Post Management

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

Comment System

- Develop a comment system that allows 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)

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

▸ **Tasks**

▾ **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, ensuring sensitive data like passwords is not exposed.
- **/api/users/get-all-details:** Retrieve information for all users, avoiding display of sensitive credentials like passwords.
- **/api/users/update-details/userId:** Update user details while ensuring that sensitive data like passwords remains secure and undisclosed.

Post Routes

- /api/posts/all: Retrieve all posts from various users to compile a news feed.
- **/api/posts/postId:** Retrieve a specific post by ID.
- **/api/posts/:** Retrieve all posts for a specific user to display on their profile page.
- **/api/posts/:** Create a new post.
- **/api/posts/postId:** Delete a specific post.
- **/api/posts/postId:** Update a specific post.

Note that for the same routes, you can change the HTTP methods (GET, POST, PUT, DELETE).
For example,
- **Use DELETE("/api/posts/:postId")** to delete a specific post.
- **Use PUT("/api/posts/:postId")** to update a specific post.

In both cases, the route remains the same; only the HTTP method is changed.

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.

Note: For the same routes, change the HTTP methods (GET, POST, PUT, DELETE).

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.

1 Step 1
Complete project on local IDE

2 Step 2
Export code as .zip file

3 Step 3
Upload .zip file max 50mb

Evaluation criteria	Score
EXPRESS JS, MONGOOSE & MONGODB SETUP	50
CODE MODULARITY AND ORGANIZATION	100
USER AUTHENTICATION	100
POST MANAGEMENT AND COMMENT SYSTEM	75
LIKE FUNCTIONALITY AND POPULATED DATA	75
FRIENDSHIP FEATURES AND USER PROFILE UPDATES	50
OTP-BASED PASSWORD RESET	50
ADDITIONAL TASKS	50
INNOVATION	50
Total score	600

Drag and drop .zip here or Browse

Submit

Ask a doubt