Social Media App (MERN) Documentation

Introduction

Tired of outdated social media? Introducing a revolutionary app that redefines connections. Seamless messaging, saved discoveries, and top-notch security - all in one place! Get ready to connect and explore like never before.

Description

Our revolutionary social media app is designed to redefine online connections, offering an intuitive and innovative space for seamless communication and engagement. Key features include:

- Real-time messaging
- Post saving functionality
- Top-notch security with robust encryption
- Stories and in-app notifications

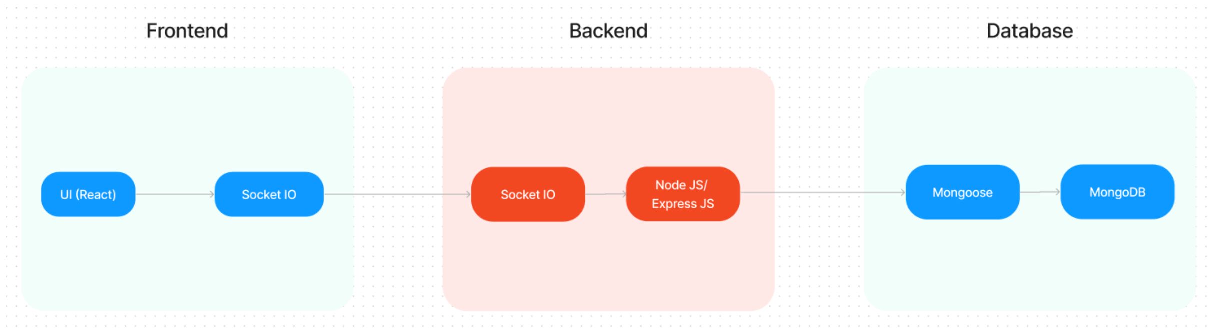
Step into a new era of online connections and communication. Connect, engage, and explore more together!

Scenario-Based Use Case

Imagine you're a student working on a group project:

- **Collaboration:** Discuss ideas through real-time messaging.
- **Save Important Content:** Bookmark research articles using the post-saving feature.
- **Secure Communication:** Encrypted messaging ensures your files and conversations remain private.

Technical Architecture



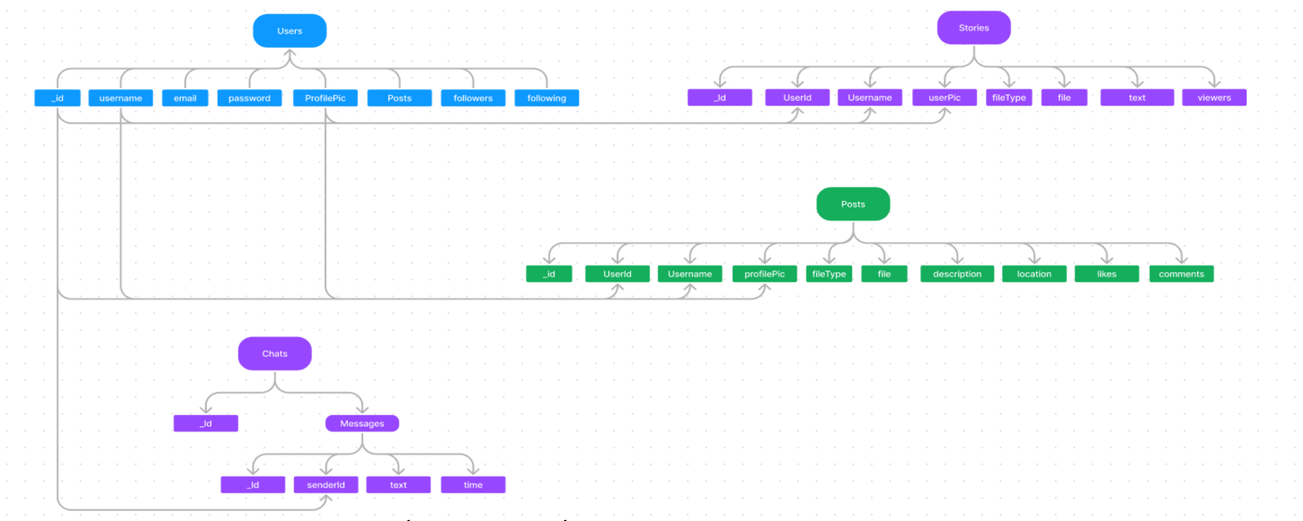
The technical architecture follows the **Client-Server Model**:

- **Frontend:** React.js with socket.io-client for real-time communication.
- **Backend:** Node.js with Express.js for server-side logic and Socket.io for real-time messaging.
- **Database:** MongoDB for data storage.
- **Authentication:** REST API with JWT tokens for secure access.
- **File Uploads:** Post and story uploads with media support.

Flow

1. User Authentication via REST API.
2. Persistent Socket Connection for real-time messaging.
3. CRUD Operations on Posts and Stories.
4. Secure Data Storage with MongoDB.

ER Diagram



Key Features

- Real-time Updates
- Explore & Discover
- Messaging and Chat
- Interactive Features (Likes, Comments, Shares)
- Follow and Connect
- Data Privacy and Security

Pre-requisites

Technologies Required

- Node.js & npm
- Express.js
- MongoDB
- React.js
- Socket.io
- HTML, CSS, and JavaScript
- Git for Version Control

Setup Instructions

1. **Node.js & npm Installation:** nodejs.org/en/download/
2. **MongoDB Installation:** mongodb.com/try/download/community
3. **React.js Setup:** reactjs.org/docs/create-a-new-react-app.html
4. **Socket.io Installation:**

```
npm install socket.io socket.io-client
```

5. **IDE Recommendation:** Visual Studio Code

Installation Guide

1. Navigate to the project directory:
   `cd SocialeX`
3. Install Dependencies:

```
cd client
npm install
cd ../server
npm install
```

5. Start the Development Server:

```
npm start
```

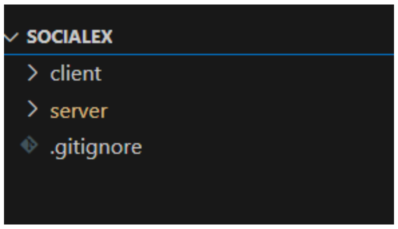
7. Access the app at http://localhost:3000

Conclusion

With its comprehensive features and robust technical architecture, this social media app empowers users to connect, engage, and explore like never before. Enjoy seamless messaging, enhanced security, and real-time updates!

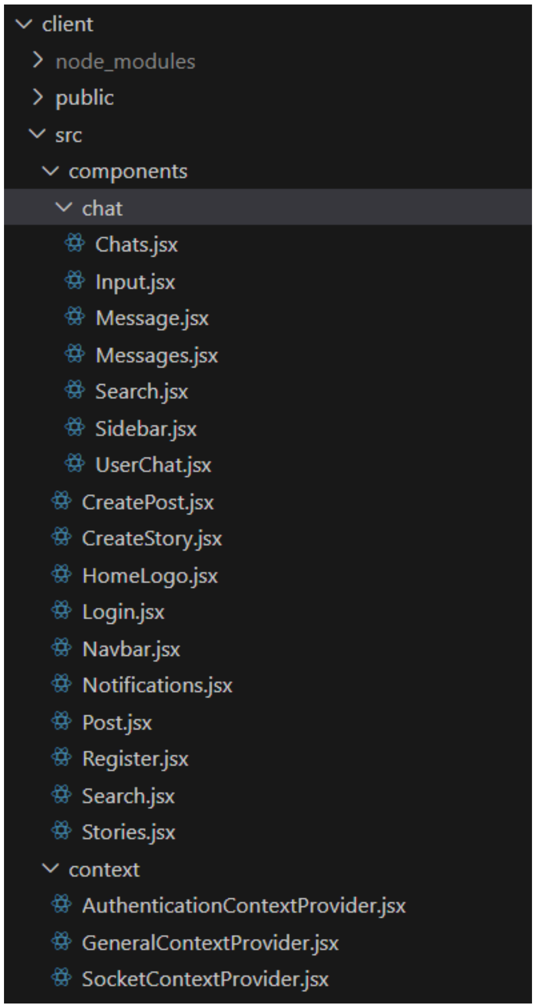
Project structure:

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

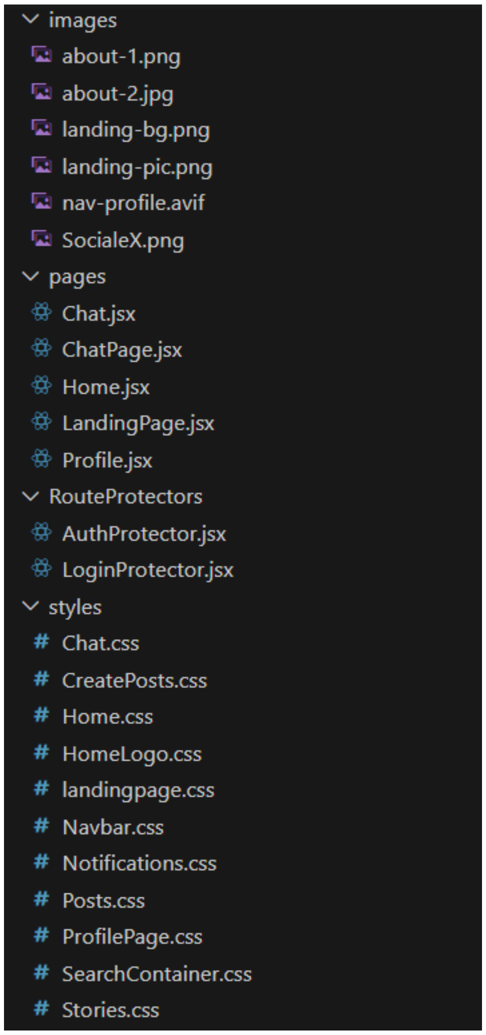


- **Client directory:**

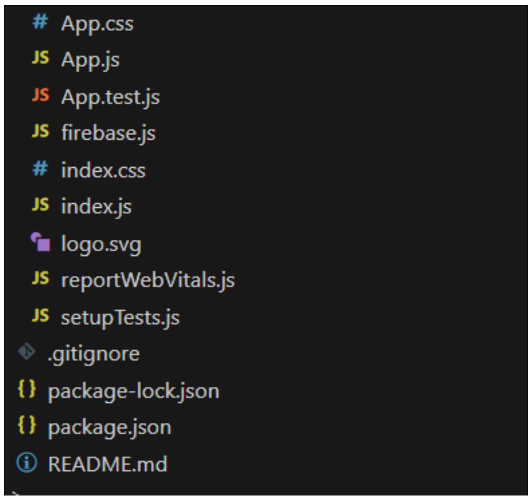
  The below directory structure represents the directories and files in the client folder (front end) where, react Js is used along with Api's such as socket.io.



```
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
      AuthenticationContextProvider.jsx
      GeneralContextProvider.jsx
      SocketContextProvider.jsx
```



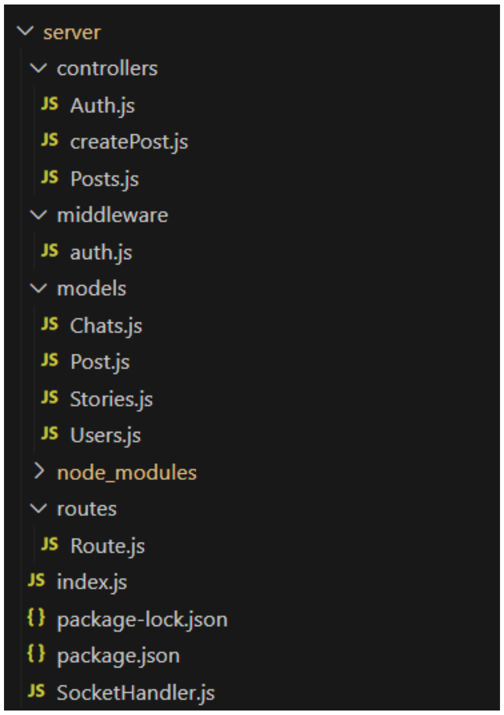
```
images
  about-1.png
  about-2.jpg
  landing-bg.png
  landing-pic.png
  nav-profile.avif
  SocialeX.png
pages
  Chat.jsx
  ChatPage.jsx
  Home.jsx
  LandingPage.jsx
  Profile.jsx
RouteProtectors
  AuthProtector.jsx
  LoginProtector.jsx
styles
  Chat.css
  CreatePosts.css
  Home.css
  HomeLogo.css
  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 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.



```
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
```

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:

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:

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

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:

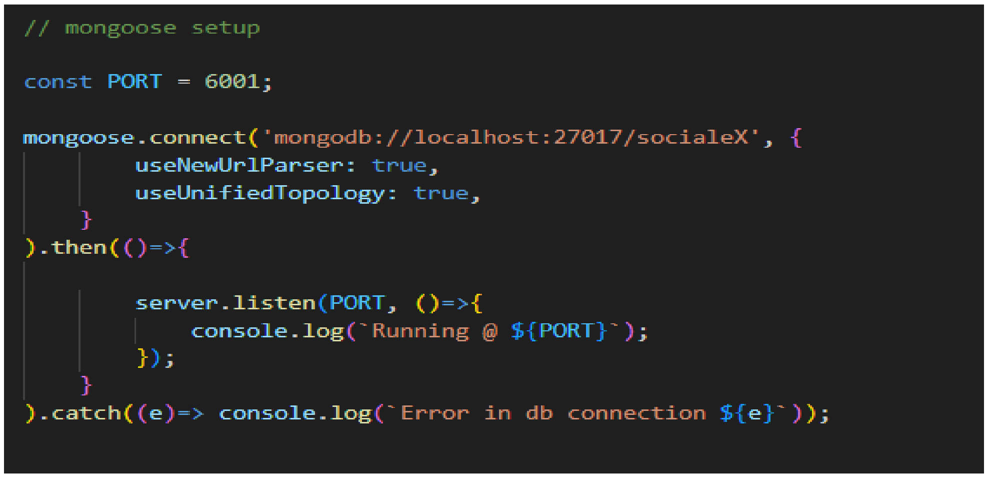
https://drive.google.com/file/d/19J2FH9sfmXMEcgdLhxZQjgoXPMPnU3BB/view?usp=sharing

Milestone 3: Database development:

- Set up a MongoDB database either locally or using a cloud-based MongoDB service like MongoDB Atlas.

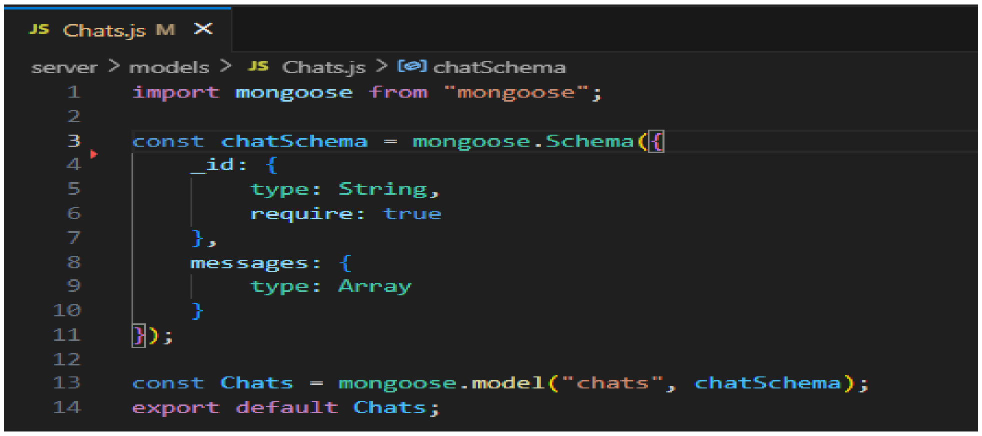
- Create a database and define the necessary collections for users, posts, stories, chats, etc.,

  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}`));
```

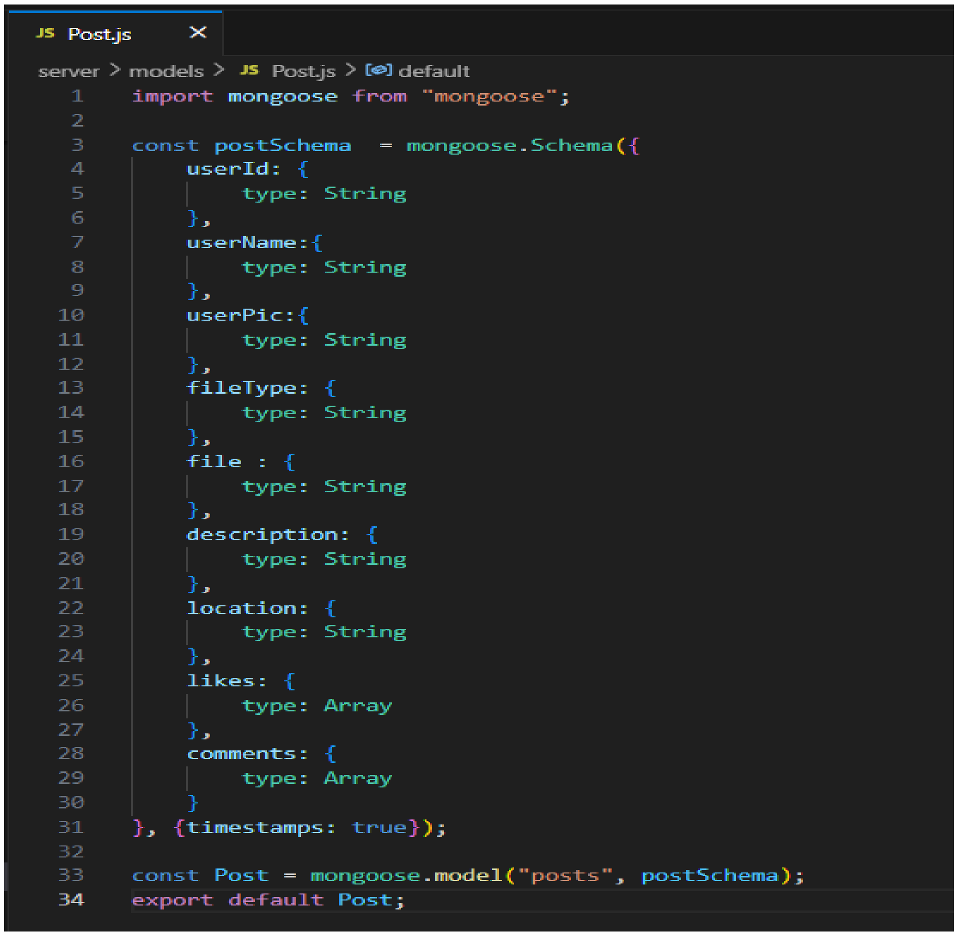
The code for the schemas in the database is provided below



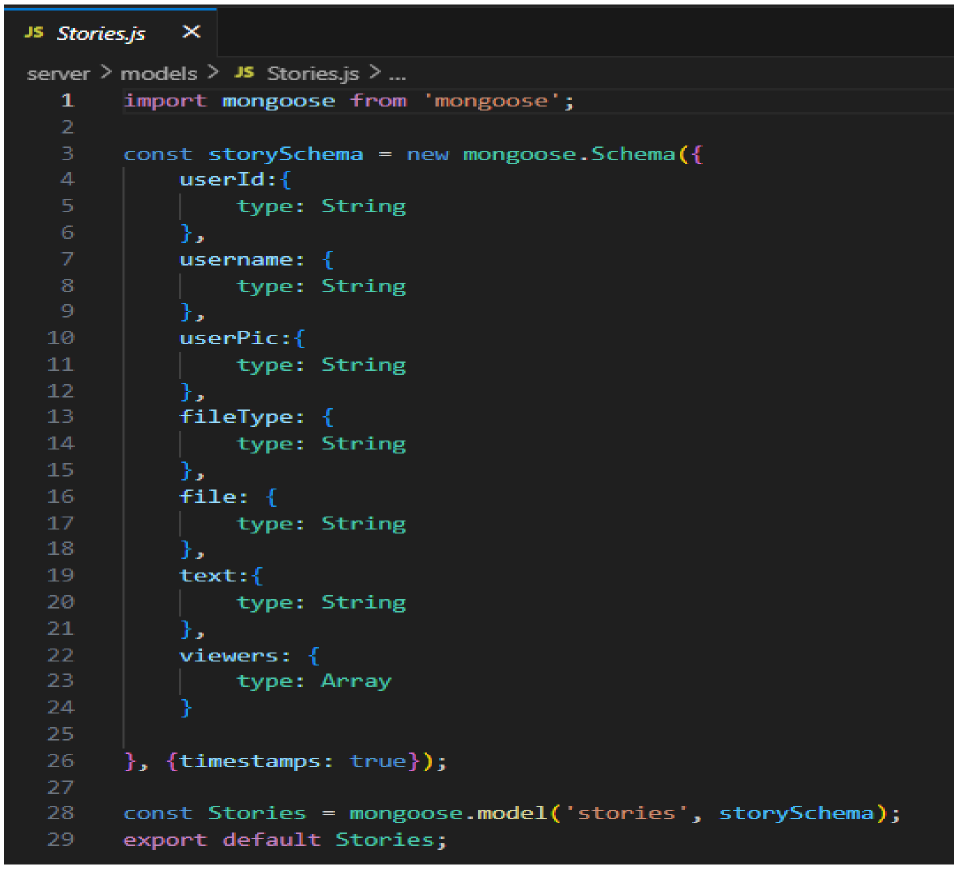
Chats.js M

server > models > Chats.js > chatSchema

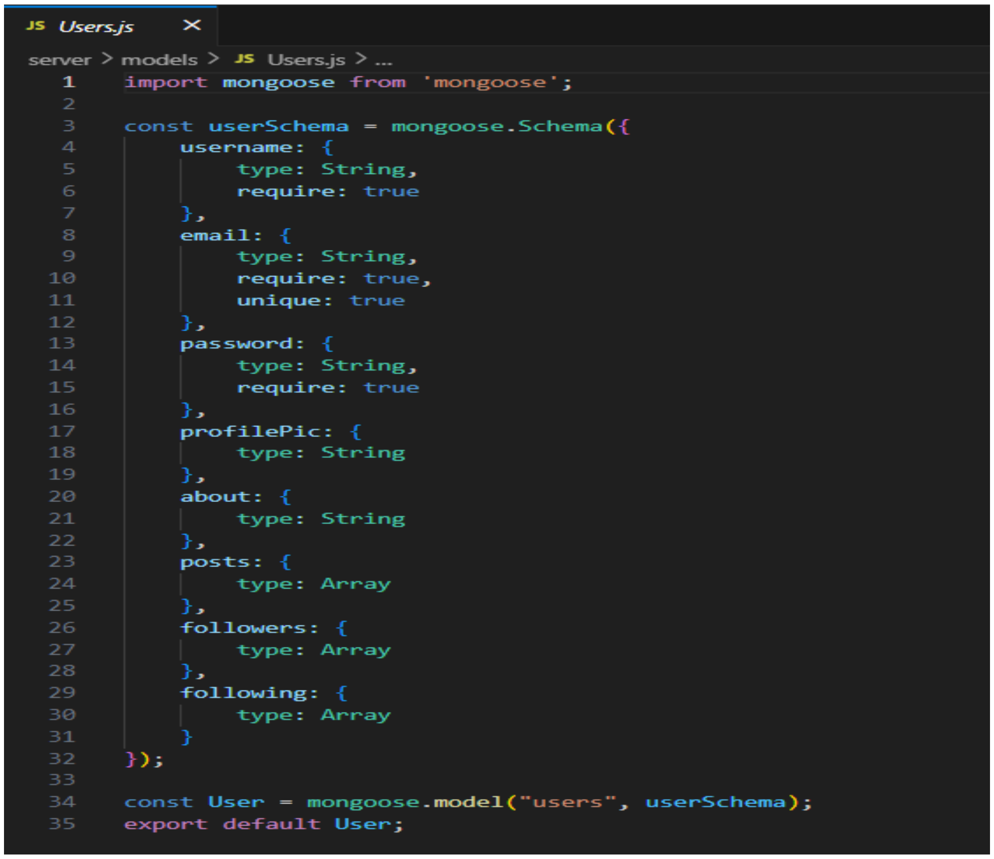
```
import mongoose from "mongoose";

const chatSchema = mongoose.Schema({
    _id: {
        type: String,
        require: true
    },
    messages: {
        type: Array
    }
});

const Chats = mongoose.model("chats", chatSchema);
export default Chats;
```



```
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;
```



```
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;
```


Users.js
server > models > Users.js > ...

```
import mongoose from 'mongoose';

const userSchema = mongoose.Schema({
    username: {
        type: String,
        require: true
    },
    email: {
        type: String,
        require: true,
        unique: true
    },
    password: {
        type: String,
        require: true
    },
    profilePic: {
        type: String
    },
    about: {
        type: String
    },
    posts: {
        type: Array
    },
    followers: {
        type: Array
    },
    following: {
        type: Array
    }
});

const User = mongoose.model("users", userSchema);
export default User;
```

Milestone 4: Frontend development & Integration

- **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.

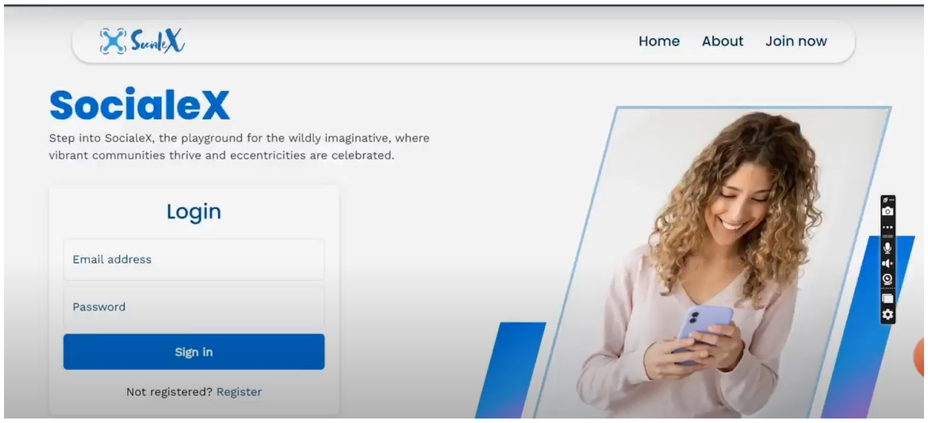
2. Use socket.io for real-time updates.
3. Allow users to share media files in the chat.

- **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.

Milestone 5: Project Implementation:

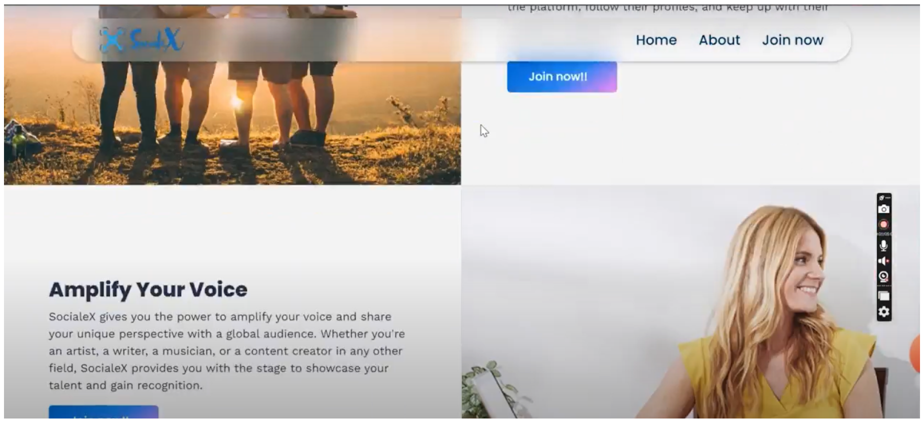
On completing the development part, we then run the application one last time to verify all the functionalities and look for any bugs in it. The user interface of the application looks a bit like the one's provided below.

- **Landing page**



SocialeX

Step into SocialeX, the playground for the wildly imaginative, where vibrant communities thrive and eccentricities are celebrated.

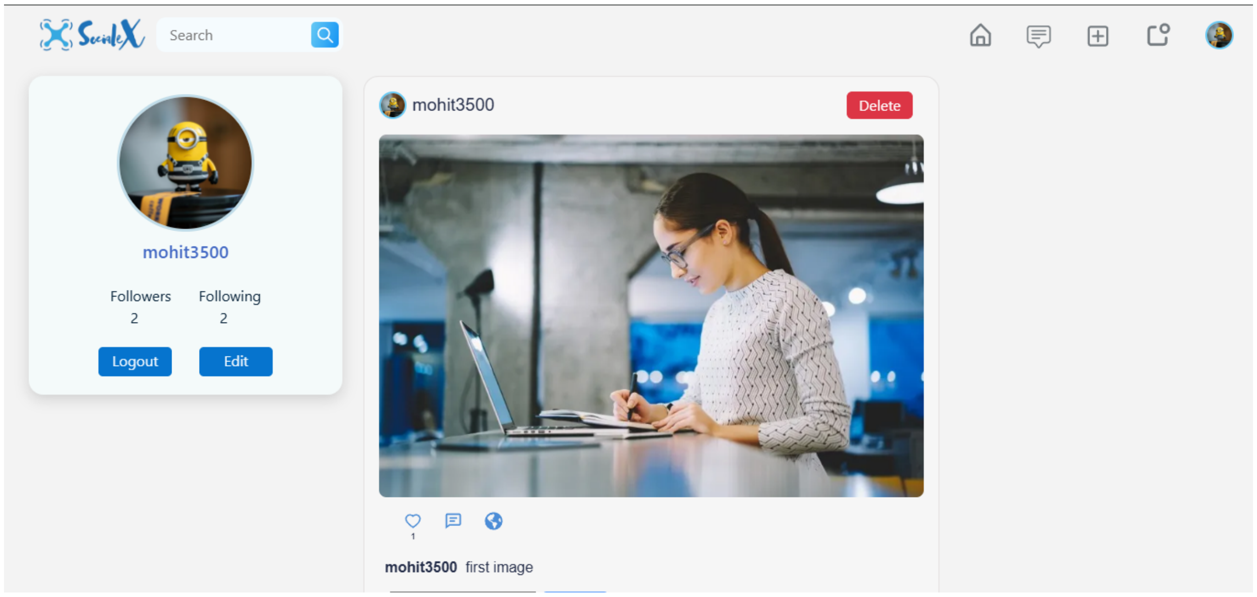


the platform, follow their profiles, and keep up with their

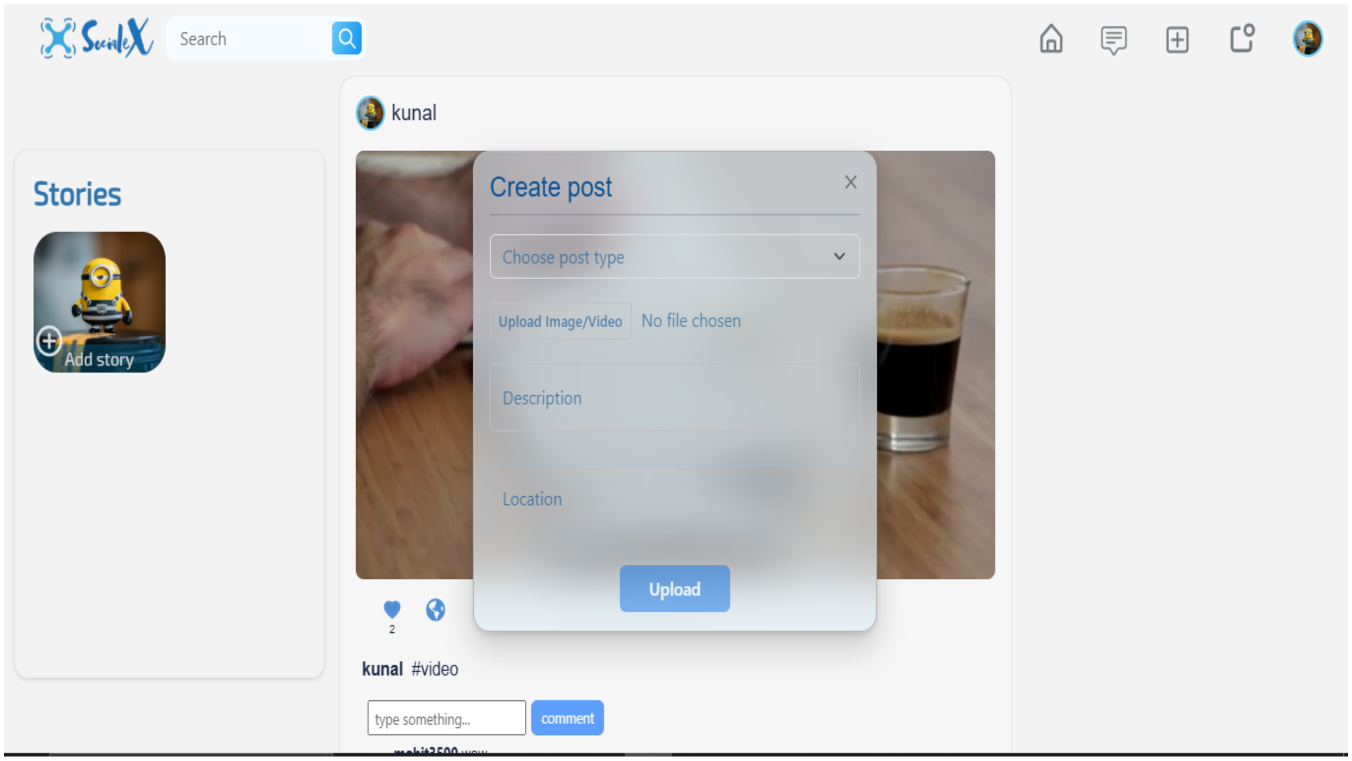
Amplify Your Voice

SocialeX gives you the power to amplify your voice and share your unique perspective with a global audience. Whether you're an artist, a writer, a musician, or a content creator in any other field, SocialeX provides you with the stage to showcase your talent and gain recognition.

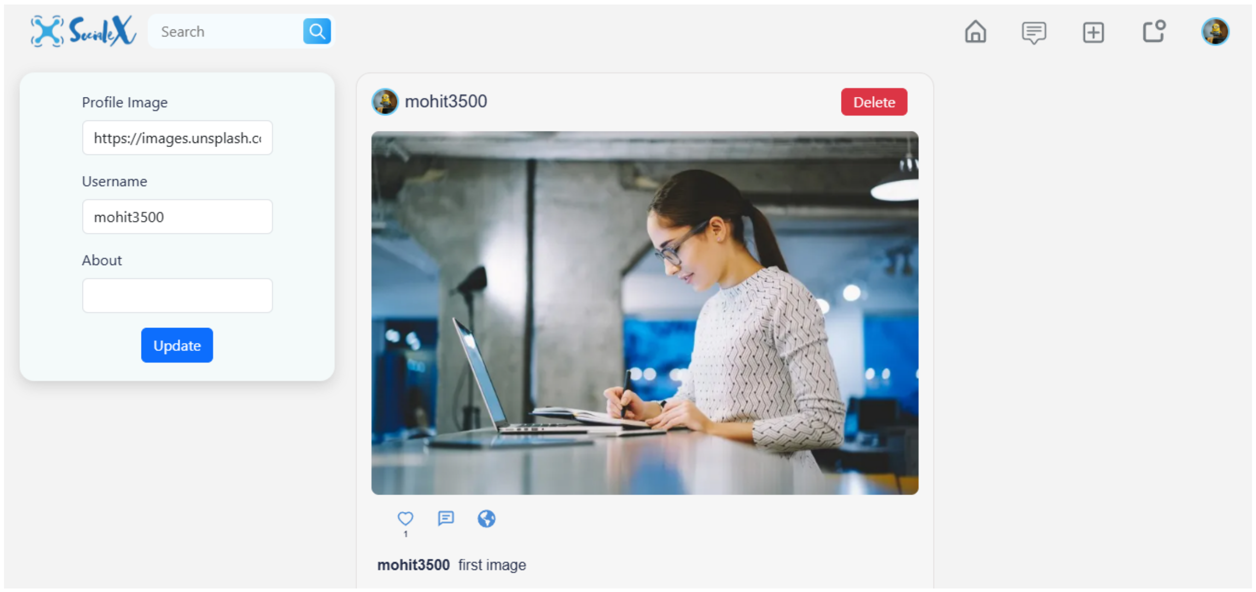
- **Home page**



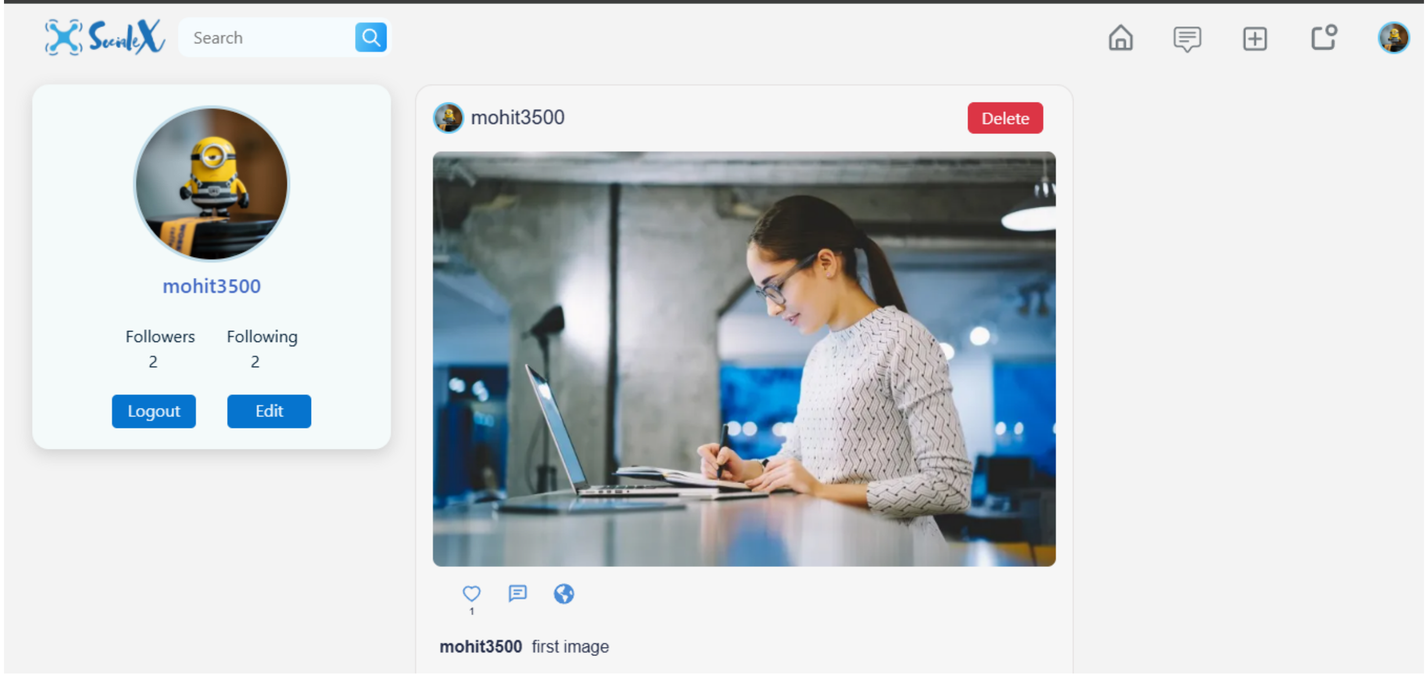
- **Create posts**



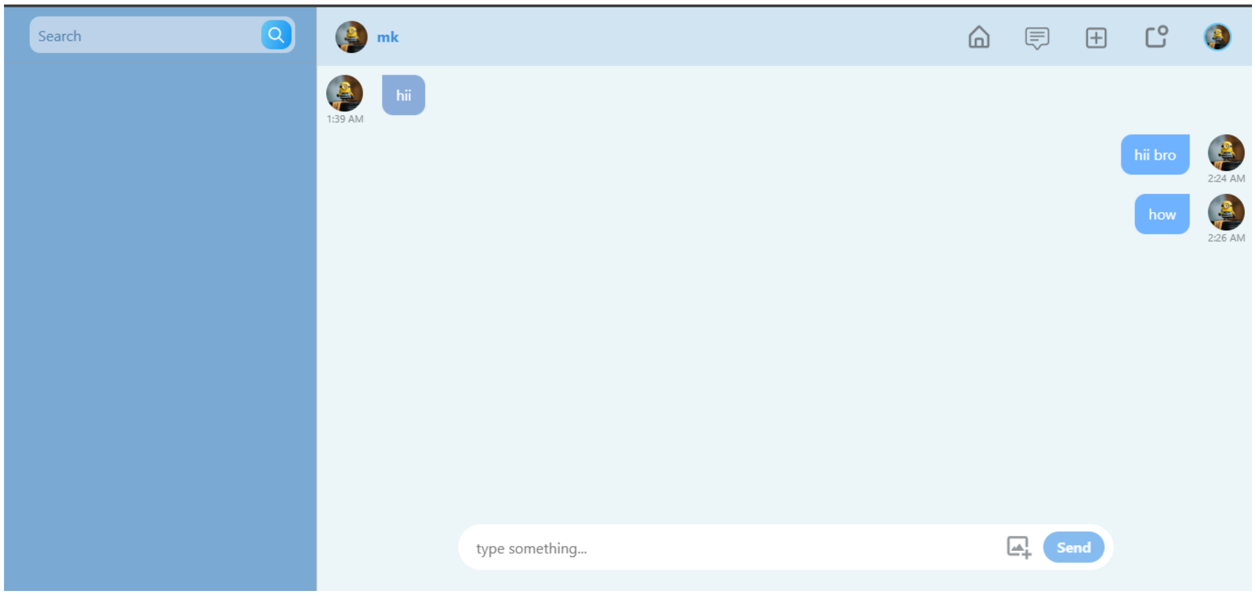
- **Update profile**



- **Profile**



- **Chats**



Thank you for this project

I am happy to be part of smart internz project