Node.JS Certification Training

Project - 2

Problem Statement:

We need to create APIs for a social media platform that allow users to upload, like, comment on, and delete photos and videos, along with a real-time chat feature. This involves handling media uploads and deletions, managing files, integrating MongoDB for likes and comments, and using socket connections for real-time messaging with instant chat updates.

Goal:

The goal of this project is to:

- Develop APIs for user registration, profile updates, account deletion, and authentication.
- Implement APIs for media upload, deletion, and file system management.
- Create APIs for media interactions such as liking and commenting on posts.
- Integrate real-time chat functionality using socket connections.

Web Application Requirement:

- **User Management**

1. **User Registration:** API to register a new user.

   Input: { username, email, password }

   Response: { userId, username, email }

2. **Update User Profile:** API to update user profile details.

   Input: { username (optional), email (optional) }

   Response: { userId, username, email }

3. **Delete User Account:** API to delete a user account.

   Input: { userId }

   Response: { message: "User account deleted successfully" }

4. **User Authentication:** Implement JWT-based authentication for secure access.

   Input: { email, password }

   Response: { token }

- **User Media Management**

1. **Upload Media:**API for users to upload photos or videos.

   Input: { userId, mediaFile, description }

   Response: { mediaId, mediaURL, description }

2. **Delete Media:**API to delete user-uploaded media.

   Input: { mediaId }

   Response: { message: "Media deleted successfully" }

3. **File System Management:**Ensure media files are stored and managed in the appropriate file system.

- **Media Interactions**

  1. **Like Media:** API to like a media post.

     Input: { userId }

     Response: { mediaId, likesCount }

  2. **Comment on Media:** API to comment on a media post.

     Input: { userId, comment }

     Response: { commentId, comment }

- **Real-time Chat**

  1. **Real-time Messaging**: Implement real-time chat using socket connections.

     Data: { senderId, receiverId, message }

Web Application Implementation:

Technologies:

- Backend: Node.js, Express.js
- Database: MongoDB
- Real-time Communication: Socket.IO

Endpoints:

Define appropriate route names and API endpoint names for user management, media management, media interactions, and real-time chat functionalities.

Development Considerations:

- Ensure all API operations are authenticated.

- Use appropriate HTTP methods (GET, PUT, POST, DELETE) and status codes.
- Maintain a readable and maintainable code structure.
- Implement unit test cases for APIs.
- Ensure easy environment setup for a large team.
- Configure environment-specific data such as database connections.