserId});
43     socket.emit('userFollowed', {following: user1.following});
44
45     if ( user2.following.includes(user1._id) && user1.following.includes(user2._id) ){
46       const newChat = new Chats({
47         _id: user1._id > user2._id ? user1._id + user2._id : user2._id + user1._id
48       })
49       const chat = await newChat.save();
50     }
51   })
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570

```

```

JS SocketHandler.js X
server > JS SocketHandler.js > SocketHandler > socket.on('story-played') callback
86
87 socket.on('fetch-messages', async ({chatId}) =>{
88   const chat = await Chats.findOne({_id: chatId});
89
90   await socket.join(chatId);
91
92   await socket.emit('messages-updated', {chat: chat});
93
94 })
95
96 socket.on('update-messages', async ({ chatId }) => {
97   try {
98     const chat = await Chats.findOne({ _id: chatId });
99     console.log('updating messages');
100    socket.emit('messages-updated', { chat });
101  } catch (error) {
102    console.error('Error updating messages:', error);
103  }
104 });
105
106 socket.on('new-message', async ({ chatId, id, text, file, senderId, date }) => {
107   try {
108     await Chats.findOneAndUpdate(
109       { _id: chatId },
110       { $addToSet: { messages: { id, text, file, senderId, date } } },
111       { new: true }
112     );
113
114     const chat = await Chats.findOne({ _id: chatId });
115     console.log(chat);
116     socket.emit('messages-updated', { chat });
117     socket.broadcast.to(chatId).emit('message-from-user');
118   } catch (error) {
119     console.error('Error adding new message:', error);
120   }
121 });
122
123
124 socket.on('chat-user-searched', async ({ownId, username})=>{
125   const user = await User.findOne({username:username});
126   if(user){
127     if (user.followers.includes(ownId) && user.following.includes(ownId)){
128
129       socket.emit('searched-chat-user', {user});
130     }else{
131       socket.emit('no-searched-chat-user');
132     }
133   }else{
134     socket.emit('no-searched-chat-user');
135   }
136 }
137 });
138

```

```

JS SocketHandler.js X
server > JS SocketHandler.js > SocketHandler > socket.on('story-played') callback
139
140 socket.on('fetch-all-posts', async())=>{
141   const posts = await Post.find();
142   socket.emit('all-posts-fetched', {posts});
143 }
144
145
146 socket.on('delete-post', async ({postId}) =>{
147   await Post.deleteOne({_id: postId});
148   const posts = await Post.find();
149   socket.emit('post-deleted', {posts});
150 });
151
152
153 socket.on('create-new-story', async({userId, username, userPic, fileType, file, text})=>{
154   const newStory = new Stories({userId, username, userPic, fileType, file, text});
155   await newStory.save();
156 })
157
158 socket.on('fetch-stories', async())=>{
159   const stories = await Stories.find();
160   socket.emit('stories-fetched', {stories});
161 });
162
163 socket.on('story-played', async ({storyId, userId})=>{
164   await Stories.updateOne({_id: storyId}, {$addToSet: {viewers: userId}});
165 }
166 })
167 }
168
169 export default SocketHandler;

```


8. Fetch Posts:

As we used axios to fetch posts, let's define code for that. Along with fetching posts, we also included code for fetching stories.

```
JS Posts.js X
server > controllers > JS Posts.js > [0] fetchAllStories
1  import Post from '../models/Post.js';
2  import Stories from '../models/Stories.js';
3  import User from '../models/Users.js'
4
5  export const fetchAllPosts = async (req, res) =>{
6      try {
7          const posts = await Post.find().sort({ _id: -1 });
8
9          res.json(posts);
10     } catch (error) {
11         console.error(error);
12         res.status(500).json({ error: 'Server error' });
13     }
14 }
15
16 export const fetchUserName = async (req, res) =>{
17     try {
18         const userId = req.body.userId;
19         const user = await User.findById(userId);
20         console.log(userId);
21         res.status(200).json(user);
22     } catch (error) {
23         console.error(error);
24         res.status(500).json({ error: 'Server error' });
25     }
26 }
27
28 export const fetchUserImg = async (req, res) =>{
29     try {
30         const userId = req.body.userId;
31         const user = await User.findOne({ _id: userId });
32         console.log(userId);
33         res.status(200).json(user);
34     } catch (error) {
35         console.error(error);
36         res.status(500).json({ error: 'Server error' });
37     }
38 }
39
40 export const fetchAllStories = async (req, res) =>{
41     try {
42         const stories = await Stories.find();
43
44         res.status(200).json(stories);
45     } catch (error) {
46         console.error(error);
47         res.status(500).json({ error: 'Server error' });
48     }
49 }
```

9. Create Stories:

- **Frontend:**

Let's design the posts display UI and then we create a pop-up modal to create new story.

```

client > src > components > Stories.jsx > Stories
1 import React, { useContext, useEffect, useState } from 'react'
2 import '../styles/Stories.css'
3 import { BiPlusCircle } from 'react-icons/bi'
4 import { GeneralContext } from '../context/GeneralContextProvider';
5 import axios from 'axios';
6 import { RxCross2 } from 'react-icons/rx'
7
8 const Stories = () => {
9
10   const {socket, setIsCreateStoryOpen} = useContext(GeneralContext);
11
12   const [stories, setStories] = useState([])
13   const [isStoryPlaying, setIsStoryPlaying] = useState(false);
14
15   const [story, setStory] = useState();
16
17   const addStory = async () =>{
18     setIsCreateStoryOpen(true)
19   }
20
21   useEffect(() => {
22     fetchStories();
23   }, []);
24
25   const fetchStories = async () => {
26     try {
27       const response = await axios.get('http://localhost:6001/fetchAllStories');
28       setStories(response.data)
29       console.log(response.data[0])
30     } catch (error) {
31       console.error(error);
32     }
33   };
34
35   const handleOpenStory = async (story) =>{
36
37     setStory(story);
38     await socket.emit('story-played', {storyId: story._id, userId: localStorage.getItem('userId')});
39     setIsStoryPlaying(true);
40
41   }
42
43

```

```

43
44
45     return (
46       <div className="storiesContainer">
47         <div className="storiesTitle">
48           <h3>Stories</h3>
49         </div>
50
51         <div className="storiesBody" style={isStoryPlaying ? {display: 'none'} : {}}>
52
53           <div className="stories">
54
55             <div className="story self-story" onClick={addStory}>
56               <img src={localStorage.getItem('profilePic')} alt="" />
57               <p>Add story</p>
58               <span><BiPlusCircle /></span>
59             </div>
60
61             {
62               stories && stories.filter(story => ((localStorage.getItem('following')).includes(story.userId
63                                                         || story.userId == localStorage.getItem('userId'))
64                                                         && (Math.abs(Math.round((new Date().getTime() - new Date(story.createdAt).getTime()) / (1000 * 60 * 60))) < 24 )))
65               .map((story)=>{
66
67                 <div className="story user-story" key={story._id}
68                   onClick={()=> handleOpenStory(story)}
69                   style={story.viewers.includes(localStorage.getItem('userId'))
70                     ? {border: '3px solid #aa5a7e995'}
71                     : {border: '3px solid #569bdc9'}
72                   } >
73                   <img src={story.userPic} alt="" />
74                   <p>{story.username}</p>
75                 </div>
76               })
77             }
78
79           </div>
80
81         </div>
82
83       </div>
84     )
85   }
86 }
87
88 export default StoryPage
89
90
91
92
93
94
95
96
97
98
99
100

```

```

Stories.jsx M X
client > src > components > Stories.jsx > Stories
83 {story &&
84
85   <div className="storyPlayContainer" style={isStoryPlaying ? {} : {display: 'none'}}>
86     <div className="storyPlayBodyTop">
87       <p>{story.username}</p>
88       <span onClick={()=>setIsStoryPlaying(false)}><RxCross2 /></span>
89     </div>
90     <div className="storyPlayBodyContent">
91       {story.fileType === 'photo' ?
92         <img src={story.file} alt="" />
93       :
94         <video id="videoPlayer" className='postimg' controls autoPlay muted>
95           <source src={story.file} />
96         </video>
97       }
98     <p>{story.text}</p>
99   </div>
100 </div>
101 }
102
103
104
105
106 </div>
107 )
108 }
109
110 export default Stories

```

Create new story:

```

CreateStory.jsx M X
client > src > components > CreateStory.jsx > CreateStory
1 import React, { useContext, useState } from 'react';
2 import '../styles/CreatePosts.css'
3 import { GeneralContext } from '../context/GeneralContextProvider';
4 import { RxCross2 } from 'react-icons/rx';
5 import { ref, uploadBytesResumable, getDownloadURL } from "firebase/storage";
6 import { storage } from '../firebase.js';
7 import { v4 as uuidv4 } from 'uuid';
8
9 const CreateStory = () => {
10
11   const {socket, isCreateStoryOpen, setIsCreateStoryOpen} = useContext(GeneralContext);
12
13   const [storyType, setStoryType] = useState('photo');
14   const [storyDescription, setStoryDescription] = useState('');
15   const [storyFile, setStoryFile] = useState(null);
16
17   const [uploadProgress, setUploadProgress] = useState();
18
19   if (uploadProgress === 100){
20     setStoryDescription('');
21     setStoryFile(null);
22     setIsCreateStoryOpen(false);
23     setUploadProgress();
24   }
25

```

```

CreateStory.jsx M X
client > src > components > CreateStory.jsx > CreateStory > handleStoryUpload
27 const handleStoryUpload = async (e) =>{
28   e.preventDefault();
29   const storageRef = ref(storage, uuidv4());
30   const uploadTask = uploadBytesResumable(storageRef, storyFile);
31
32   uploadTask.on('state_changed',
33     (snapshot) => {
34       setUploadProgress((snapshot.bytesTransferred / snapshot.totalBytes) * 100);
35     },
36     (error) => {
37       console.log(error);
38     },
39     () => {
40       getDownloadURL(uploadTask.snapshot.ref).then( async (downloadURL) => {
41         console.log('File available at', downloadURL);
42
43         try{
44           await socket.emit('create-new-story', {userId: localStorage.getItem('userId'), username: localStorage.getItem('username'),
45             userPic: localStorage.getItem('profilePic'), fileType: storyType, file: downloadURL,
46             text: storyDescription});
47           setIsCreateStoryOpen(false);
48           setStoryDescription('');
49           setStoryFile(null);
50           setIsCreateStoryOpen(false);
51           setUploadProgress();
52         }catch(err){
53           console.log(err);
54         }
55       });
56     }
57   );

```

```

CreateStory.jsx M X
client > src > components > CreateStory.jsx > [0] CreateStory > [0] handleStoryUpload
57
58 return (
59   <div className="createPostModalBg" style={isCreateStoryOpen ? {display: 'contents'} : {display: 'none'}} >
60     <div className="createPostContainer">
61       <RxCross2 className="closeCreatePost" onClick={() => setIsCreateStoryOpen(false)} />
62       <h2 className="createPostTitle">Add new story</h2>
63       <hr className="createPostHr" />
64
65       <div className="createPostBody">
66         <form>
67           <select className="form-select" aria-label="Select Post Type" onChange={(e) => setStoryType(e.target.value)} >
68             <option defaultValue="photo">Choose post type</option>
69             <option value="photo">Photo</option>
70             <option value="video">Video</option>
71           </select>
72
73           <div className="uploadBox">
74             <input type="file" name="PostFile" id="uploadPostFile" onChange={(e) => setStoryFile(e.target.files[0])} />
75           </div>
76           <div className="form-floating mb-3 authFormInputs descriptionInput">
77             <input type="text" className="form-control descriptionInput" id="floatingDescription" placeholder="Description"
78               onChange={(e) => setStoryDescription(e.target.value)} value={storyDescription} />
79             <label htmlFor="floatingDescription">Text</label>
80           </div>
81           {uploadProgress ?
82             <button disabled>Uploading... (Math.round(uploadProgress))%</button>
83             :
84             <button onClick={handleStoryUpload}>Upload</button>
85           }
86         </form>
87       </div>
88     </div>
89   </div>
90 )
91
92 }
93
94
95 export default CreateStory

```

- **Backend:**

The backend for stories is already covered in socket handling file and posts file in server.

10. Chat feature:

The backend for the chat is already covered in socket handling. In frontend, we use multiple components. Let's go through each of them.

- **Chat page(main):**

Let's create a new file in pages folder for chat feature.

```

Chat.jsx M X
client > src > pages > Chat.jsx > ...
1 import React from 'react'
2 import '../styles/Chat.css'
3 import Navbar from '../components/Navbar'
4 import Sidebar from '../components/chat/Sidebar'
5 import UserChat from '../components/chat/UserChat'
6
7 const Chat = () => {
8   return (
9     <div className='chatPage'>
10       <div>
11         <Navbar />
12
13         <div className="home">
14
15           <Sidebar />
16           <UserChat />
17
18         </div>
19       </div>
20     </div>
21   )
22 }
23 export default Chat

```

-

Sidebar:

The sidebar in chat page displays the search component and the users list.

```
client > src > components > chat > Sidebar.jsx > ...
1  import React from 'react'
2  import Search from './Search'
3  import Chats from './Chats'
4  // import Navbar from './'
5
6  const Sidebar = () => {
7    return (
8      <div className='sidebar' >
9
10     { /* <Navbar /> */ }
11
12     <Search />
13     <Chats />
14
15     </div>
16   )
17 }
18
19 export default Sidebar
```

- **Search:**

The search feature helps to search for users to chat with them.

```

Search.jsx M X
client > src > components > chat > Search.jsx > Search > handleSelect
1 import React, { useContext, useEffect, useState } from 'react'
2 import { TbSearch } from 'react-icons/tb'
3 import { GeneralContext } from '../../context/GeneralContextProvider';
4
5 const Search = () => {
6
7   const {dispatch, socket} = useContext(GeneralContext)
8   const [search, setSearch] = useState('');
9   const userId = localStorage.getItem('userId');
10  const [user, setUser] = useState();
11  const [err, setErr] = useState(false);
12
13  const handleSearch = async (e) =>{
14    e.preventDefault();
15    setErr(false);
16    setUser();
17    await socket.emit('chat-user-searched', {ownId: userId, username: search});
18    setSearch('')
19  }
20
21  useEffect(()=>{
22    socket.on('searched-chat-user', async ({user})=>{
23      setUser(user);
24    });
25    socket.on('no-searched-chat-user', async ()=>{
26      setErr(true);
27    });
28  },[socket])
29
30  const handleSelect = async (user) =>{
31    await dispatch({type:"CHANGE_USER", payload: user});
32    setUser();
33  }
34
35  return (
36    <div className="search">
37      <div className="searchform">
38        <input type="text" placeholder="Search"
39          onChange={(e)=> {setSearch(e.target.value)}} value={search} />
40        <div className="s-icon" onClick={handleSearch}>
41          <TbSearch />
42        </div>
43      </div>
44
45      {err && <span>No User Found!!</span>}
46
47      {user && <div className="userInfo" onClick={() => handleSelect(user)} >
48        <img src={user.profilePic} alt="" />
49        <div className="userChatInfo"> <span>{user.username}</span> </div>
50      </div>
51    </div>
52  )
53 }
54 export default Search

```

Chats:

In chats component, we display the list of users available to chat.


```

Chats.jsx 3, M X
client > src > components > chat > Chats.jsx > Chats > chatFirends.map() callback
1 import React, { useContext, useEffect, useState } from 'react'
2 import { GeneralContext } from '../../context/GeneralContextProvider';
3 const Chats = () => {
4
5     const {socket, chatFirends, setChatFriends, dispatch, chatData} = useContext(GeneralContext)
6     const userId = localStorage.getItem('userId');
7
8     useEffect(()=>{
9
10         socket.emit('fetch-friends', {userId});
11
12         socket.on("friends-data-fetched", ({friendsData})=>{
13             setChatFriends(friendsData);
14         });
15     },[])
16
17     const handleSelect = (data) =>{
18         dispatch({type:"CHANGE_USER", payload: data});
19         console.log(chatData);
20     }
21     useEffect(()=>{
22
23         if(chatData.chatId !== null){
24             socket.emit('fetch-messages', {chatId: chatData.chatId})
25         }
26     }, [chatData])
27
28     return (
29         <div className='chats'>
30
31             {chatFirends.map((data)=>{
32                 return(
33                     <div className="userInfo" key={data._id} onClick={()=> handleSelect(data)} >
34                         <img src={data.profilePic} alt="" />
35                         <div className="userChatInfo">
36                             <span>{data.username}</span>
37                         </div>
38                     </div>
39                 )
40             })}
41         </div>
42     )
43 }
44 export default Chats

```

- **UserChat:**

UserChat contains components such as messages, inputs, etc., that let's users interact.

```

UserChat.jsx M X
client > src > components > chat > UserChat.jsx > UserChat
1 import React, { useContext } from 'react'
2 import Input from './Input';
3 import Messages from './Messages';
4 import { GeneralContext } from '../../context/GeneralContextProvider';
5
6 const UserChat = () => {
7
8     const {chatData} = useContext(GeneralContext);
9
10    return (
11        <div className='chat'>
12            { chatData.user &&
13
14                <div className="chatInfo">
15                    <img src={chatData.user?.profilePic} alt="" />
16                    <span>{chatData.user.username}</span>
17                </div>
18            }
19        </div>
20        <Messages />
21        <Input />
22    )
23 }
24
25 export default UserChat

```

Input:

Input component helps to type and send the message to the user at other end.

```
Input.jsx M X
client > src > components > chat > Input.jsx > Input > handleSend
1  import React, { useContext, useState } from 'react'
2  import { BiImageAdd } from 'react-icons/bi'
3  import { GeneralContext } from '../../context/GeneralContextProvider'
4  import { v4 as uuid } from 'uuid';
5  import { getDownloadURL, ref, uploadBytesResumable } from 'firebase/storage';
6  import { storage } from '../../firebase';
7
8  const Input = () => {
9
10     const {socket, chatData} = useContext(GeneralContext);
11     const [text, setText] = useState('');
12     const [file, setFile] = useState(null);
13     const [uploadProgress, setUploadProgress] = useState();
14     const userId = localStorage.getItem('userId');
15
16     const handleSend = async () =>{
17
18         if (file){
19             const storageRef = ref(storage, uuid());
20             const uploadTask = uploadBytesResumable(storageRef, file);
21             uploadTask.on('state_changed',
22                 (snapshot) => {
23                     setUploadProgress((snapshot.bytesTransferred / snapshot.totalBytes) * 100);
24                 },
25                 (error) => {
26                     console.log(error);
27                 },
28                 () => {
29                     getDownloadURL(uploadTask.snapshot.ref).then( async (downloadURL) => {
30                         console.log('File available at', downloadURL);
31
32                         try{
33                             let date = new Date()
34                             await socket.emit('new-message', {chatId: chatData.chatId ,id: uuid(),
35                                                                 text: text, file: downloadURL,
36                                                                 senderId: userId, date: date});
37                             setUploadProgress();
38                             setText('');
39                             setFile(null);
40                         }catch(err){
41                             console.log(err);
42                         }
43                     });
44                 });
45         }else{
46             let date = new Date()
47             await socket.emit('new-message', {chatId: chatData.chatId ,id: uuid(),
48                                             text: text,file: '', senderId: userId, date: date});
49             setText('');
50         }
51     }
52 }
```

```
Input.jsx M X
client > src > components > chat > Input.jsx > Input > handleSend
53
54     return (
55         <div className='input' >
56             <input type="text" placeholder='type something...' onChange={e => setText(e.target.value)} value={text} />
57             <div className="send">
58                 <input type="file" style={{display: 'none'}} id='file' onChange={e=> setFile(e.target.files[0])} />
59                 <label htmlFor="file" style={{display:'flex'}}>
60                     <BiImageAdd />
61                     <p style={{fontSize: '12px'}}><uploadProgress ? Math.floor(uploadProgress) + '%': ''></p>
62                 </label>
63                 <button onClick={handleSend} >Send</button>
64             </div>
65         </div>
66     )
67
68     export default Input
```


Messages:

The messages component is a group of all the messages in the chat. Each message will further be considered as a separate component.

```
Messages.jsx M X
client > src > components > chat > Messages.jsx > Messages
1 import React, { useContext, useEffect, useState } from 'react'
2 import Message from './Message'
3 import { GeneralContext } from '../../../context/GeneralContextProvider';
4
5 const Messages = () => {
6
7   const {socket} = useContext(GeneralContext)
8   const [messages, setMessages] = useState([]);
9   const {chatData} = useContext(GeneralContext);
10
11   useEffect(()=>{
12     const handleMessagesUpdated = ({ chat }) => {
13       console.log('chatuu', chat);
14       if (chat) {
15         setMessages(chat.messages);
16       }
17     };
18
19     const handleNewMessage = async () => {
20       console.log('new message', chatData.chatId);
21       socket.emit('update-messages', { chatId: chatData.chatId });
22     };
23
24     socket.on('messages-updated', handleMessagesUpdated);
25     socket.on('message-from-user', handleNewMessage);
26
27     return () => {
28       // Clean up event listeners when the component unmounts
29       socket.off('messages-updated', handleMessagesUpdated);
30       socket.off('message-from-user', handleNewMessage);
31     };
32   },[socket, chatData])
33
34   return (
35     <div className='messages' >
36
37       {messages.length > 0 && messages.map((message)=>(
38         <Message message={message} key={message.id} />
39       ))}
40     </div>
41   )
42 }
43
44 export default Messages
```

Message:

The message component is an individual component for each message in the chat.

```
Message.jsx M X
client > src > components > chat > Message.jsx > Message
1 import React, { useContext, useEffect, useRef } from 'react'
2 import { GeneralContext } from '../../../context/GeneralContextProvider';
3
4 const Message = ({message}) => {
5
6   const {chatData} = useContext(GeneralContext);
7   const ref = useRef();
8   let date = new Date(message.date);
9
10   useEffect(() => {
11     ref.current?.scrollIntoView({behavior: 'smooth'})
12   }, [message]);
13
14   const userId = localStorage.getItem('userId');
15   return (
16     <div>
17       <div ref={ref} className='message ${message.senderId === userId ? "owner" : ""}'>
18         <div className='messageInfo'>
19           <img src={message.senderId === userId ? localStorage.getItem('profilePic') : chatData.user-profilePic} alt="" />
20           <span>{date.getHours() < 12 ? date.getHours() + ':' + date.getMinutes() + ' AM' : date.getHours()-12 + ':' + date.getMinutes() + ' PM'}</span>
21         </div>
22         <div className='messageContent'>
23           <p>{message.text}</p>
24           {message.file && <img src={message.file} alt="" />}
25         </div>
26       </div>
27     </div>
28   )
29 }
30
31 export default Message
```

11. Navbar:

Now let's look into the navbar component.

```
client > src > components > Navbar.jsx > Navbar
8 import { GeneralContext } from '../context/GeneralContextProvider';
9 import { useNavigate } from 'react-router-dom';
10
11 const Navbar = () => {
12
13   const { isCreatPostOpen, setIsCreatPostOpen, setIsCreateStoryOpen, isNotificationsOpen, setNotificationsOpen } = useContext(GeneralContext);
14
15   const navigate = useNavigate();
16   const profilePic = localStorage.getItem('profilePic');
17   const userId = localStorage.getItem('userId');
18
19   return (
20     <div className="Navbar">
21       <BiHomeAlt className="homebtn btns" onClick={()=> navigate('/')}/>
22       <BsChatSquareText className="chatbtn btns" onClick={()=> navigate('/chat')}/>
23       <GAddR className="createPostbtn btns" onClick={()=> {setIsCreatPostOpen(!isCreatPostOpen); setIsCreateStoryOpen(false)}}/>
24       <TbNotification className="Motifybtn btns" onClick={()=> setNotificationsOpen(!isNotificationsOpen)}/>
25       <img className="profile" src={profilePic} alt="" onClick={()=> navigate(`/profile/${userId}`)}/>
26     </div>
27   );
28 }
29 export default Navbar
```

12. Logo & User Search:

The logo and search components are implemented together. A separate Search component is created to make it better understandable.

```
client > src > components > HomeLogo.jsx > HomeLogo
1 import React, { useContext, useEffect, useState } from 'react';
2 import logoimg from '../images/SocialeX.png';
3 import '../styles/HomeLogo.css';
4 import { TbSearch } from 'react-icons/tb';
5 import { GeneralContext } from '../context/GeneralContextProvider';
6 import Search from './Search';
7
8 const HomeLogo = () => {
9
10   const { socket } = useContext(GeneralContext);
11   const [search, setSearch] = useState('');
12   const [searchedUser, setSearchedUser] = useState();
13
14   const handleSearch = async () => {
15     await socket.emit('user-search', { username: search });
16     setSearch('');
17   }
18
19   useEffect(() => {
20     socket.on('searched-user', (user) => {
21       setSearchedUser(user);
22     });
23   }, [socket]);
24
25   return (
26     <div className="LogoSearch">
27       <img className="logoimg" src={logoimg} alt="" />
28       <div className="Search">
29         <input type="text" placeholder="Search" onChange={(e) => {setSearch(e.target.value)}} value={search} />
30         <div className="s-icon" onClick={handleSearch}>
31           <TbSearch />
32         </div>
33       </div>
34       <Search searchedUser={searchedUser} setSearchedUser={setSearchedUser} />
35     </div>
36   );
37 }
38 export default HomeLogo
```

Search Component:

```

Search.jsx M X
client > src > components > Search.jsx > [0] Search
1 import React from 'react'
2 import '../styles/SearchContainer.css';
3 import { useNavigate } from 'react-router-dom';
4
5 const Search = ({searchedUser, setSearchUser}) => {
6   const navigate = useNavigate();
7   return (
8     <div className="searchContainer">
9
10      {searchedUser && <div className="searchedUserInfo" onClick={()=> {navigate(`/profile/${searchedUser._id}`); setSearchUser({});}} >
11        <img src={searchedUser.profilePic} alt="" />
12        <div className="searchedUserChatInfo">
13          <span>{searchedUser.username}</span>
14        </div>
15      </div>
16    </div>
17  )
18 }
19
20
21 export default Search

```

13. Routes in Backend:

As we used a separate routing file for routing at server side, let's implement it.

```

JS Route.js X
server > routes > JS Route.js > [0] default
1 import express from 'express';
2 import { login, register } from '../controllers/Auth.js';
3 import { createPost } from '../controllers/createPost.js';
4 import { fetchAllPosts, fetchAllStories, fetchUserImg, fetchUserName } from '../controllers/Posts.js';
5
6 const router = express.Router();
7
8 router.post('/register', register);
9 router.post('/login', login);
10 router.post('/createPost', createPost);
11 router.get('/fetchAllPosts', fetchAllPosts);
12 router.get('/fetchUserName', fetchUserName);
13 router.get('/fetchUserImg', fetchUserImg);
14 router.get('/fetchAllStories', fetchAllStories);
15
16 export default router;

```

Finally, for any further assistance, use the links below:

Demo link:

<https://drive.google.com/file/d/15JCHvQTNjOBfsxkzomK6s4YkhQBHE18o/view?usp=sharing>

Use the code in: <https://github.com/harsha-varadhan-reddy-07/SocialeX>

*** Happy coding!! ***