Social Media Backend REST-API

Goal

Develop a robust social media backend REST-API that empowers users to post, comment, like, send friend requests, and reset their passwords using OTP for enhanced security.

Acceptance Criteria

RESTful Architecture

- Develop a RESTful API using Node.js, ExpressJS, and MongoDB for efficient data handling and routing control.

Code Modularity

- Organize code using ES6 Modules for maintainability and modularity.

User Authentication

- Implement a user authentication system with features such as signup, login, and logout.
- Bonus: Implement the ability to log out from all devices by storing each login token in an additional array field within the user's document.
- Registration includes user details such as name, email, password, and gender.

Post Management

- Implement CRUD operations for posts, including fields like caption and an image URL.
- Ensure each post references the user who created it.
- Posts can be updated or deleted only by the post owner.

Comment System

- Develop a comment system allowing users to add, update, and delete comments on posts.
- Comments can be updated or deleted only by the post owner or the commenter.

Like Functionality

- Create a like system for posts, including logic with MongoDB and population of documents.
- Display counts of likes and comments on posts.
- Populate user information (id, name, and email) for likes, comments, and posts.

Friendship Features

- Implement a friendship system with features like getting user friends, managing pending friend requests, toggling friendships, and accepting/rejecting friend requests.

User Profile Updates

- Enable users to update their profiles, including fields like name, gender, or avatar.
- Implement avatar uploads for user profiles.

OTP-Based Password Reset (Additional Task)

- Implement OTP-based password reset feature.
- Create controllers, models, and repositories for OTP management.
- Utilize the Nodemailer library for email communication.

Tasks

1. Project Setup

- Set up an Express.js application and configure related settings.

2. Dependency Installation

- Install the necessary project dependencies based on the required functionalities.

3. User Authentication

- Implement user registration and login routes.
- Develop user logout routes.

4. User Profile

- Create routes for getting user details and updating user profiles.
- Implement avatar uploads.

5. Post Management

- Set up routes and controllers for CRUD operations on posts.
- Handle image uploads for post images.

6. Comment System

- Develop routes and controllers for managing comments on posts.

7. Like Functionality

- Create routes and logic for liking and unliking posts and comments.

8. Friendship Features

- Implement routes and controllers for user friendships, including getting friends, and accepting/rejecting requests.

9. OTP-Based Password Reset

- Set up routes and controllers for sending OTPs, verifying OTPs, and resetting passwords.

10. Error Handling and Logging

- Implement error handling middleware and request logging.

11. Testing and Documentation

- Thoroughly test the API to ensure it meets acceptance criteria.
- Document the application's functionalities, dependencies, and code organization for clarity.

API Structure

The API structure for the "Social-Media" project can be organized as follows:

Authentication Routes

- `/api/users/signup`: Register a new user account.
- `/api/users/signin`: Log in as a user.
- `/api/users/logout`: Log out the currently logged-in user.
- `/api/users/logout-all-devices`: Log out the user from all devices.

User Profile Routes

- `/api/users/get-details/:userId`: Retrieve user information.
- `/api/users/get-all-details`: Retrieve information for all users.
- `/api/users/update-details/:userId`: Update user details.

Post Routes

- `/api/posts/all`: Retrieve all posts from various users.
- `/api/posts/:postId`: Retrieve a specific post by ID.
- `/api/posts/`: Retrieve all posts for a specific user.
- `/api/posts/`: Create a new post.

- `/api/posts/:postId`: Delete a specific post.
- `/api/posts/:postId`: Update a specific post.

Comment Routes

- `/api/comments/:postId`: Get comments for a specific post.
- `/api/comments/:postId`: Add a comment to a specific post.
- `/api/comments/:commentId`: Delete a specific comment.
- `/api/comments/:commentId`: Update a specific comment.

Like Routes

- `/api/likes/:id`: Get likes for a specific post or comment.
- `/api/likes/toggle/:id`: Toggle like on a post or comment.

Friendship Routes

- `/api/friends/get-friends/:userId`: Get a user's friends.
- `/api/friends/get-pending-requests`: Get pending friend requests.
- `/api/friends/toggle-friendship/:friendId`: Toggle friendship with another user.
- `/api/friends/response-to-request/:friendId`: Accept or reject a friend request.

OTP Routes

- `/api/otp/send`: Send an OTP for password reset.
- `/api/otp/verify`: Verify an OTP.
- `/api/otp/reset-password`: Reset the user's password.

Error Handling and Logging

Implement error handling middleware and request logging to ensure smooth API operation and easy debugging.

Testing and Documentation

Thoroughly test the API to ensure it meets acceptance criteria and document the application's functionalities, dependencies, and code organization for clarity.

Evaluation Criteria

Your project will be evaluated on the following parameters:

EXPRESS.JS, MONGOOSE & MONGODB SETUP

(Max Score 10)

- Implementation of Express.js using MVC architecture with ES6 Modules.
- Configuration of Mongoose for efficient MongoDB interaction.

CODE MODULARITY AND ORGANIZATION

(Max Score 20)

- Well-structured and modular code following naming conventions and best practices.
- Thorough comments and documentation for future development and collaboration.

USER AUTHENTICATION

(Max Score 20)

- Implementation of a secure and robust user authentication system with necessary security measures.

POST MANAGEMENT AND COMMENT SYSTEM

(Max Score 15)

- Development of a functional post management system with an integrated comment system.

LIKE FUNCTIONALITY AND POPULATED DATA

(Max Score 15)

- Implementation of 'like' functionality and proper data population for a seamless user experience.

FRIENDSHIP FEATURES AND USER PROFILE UPDATES

(Max Score 10)

- Incorporation of user connection features and user profile updates.

OTP-BASED PASSWORD RESET

(Max Score 10)

- Integration of a secure OTP-based password reset mechanism.

ADDITIONAL TASKS

(Max Score 10)

- Successful completion of all additional tasks specified for the project.

INNOVATION

(Max Score 10)

- Innovative utilization of Node.js and Express.js features.
- Performance optimization best practices.
- Creative user interface and user experience design with minimized reliance on external dependencies.