**Project Documentation**

**Title: Sociopedia (Social Media Webapp)**

**Team No. : 145206**

**Team Members**

**Ansh Jindal (22BCE10333)**

**Laskshay Peswani (22BCE10484)**

**Manas Dwivedi (22BCE10855)**

**Samrath Thapar (22BCE10978)**

**1. Introduction**

**Project Title**: Sociopedia - A Minimalistic Social Media Platform

**Team Members**:

* **Ansh Jindal (Project Manager)**: Responsible for overseeing the project, ensuring smooth coordination, and managing the overall timeline.
* **Samrath Thapar (Frontend Developer)]**: Handled the development of the frontend using React, ensuring a smooth user interface and experience.
* **Lakshay Peswani (Backend Developer)]**: Managed the backend using Node.js and Express.js, developing the necessary APIs and server-side logic.
* **Manas Dwivedi (Database Administrator)**: Set up and maintained the MongoDB database, including schema creation, data management, and optimization.

**2. Project Overview**

**Purpose**:  
 Sociopedia is a minimalistic social media platform designed to provide users with a simple yet engaging environment to connect, share content, and interact with others. The goal of the project is to offer a clean, user-friendly interface for posting, liking, commenting, and following others, without the complexity and distractions seen in larger platforms. Sociopedia aims to create a space where users can share their thoughts and experiences freely, with an emphasis on privacy and a positive social experience.

**Features**:

* **User Authentication**: Secure login and signup functionality using JSON Web Tokens (JWT) to ensure safe access.
* **Feed**: Real-time feed displaying posts from users that you follow.
* **Post Creation**: Ability for users to create, edit, and delete posts, including text and multimedia content.
* **Likes and Comments**: Users can interact with posts by liking or commenting, enabling social engagement.
* **Follow System**: Users can follow or unfollow other users to stay updated on their posts.
* **Responsive Design**: Optimized for mobile, tablet, and desktop viewing to ensure accessibility across devices.

**3. Architecture**

* **Frontend**:  
   The frontend of Sociopedia is built using **React.js**, a powerful JavaScript library for building user interfaces. It follows a **component-based architecture**, where each part of the UI is broken down into reusable components. React's state management handles dynamic data changes, such as new posts, likes, or comments, ensuring that the UI is always up to date.  
   The frontend is responsible for handling all user interactions, displaying the feed, managing user authentication, and sending requests to the backend APIs. It communicates with the backend using **Axios** for HTTP requests.
* **Backend**:  
   The backend of Sociopedia is built using **Node.js** and **Express.js**, which provide a lightweight and fast environment for building scalable applications.  
   Express.js handles routing, request parsing, and middleware, allowing the backend to manage requests such as creating posts, user authentication, and managing user data. The backend also handles the logic for liking, commenting, and following other users.  
   Authentication is managed through **JWT (JSON Web Tokens)**, ensuring secure and stateless sessions.  
   The server is designed to be lightweight and efficient, handling requests quickly, even under a moderate load.
* **Database**:  
   Sociopedia uses **MongoDB** as its database, which is a NoSQL document-based database.  
   The database stores user data, posts, comments, likes, and follower information in collections. The following is the basic schema design:  
  + **Users Collection**: Contains user profiles, authentication data, and follower/following relationships.
  + **Posts Collection**: Stores all user posts, including text and media content, along with metadata (timestamps, likes, comments).
  + **Comments Collection**: Stores comments on posts, associated with both users and posts.
  + **Likes Collection**: Keeps track of likes on posts, including the users who liked them. MongoDB allows for flexible data storage and scalability, making it a good choice for this project.

**4. Setup Instructions**

**Prerequisites**:  
 Before setting up the project, make sure you have the following software installed on your machine:

* **Node.js** (version 14 or above) - for running JavaScript on the server-side.
* **MongoDB** - a NoSQL database for storing project data.
* **Git** - for cloning the repository from GitHub.
* **npm** - Node package manager, usually comes installed with Node.js.

**Installation**:  
 Follow these steps to set up the application on your local machine:

1. **Clone\_the\_repository** Open a terminal window and run the command to clone the repository:  
     
    git clone https://github.com/your-username/sociopedia.git
2. **Navigate to the project folder** After cloning, navigate into the project directory:  
     
    cd sociopedia
3. **Set up environment variables** Copy the .env.example file to .env and configure your environment variables, such as MongoDB URI, JWT secret, etc.  
    Example:  
     
    MONGO\_URI=mongodb://localhost:27017/sociopedia  
    JWT\_SECRET=your\_jwt\_secret\_key
4. **Install frontend dependencies** Go to the **client** directory and install the required dependencies:  
     
    cd client  
    npm install
5. **Install backend dependencies** Go to the **server** directory and install the required dependencies:  
     
    cd ../server  
    npm install
6. **Run the application**
   * **Frontend**: Start the frontend development server by running the following command inside the **client** directory:  
       
      npm start
   * **Backend**: Start the backend server by running the following command inside the **server** directory:  
       
      npm start
7. **Access the app** After both the frontend and backend servers are running, open your browser and go to:  
     
    http://localhost:3000

**5. Folder Structure**

**Client**:  
 The **client** folder contains all the files related to the frontend of the application. This includes React components, styles, and configuration files for building the user interface.  
 Key subdirectories and files:

* **public**: Contains the static files such as HTML templates, images, and favicon.
* **src**: Holds all the React components, styles, and utility functions for the frontend.  
  + **components**: Contains the various React components like Header, Feed, Post, Comment, etc.
  + **pages**: Contains the pages of the app like Login, Dashboard, and Profile.
  + **styles**: Includes the CSS or SCSS files for styling the application.
  + **utils**: Utility functions for handling repetitive tasks like API calls or validation.
* **package.json**: Manages the dependencies and scripts for the frontend.

**Server**:  
 The **server** folder contains all the files for the backend of the application. It includes the logic for managing the database, authentication, and API routes.  
 Key subdirectories and files:

* **controllers**: Contains the logic for handling requests, such as creating posts or managing user data.
* **models**: Defines the database schemas for users, posts, comments, etc.
* **routes**: Defines the API endpoints like POST /login, GET /posts, etc.
* **middleware**: Contains middleware for authentication, error handling, and other custom functionality.
* **config**: Holds configuration files like environment variables and database connections.
* **app.js**: The entry point to the backend, where the Express app is set up and routes are defined.
* **package.json**: Manages the dependencies and scripts for the backend.

**6. Running the Application**

To run the application locally, follow these steps:

* **Frontend**:  
  1. Open a terminal window and navigate to the **client** directory.
  2. Install the necessary dependencies by running:  
       
      npm install
  3. Start the frontend server by running the following command:  
       
      npm start
  4. This will start the development server and open the application in your default browser. The frontend will be available at:  
       
      http://localhost:3000
* **Backend**:  
  1. Open a terminal window and navigate to the **server** directory.
  2. Install the necessary dependencies by running:  
       
      npm install
  3. Start the backend server by running the following command:  
       
      npm start
  4. The backend server will run and listen for requests, usually on:  
       
      http://localhost:5000

Once both the frontend and backend are running, your application will be fully functional. The frontend will interact with the backend via API calls.

**7. API Documentation**

Below are the details of the API endpoints exposed by the backend of Sociopedia:

1. **User Authentication**

* **POST /api/auth/register** Registers a new user.  
   Request body:  
   {  
   "username": "user123",  
   "email": "user@example.com",  
   "password": "password123"  
   }  
   Response:
  + 200 OK: User registered successfully.
  + 400 Bad Request: Missing or invalid data.
  + 500 Internal Server Error: Server issue.
* **POST /api/auth/login** Logs in an existing user. Returns a JWT token for authentication.  
   Request body:  
   {  
   "email": "user@example.com",  
   "password": "password123"  
   }  
   Response:
  + 200 OK: JWT token.
  + 400 Bad Request: Invalid credentials.
  + 500 Internal Server Error: Server issue.

1. **User Operations**

* **GET /api/users/:userId** Retrieves the profile information of a user.  
   URL parameters:
  + userId: The ID of the user.  
     Response:
  + 200 OK: User data.
  + 404 Not Found: User not found.
* **PUT /api/users/:userId** Updates user profile information.  
   Request body:  
   {  
   "username": "newusername",  
   "bio": "Updated bio"  
   }  
   URL parameters:
  + userId: The ID of the user.  
     Response:
  + 200 OK: User profile updated successfully.
  + 400 Bad Request: Invalid data.
  + 404 Not Found: User not found.

1. **Posts**

* **POST /api/posts** Creates a new post.  
   Request body:  
   {  
   "content": "This is a new post",  
   "image": "image\_url" // optional  
   }  
   Response:
  + 201 Created: Post created successfully.
  + 400 Bad Request: Missing or invalid data.
  + 500 Internal Server Error: Server issue.
* **GET /api/posts** Retrieves a list of all posts.  
   Response:
  + 200 OK: Array of posts.
  + 500 Internal Server Error: Server issue.
* **GET /api/posts/:postId** Retrieves a specific post by its ID.  
   URL parameters:
  + postId: The ID of the post.  
     Response:
  + 200 OK: Post data.
  + 404 Not Found: Post not found.
* **PUT /api/posts/:postId** Updates a specific post.  
   Request body:  
   {  
   "content": "Updated content for the post"  
   }  
   URL parameters:
  + postId: The ID of the post.

Response:

* + 200 OK: Post updated successfully.
  + 404 Not Found: Post not found.
  + 400 Bad Request: Invalid data.

1. **Comments**

* **POST /api/posts/:postId/comments** Adds a comment to a specific post.  
   Request body:  
   {  
   "comment": "This is a comment"  
   }  
   URL parameters:
  + postId: The ID of the post.  
     Response:
  + 201 Created: Comment added successfully.
  + 404 Not Found: Post not found.
* **GET /api/posts/:postId/comments** Retrieves all comments for a specific post.  
   URL parameters:
  + postId: The ID of the post.

Response:

* + 200 OK: Array of comments.
  + 404 Not Found: Post not found.

1. **Likes**

* **POST /api/posts/:postId/like** Likes a specific post.  
   URL parameters:
  + postId: The ID of the post.  
     Response:
  + 200 OK: Post liked successfully.
  + 404 Not Found: Post not found.
* **DELETE /api/posts/:postId/like** Removes a like from a specific post.  
   URL parameters:
  + postId: The ID of the post.  
     Response:
  + 200 OK: Like removed successfully.
  + 404 Not Found: Post not found.

**8. Authentication**

Authentication in Sociopedia is handled using JWT (JSON Web Tokens). This ensures that the users can securely log in and access the application with their credentials. Below is the process used for authentication:

* **User Registration**
  + A new user can register by providing their **username**, **email**, and **password**. The password is hashed before storing it in the database for security purposes.
  + The **POST /api/auth/register** endpoint is used for user registration.
  + If the registration is successful, the user can log in immediately using the credentials provided.
* **User Login**
  + Once the user registers, they can log in using their **email** and **password**.
  + The **POST /api/auth/login** endpoint is used for logging in.
  + Upon successful login, the server returns a **JWT token**, which can be used to authenticate further requests.
  + The JWT token is stored on the client-side (usually in **localStorage** or **sessionStorage**) and is sent in the **Authorization** header of subsequent requests to access protected routes.
* **Token-Based Authentication**
  + For every request that requires authentication, the **Authorization** header should include the JWT token. The format of the header is as follows: Authorization: Bearer <jwt-token>
  + The backend verifies the token on every request. If the token is valid, the user is allowed to access the requested resource. If the token is invalid or expired, the user is redirected to the login page or receives a **401 Unauthorized** response.
* **Token Expiry and Refresh**
  + JWT tokens have a limited lifespan, and after expiration, the user must log in again to obtain a new token.
  + For improved user experience, an optional **refresh token** can be implemented to allow automatic token renewal without requiring the user to log in again. This can be handled by creating an endpoint to refresh the token before the current one expires.
* **Role-Based Access Control**
  + In addition to authentication, **authorization** is also implemented, which ensures that users can only access the resources they are allowed to. This is done by assigning roles (e.g., **admin**, **user**) and verifying the role during the request processing.
  + For example, only **admin** users can access certain protected routes like managing users or posts.

**9. User Interface**

The user interface (UI) of Sociopedia is designed to be clean, intuitive, and easy to navigate. Below are the key UI features:

1. **Homepage**
   * The homepage displays the latest posts and a navigation bar for accessing various sections like Profile, Posts, Notifications, etc.
   * Users can view, like, comment on posts, and create their own posts from this page.
2. **Login and Registration Screens**
   * The login screen allows users to enter their credentials (email and password) to access the platform.
   * The registration screen allows new users to create an account by providing their username, email, and password.
   * Both screens have clear input fields, labels, and error messages for easy user interaction.
3. **Profile Page**
   * Each user has their own profile page, where they can view their posts, bio, and profile picture.
   * Users can edit their profile information, such as username, bio, and profile image.
   * The profile page displays the number of followers and posts.
4. **Post Creation**
   * Users can create posts directly from the homepage or the profile page.
   * A post consists of text content, and optionally, users can upload an image.
   * The post creation interface includes a text editor for content and an option to attach an image.
5. **Comments and Likes**
   * Every post has a comment section where users can leave their thoughts.
   * The like button allows users to like posts and shows the number of likes each post has received.
6. **Responsive Design**
   * The design is fully responsive, meaning it works well on both desktop and mobile devices.
   * Elements adjust automatically to fit the screen size, making it user-friendly on any device.
7. **Notifications**
   * Users receive notifications about likes, comments, and new followers.
   * A notification bell icon displays the number of unread notifications.
8. **Settings**
   * Users can access their account settings, where they can update their personal details, change the password, and log out.

**10. Testing**

Testing is an essential part of ensuring that Sociopedia functions correctly and provides a seamless user experience. Below is an overview of the testing strategy employed for this project:

1. **Unit Testing**
   * Unit tests were written for individual components and functions to ensure that each part of the application works as expected.
   * Tools like **Jest** were used for unit testing the frontend components in React.
   * Each component, such as login forms, post creation, and comment handling, was tested independently to verify its behavior.
2. **Integration Testing**
   * Integration tests were used to verify that the various parts of the application interact correctly with one another.
   * The integration testing focused on ensuring that data flows properly between the frontend and the backend.
   * It tested scenarios such as posting content, user registration, login functionality, and user interactions with posts.
3. **End-to-End (E2E) Testing**
   * **Cypress** was used for end-to-end testing to simulate real user interactions with the application.
   * E2E tests were written to verify that the entire flow of the application, from logging in to posting content, works as expected.
   * The tests cover scenarios like user login, post creation, liking a post, and commenting.
4. **Performance Testing**
   * Load testing was performed to ensure the application can handle multiple simultaneous users without significant performance degradation.
   * **Apache JMeter** was used to simulate multiple user interactions to test the scalability of the application.
5. **User Acceptance Testing (UAT)**
   * User acceptance testing was conducted with a small group of beta users to gather feedback on the functionality and usability of the application.
   * The goal was to identify any issues that may not have been covered in the earlier testing phases and to ensure the application meets user expectations.
6. **Bug Tracking and Fixing**
   * A bug tracking system was used to log any issues or bugs found during testing.
   * Each bug was reviewed and prioritized for fixing, ensuring that critical issues were addressed first.
   * The team worked on fixing the bugs, and after each fix, the tests were rerun to verify the resolution.

**11. Screenshots or Demo**

<https://drive.google.com/drive/folders/1fgJbbwBz4USbli47OiTBZSYCB1LbkePb>

**12. Known Issues**

While Sociopedia has been extensively tested, there are still a few known issues that may affect some users:

1. **Slow Load Times for High-Traffic Pages**
   * Some pages, especially those with a large number of posts or comments, may experience slow load times under heavy traffic. Optimization for these pages is still in progress.
2. **Responsive Design on Older Devices**
   * While the app works well on modern devices, certain older smartphones may not render the interface correctly, particularly with images and post previews. This issue will be addressed in future updates for better cross-device compatibility.
3. **Profile Image Upload Failures**
   * A few users have reported issues uploading profile pictures on some devices. This may be related to file size or network issues. Improvements to the image upload functionality are being planned.
4. **Commenting on Posts**
   * Occasionally, users may experience difficulty submitting comments on posts if the internet connection is slow or intermittent. The team is working on enhancing the commenting system to ensure it is more resilient in low connectivity situations.
5. **Notifications Delay**
   * In certain situations, users may experience delays in receiving notifications for actions like likes, comments, or new followers. This is an ongoing issue related to real-time notifications, and improvements are being made.
6. **Email Confirmation Not Received**
   * Some users have reported not receiving email confirmation after registering. This is mostly due to email service provider filtering. Users can try resending the confirmation or manually check their spam folders.
7. **Minor UI Bugs**
   * A few minor UI inconsistencies have been noted, especially on mobile devices. These are being addressed as part of the continuous UI refinement process.

**13. Future Enhancements**

While Sociopedia is a fully functional social media platform, there are several enhancements and features planned for future versions. Below are some of the key areas for improvement:

1. **Improved Real-Time Notifications**
   * The platform will introduce a more advanced real-time notification system, allowing users to receive instant updates on activities like comments, likes, and follows without any delay.
2. **Better Image Upload and Optimization**
   * Future updates will focus on improving the image upload feature to support larger file sizes and faster upload times, ensuring better performance on all devices.
3. **Enhanced User Profiles**
   * Users will be able to personalize their profiles further with additional features such as cover images, detailed bio sections, and the ability to organize posts into albums or categories.
4. **Post Editing and Deletion**
   * A feature will be added allowing users to edit or delete their posts after they have been published. This will provide more flexibility and control over content.
5. **Advanced Search and Filtering**
   * An advanced search functionality will be introduced, allowing users to search for posts, users, and comments based on specific keywords, hashtags, or categories.
6. **Mobile App Development**
   * While the web application is fully responsive, there are plans to develop dedicated mobile applications for iOS and Android to provide a more seamless and optimized experience for mobile users.
7. **Privacy Settings**
   * Additional privacy features will be implemented, including the ability for users to control who can see their posts, who can comment, and who can follow them.
8. **Content Moderation Tools**
   * To ensure a positive and safe environment, content moderation features will be introduced, such as automatic flagging of inappropriate content and an improved reporting system for harmful behavior.
9. **User Analytics**
   * A feature for users to see insights on their posts' performance, including likes, comments, and reach, will be added, helping users track their social influence.
10. **Multi-Language Support**

* Sociopedia will expand its user base by supporting multiple languages, making the platform accessible to users worldwide.

1. **Monetization Options for Content Creators**

* Future versions may offer content creators opportunities to monetize their posts, such as through ads or paid subscriptions, providing new ways to earn income on the platform.