Social Media App (MERN)

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 game-changing social media app. Connect, engage, and explore more together!

Description

App Overview: Revolutionizing Social Connections

Welcome to our next-generation social media application – a dynamic platform designed to enhance the way people connect, communicate, and collaborate. With a clean, modern interface and an array of innovative features, our app empowers users to interact meaningfully in both personal and professional spaces.

Core Features

- ✓ **Seamless Real-Time Messaging:** Engage in smooth, responsive conversations. Our real-time chat feature supports everything from quick check-ins to deep collaboration, ensuring communication is fast and intuitive.

- ✓ **Post Saving & Content Preservation:** Never lose valuable content again. Users can easily save important posts, articles, or media for later review and sharing, ensuring that inspiration and critical insights are always within reach.

- ✓ **Privacy-First Architecture:** Security is not optional—it's foundational. We implement end-to-end encryption and secure data handling practices to protect all user communications and personal information from unauthorized access.

- ✓ **Interactive Stories & Notifications:** Stay up to date with interactive stories and in-app notifications. Whether it's a status update or an important message, our alert system ensures you're always in the loop.

Why Choose Us?

- **User-Centric Design:** Intuitive navigation and responsive UI make interactions effortless.
- **Collaborative by Nature:** Designed for both casual and work-related communication.
- **Secure & Reliable:** Built with industry-standard security protocols to maintain trust and integrity.

Use This App To:

- Build genuine online communities
- Collaborate in real-time on ideas or projects
- Discover and archive meaningful content
- Communicate securely in a protected digital environment
- Step into a new era of social interaction—where privacy, creativity, and communication meet. Whether you're an individual user or a community builder, our app is your gateway to smarter, safer, and more engaging digital connections.

Scenario based case-study

Imagine you're a student working on a group project. You can use the app to:

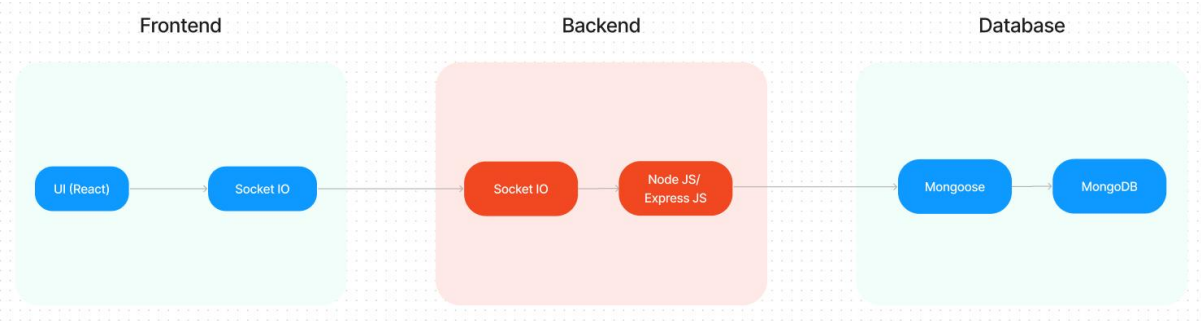
- Break down barriers and enhance collaboration: Discuss ideas and project details seamlessly through real-time messaging. Brainstorm, share files, and delegate tasks all within the app.

- Never miss a moment: Find a fascinating research article online? Save it with the post saving functionality to easily reference it later and share it with your group for a collaborative understanding.

- Your privacy and security are paramount: No need to worry about information leaks. The app's robust encryption ensures your discussions and shared files stay confidential.

- This is just one example! With its comprehensive features, the app empowers you to connect, engage, and explore effectively in any online interaction.

TECHNICAL ARCHITECTURE



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.

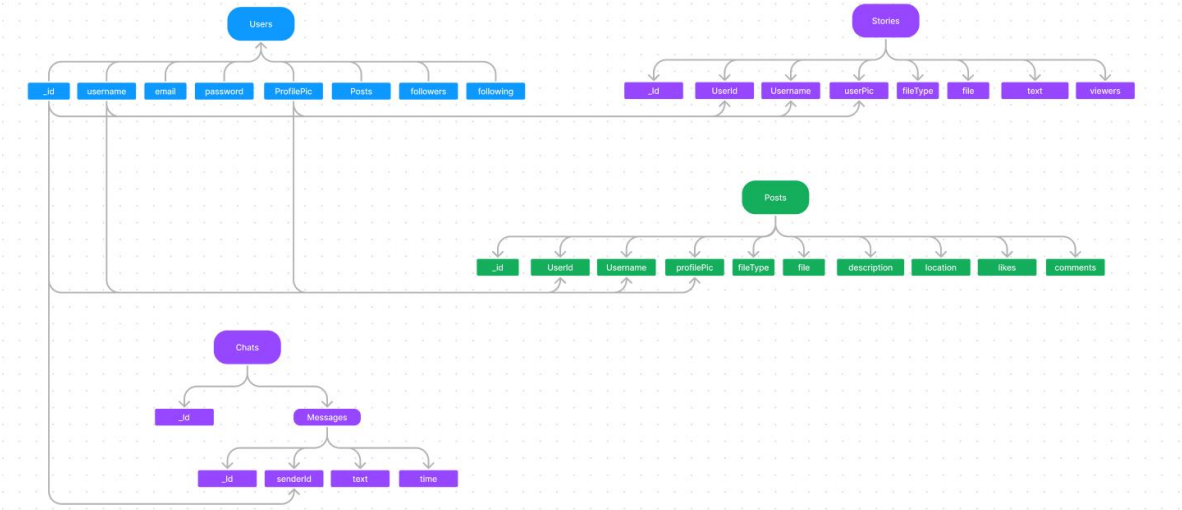
Real-time messaging is facilitated through the socket.io library, enabling instant exchange of messages between users. Users can engage in chats with there friends, sharing text, emojis, images, etc., in real-time.

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



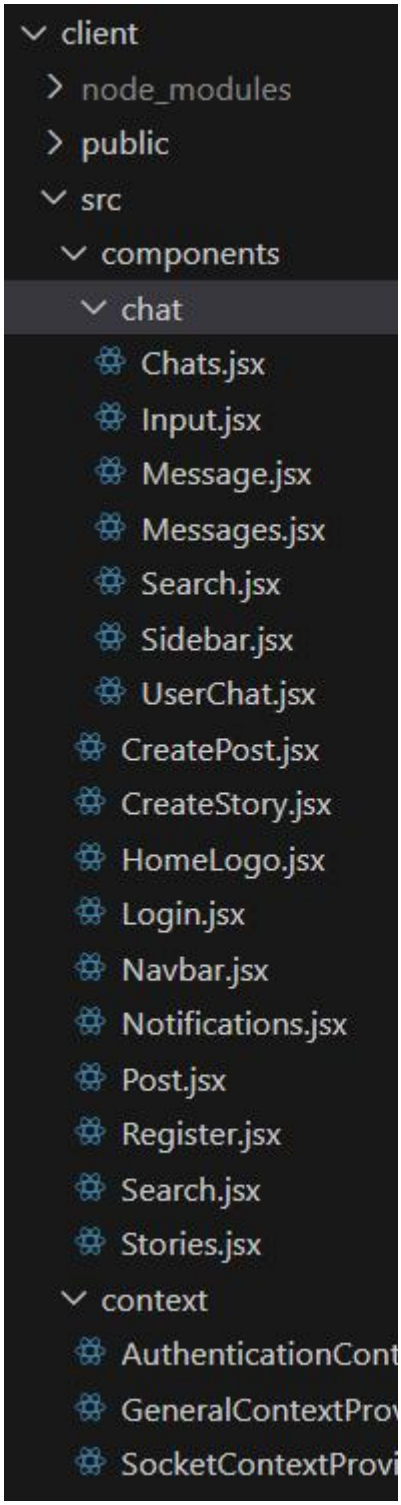
In our social media app, the ER diagram showcases entities such as users, posts, and interactions. It illustrates how these entities relate to each other, helping us understand the underlying database structure and the flow of information within the app.

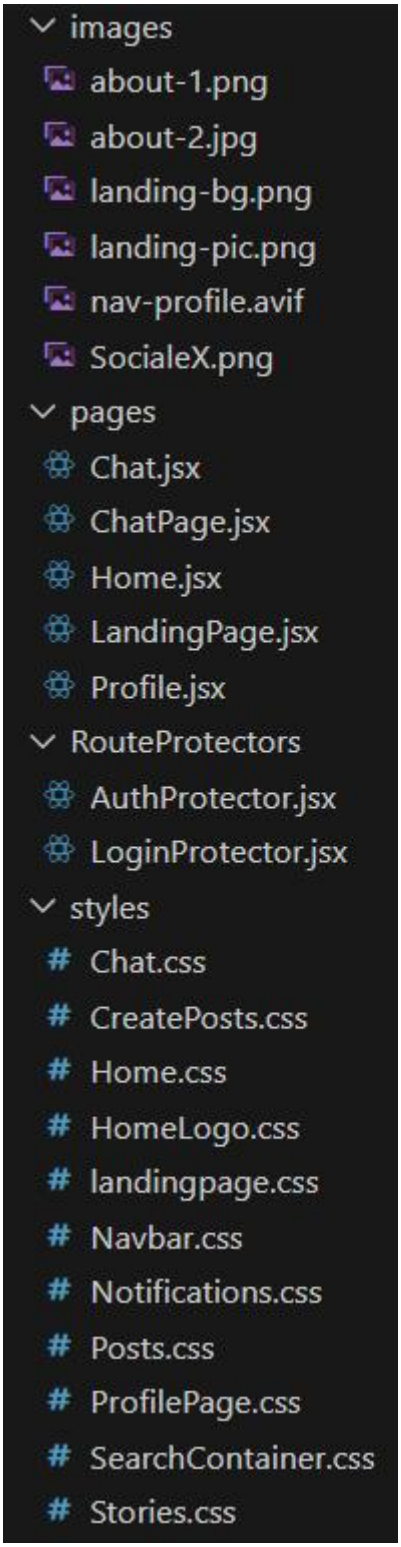
The ER diagram represents the relationship between users and posts, highlighting how users can create, share, and interact with posts within the app. It also captures the relationships between users and their followers, indicating the ability for users to follow and be followed by others.

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.

PROJECT STRUCTURE


client
node_modules
public
src
components
chat
Chats.jsx
Input.jsx
Message.jsx
Messages.jsx
Search.jsx
Sidebar.jsx
UserChat.jsx
CreatePost.jsx
CreateStory.jsx
HomeLogo.jsx
Login.jsx
Navbar.jsx
Notifications.jsx
Post.jsx
Register.jsx
Search.jsx
Stories.jsx
context
AuthenticationCont
GeneralContextProv
SocketContextProvi


images
about-1.png
about-2.jpg
landing-bg.png
landing-pic.png
nav-profile.avif
SocialeX.png
pages
Chat.jsx
ChatPage.jsx
Home.jsx
LandingPage.jsx
Profile.jsx
RouteProtectors
AuthProtector.jsx
LoginProtector.jsx
styles
Chat.css
CreatePosts.css
Home.css
HomeLogo.css
landingpage.css
Navbar.css
Notifications.css
Posts.css
ProfilePage.css
SearchContainer.css
Stories.css


App.css
App.js
App.test.js
firebase.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
.gitignore
package-lock.json
package.json
README.md


server
controllers
Auth.js
createPost.js
Posts.js
middleware
auth.js
models
Chats.js
Post.js
Stories.js
Users.js
node_modules
routes
Route.js
index.js
package-lock.json
package.json
SocketHandler.js

PRE-REQUISTIC:

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 JSON-like 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 Social media App project downloaded from google drive:

Firstly, download the code from the google drive link provided below

https://drive.google.com/drive/folders/1y9OI8R3OHP7KwYv-c91JNrD-ZQ--ZqC6?usp=sharing

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 social media app on your local machine. You can now proceed with further customization, development, and testing as needed.

Application flow:

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 Flow:

Use the code in**:** https://drive.google.com/drive/folders/10mSn2lMTaVMDWWFNjeJjiOLfmcD3-87C?usp=sharing

Demo video:
https://drive.google.com/file/d/1erdcudF8D00QyHEf0aMKioTAqWa2AjDb/view?usp=sharing

Milestone 1: Project setup and configuration.

Folder setup:

Now, firstly create the folders for frontend and backend to write the respective code and install the essential libraries.

- Client folders.
- Server folders

Installation of required tools:

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

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

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

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

Milestone 2: Backend Development

- **Set Up Project Structure:**
    - Create a new directory for your project and set up a package.json file using npm init command.
    - Install necessary dependencies such as Express.js, Mongoose, and other required packages.

- **Set Up Project Structure:**

    - Create a new directory for your project and set up a package.json file using npm init command.

    - Install necessary dependencies such as Express.js, Mongoose, and other required packages.

Create Express.js Server:

- Set up an Express.js server to handle HTTP requests and serve API endpoints.

- Configure middleware such as body-parser for parsing request bodies and cors for handling cross-origin requests.

- **Define API Routes:**

    - Create separate route files for different API functionalities such as authentication, create post, upload stories, chats, etc.,

    - Implement route handlers using Express.js to handle requests and interact with the database.

- **Implement Data Models:**

    - Define Mongoose schemas for the different data entities like posts, chats, users, etc.,

    - Create corresponding Mongoose models to interact with the MongoDB database.

    - Implement CRUD operations (Create, Read, Update, Delete) for each model to perform database operations.

- **User Authentication:**
    - Implement user authentication using strategies like JSON Web Tokens (JWT) or session-based authentication.
    - Create routes and middleware for user registration, login, and logout.
    - Set up authentication middleware to protect routes that require user authentication.
- **Handle new chats and posts:**
    - Allow users to chat with other users using the userId.
    - Also the users will make the posts on social timeline.
- **Error Handling:**
    - Implement error handling middleware to catch and handle any errors that occur during the API requests.
    - Return appropriate error responses with relevant error messages and HTTP status codes.

Reference video for backend code:

https://drive.google.com/file/d/19J2FH9sfmXMEcgdLhxZQjgoXPMPnU3BB/view?usp=sharing

Milestone 3: Database development

- Set up a MongoDB database either locally or using a cloud-based MongoDB service like MongoDB Atlas.
- Create a database and define the necessary collections for users, posts, stories, chats, etc.,

The code for the database connection to the server is

```
// mongoose setup

const PORT = 6001;

mongoose.connect('mongodb://localhost:27017/socialeX', {
        useNewUrlParser: true,
        useUnifiedTopology: true,
    }
).then(()=>{

        server.listen(PORT, ()=>{
            console.log(`Running @ ${PORT}`);
        });
    }
).catch((e)=> console.log(`Error in db connection ${e}`));
```

The code for the schemas in the database is provided below

Chats.js M

server > models > Chats.js > chatSchema

```
1   import mongoose from "mongoose";
2
3   const chatSchema = mongoose.Schema({
4       _id: {
5           type: String,
6           require: true
7       },
8       messages: {
9           type: Array
10      }
11  });
12
13  const Chats = mongoose.model("chats", chatSchema);
14  export default Chats;
```

Post.js

server > models > Post.js > default

```
import mongoose from "mongoose";

const postSchema  = mongoose.Schema({
    userId: {
        type: String
    },
    userName:{
        type: String
    },
    userPic:{
        type: String
    },
    fileType: {
        type: String
    },
    file : {
        type: String
    },
    description: {
        type: String
    },
    location: {
        type: String
    },
    likes: {
        type: Array
    },
    comments: {
        type: Array
    }
}, {timestamps: true});

const Post = mongoose.model("posts", postSchema);
export default Post;
```

Stories.js

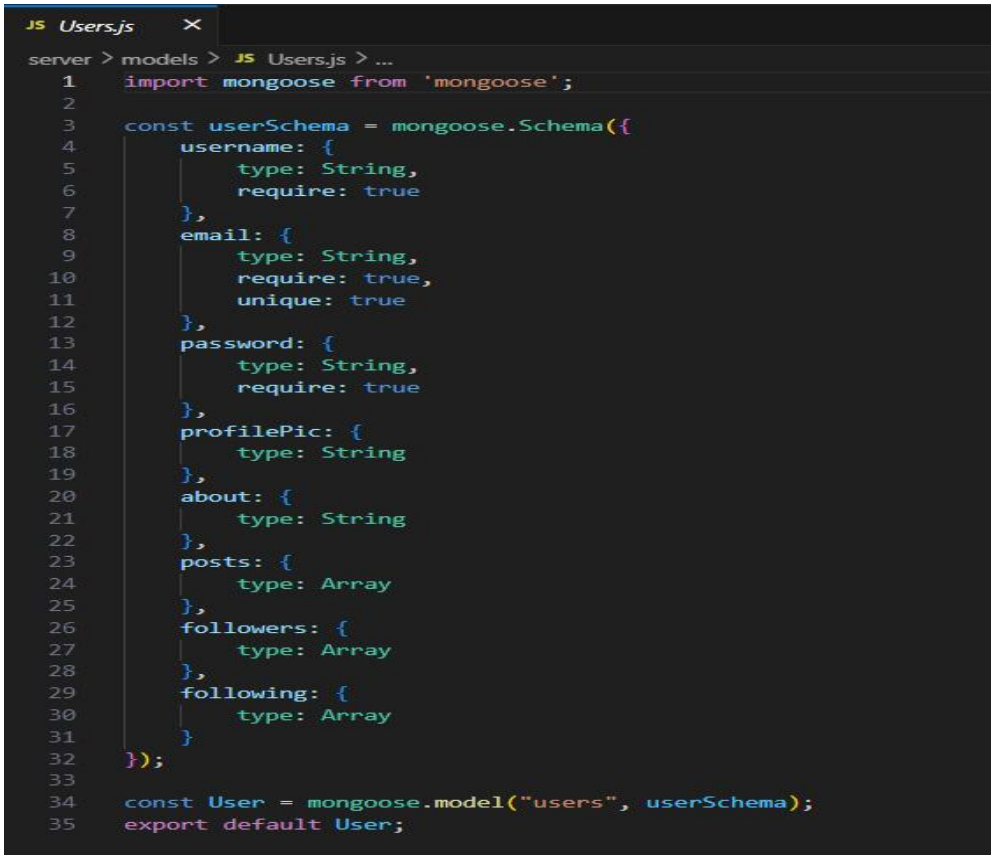
server > models > Stories.js > ...

```
import mongoose from 'mongoose';

const storySchema = new mongoose.Schema({
    userId:{
        type: String
    },
    username: {
        type: String
    },
    userPic:{
        type: String
    },
    fileType: {
        type: String
    },
    file: {
        type: String
    },
    text:{
        type: String
    },
    viewers: {
        type: Array
    }

}, {timestamps: true});

const Stories = mongoose.model('stories', storySchema);
export default Stories;
```



Milestone 4: Frontend 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.

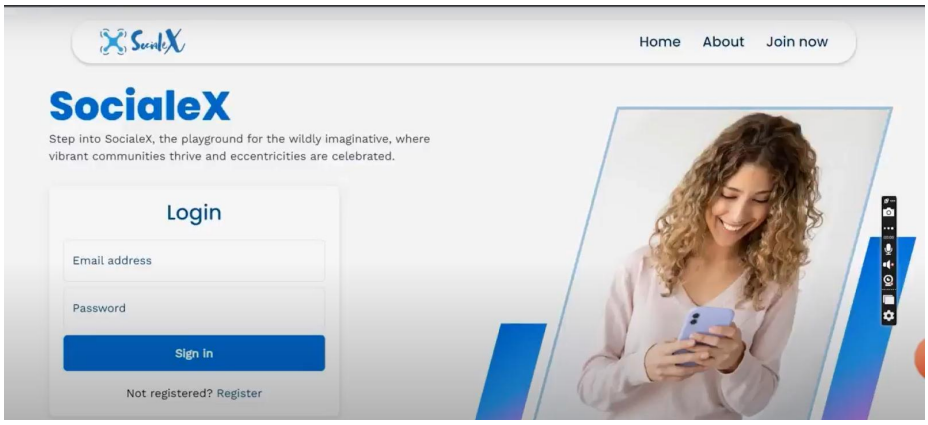
Reference for frontend code:

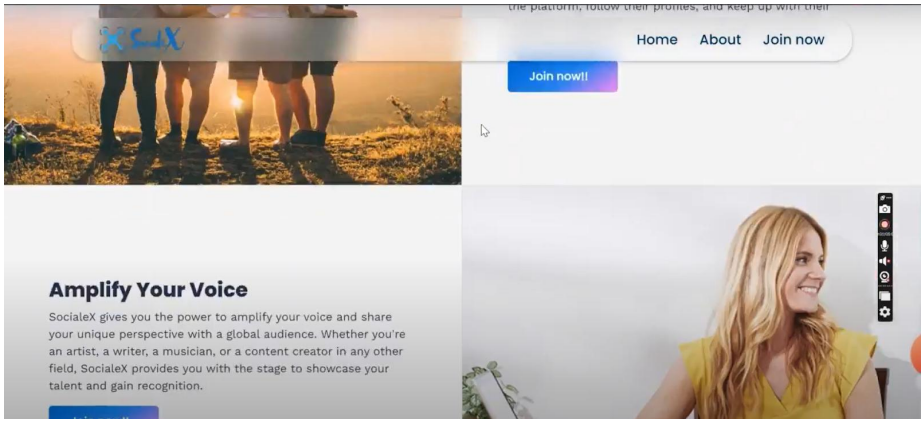
https://drive.google.com/file/d/1BnE4Ou0UneKwBx6DgaDYCmitem5wPqxc/view?usp=sharing

Milestone 5: Project Implementation

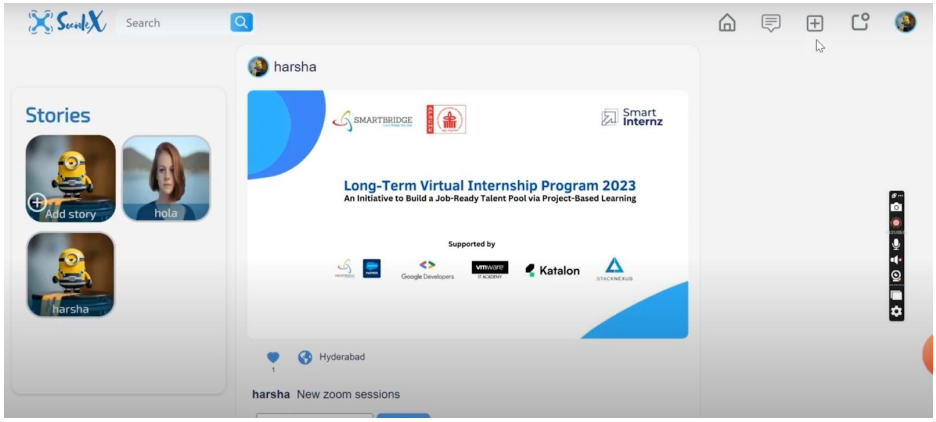
On completing the development part, we then run the application one last time to verify all the functionalities and look for any bugs in it. The user interface of the application looks a bit like the one's provided below.

- **Landing page**

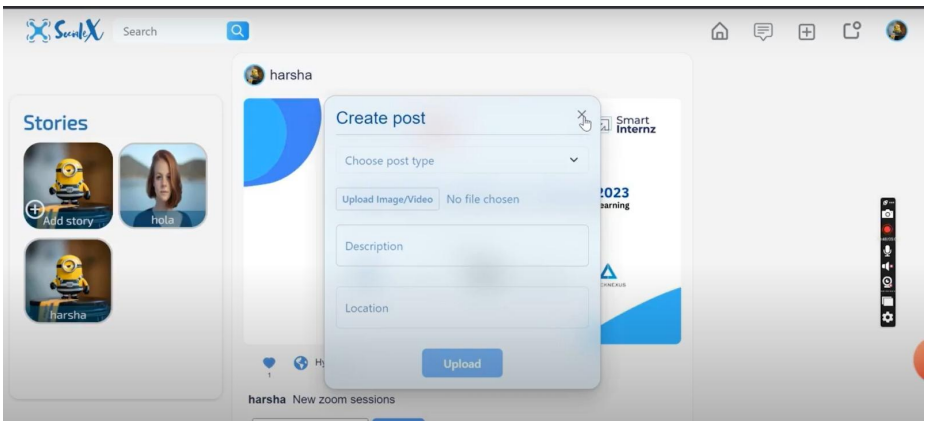




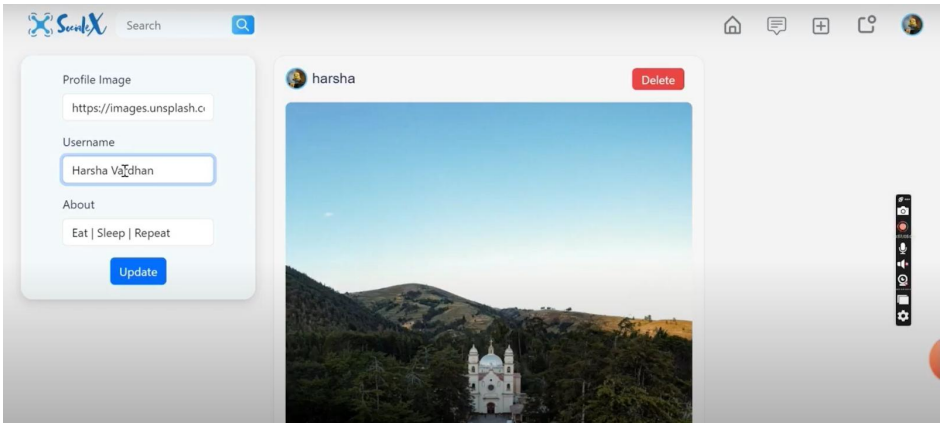
- **Home page**



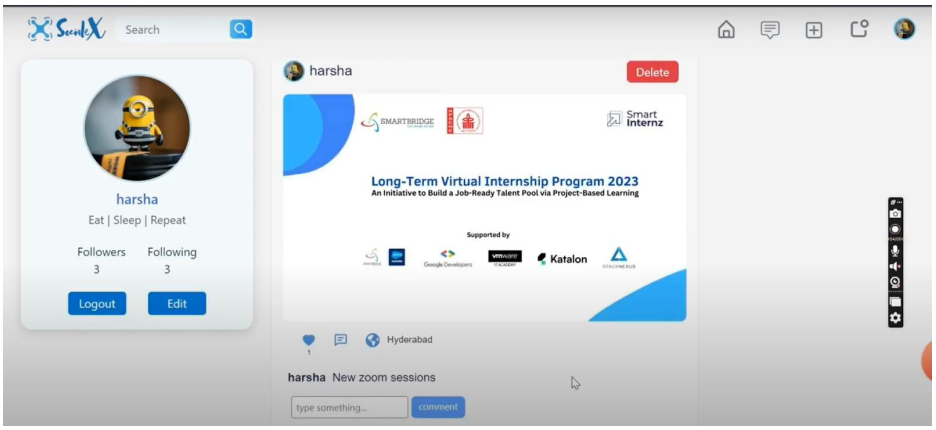
- **Create posts**



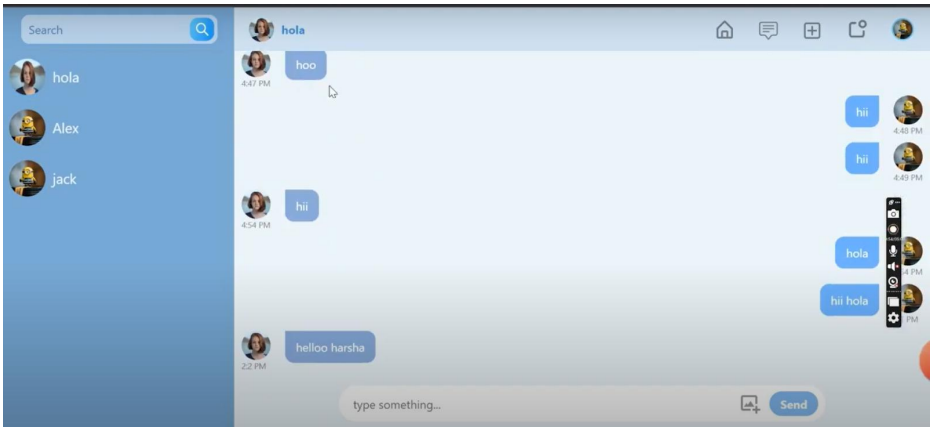
- **Update profile**



- **Profile**



- **Chats**



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://drive.google.com/drive/folders/1y9OI8R3OHP7KwYv-c91JNrD-ZQ--ZqC6?usp=sharing