Assignment - Manasvi Technologies OPC Pvt. Ltd.

===

Project: Secure Real-time Chat Application with Google OAuth and Manual Login

Overview

Create a secure real-time chat application that supports both Google OAuth and manual login/signup. The application should enable one-to-one and group chats. An admin user should have the ability to view all personal chats of users without logging in. The frontend should be a single-page application built with React.

Requirements

User Authentication

Implement Google OAuth for login using Passport.js.
Implement manual signup and login using JWT for authentication.
Use bcrypt for password hashing.
Implement password recovery using email.

Chat Functionality

Enable one-to-one and group chat functionality.
Allow users to send text, images, and videos in chats.
Use WebSockets (Socket.io) for real-time messaging.
Store chat history in MongoDB.

Admin Functionality

Admin can view all personal chats of users without needing to log in as them.
Implement role-based access control: user and admin.
Admin should have a separate dashboard to view all user activities.

Profile Management

Users can update their profile details, including changing their profile picture.
Users can view other users’ profiles by clicking on them.

Security

Secure backend routes using JWT.

Ensure frontend routes are protected and accessible only to authenticated users.

Frontend

Use React for the frontend with functional components and hooks.
Implement Context API or Redux for state management.
Implement responsive design to ensure usability on various devices.
Use React Router for navigation.

Backend

Use Express.js for the server.
MongoDB for the database, with Mongoose for ORM.

File Uploads

Implement file uploads for profile pictures and chat media (images, videos).

Deployment

Deploy the application

Use environment variables for sensitive information (e.g., database connection strings, JWT secret).

Bonus Features (Optional)
Implement read receipts in chats.
Add typing indicators in chats.

Use TypeScript for type safety in both frontend and backend.
Evaluation Criteria
Code quality and organization.
Adherence to MERN stack best practices.
Proper use of Git for version control.
Completeness and functionality of the features.
User experience and UI design.

Submission
=========
Share a GitHub repository link with the complete project code.
->Submission Date:14/6/2024
Task Breakdown
Step 1: User Authentication
Setup Passport.js for Google OAuth.
Implement JWT-based manual signup and login.
Create routes for signup, login, and password recovery.

Step 2: Real-time Chat
Setup Socket.io for real-time communication.
Implement chat models in MongoDB (one-to-one and group chat schemas).
Create routes and controllers for sending and receiving messages.

Step 3: Admin Dashboard
Create an admin dashboard to view all user activities.
Implement role-based access control in the backend.

Step 4: Profile Management
Create profile update routes.
Implement file upload for profile pictures.
Allow users to view and update their profiles.

Step 5: Frontend Implementation
Setup React with functional components and hooks.
Implement authentication flows (login, signup, password recovery).
Create chat interfaces for one-to-one and group chats.
Implement profile management and user navigation.

Step 6: Security
Secure backend routes with JWT.
Protect frontend routes and ensure only authenticated users can access certain pages.

Step 7: Deployment
Deploy the application anywhere like onrender ,netlify, heruku .
Use environment variables to manage sensitive information.