**Project Report**

**Social Media Post Management System**

**Introduction**

The **Social Media Post Management System** is a RESTful API built using **Spring Boot** and **MySQL**. This project enables users to **create, update, delete, and manage posts and comments**, along with user authentication and **likes functionality for both posts and comments**.

**Key Features:**

✔ User authentication (Sign up & Login)  
✔ Post creation, editing, and deletion  
✔ Commenting on posts  
✔ **Liking posts and comments**  
✔ Pagination for posts and comments  
✔ OpenAPI (Swagger) documentation

**Technologies Used**

| **Technology** | **Purpose** |
| --- | --- |
| **Spring Boot** | Backend framework |
| **Spring JDBC (NamedParameterJdbcTemplate)** | Database interaction |
| **MySQL** | Database management |
| **Swagger/OpenAPI** | API documentation |
| **Postman** | API testing |
| **Maven** | Dependency management |

**Project Structure**

src/main/java/org/ncu/SocialMediaPostManagement

│── Controller

│ ├── UserController.java

│ ├── PostController.java

│ ├── CommentController.java

│ ├── LikeController.java

│── Entity

│ ├── User.java

│ ├── Post.java

│ ├── Comment.java

│ ├── Like.java

│── Repository

│ ├── UserRepository.java

│ ├── PostRepository.java

│ ├── CommentRepository.java

│ ├── LikeRepository.java

│── Service

│ ├── UserService.java

│ ├── PostService.java

│ ├── CommentService.java

│ ├── LikeService.java

└── Swagger

├── SwaggerConfig.java

**Features Implemented**

**✅ Post Management**

* **Create a new post**
* **Edit an existing post**
* **Delete a post**
* **Retrieve posts with pagination (?page=1&size=5)**

**✅ Comment Management**

* **Add comments to a post**
* **Retrieve all comments for a post**

**✅ Pagination**

* Implemented pagination using page and size parameters.

**✅ Named Parameters in SQL Queries**

* Used NamedParameterJdbcTemplate for **better readability and security** in SQL queries.

**✅ Transaction Management**

* Used **@Transactional** annotation to ensure data consistency.

**Database Schema**

We used **MySQL** as the database. The schema includes two tables:

1. **posts**: Stores post details (id, content, likes).
2. **comments**: Stores comments related to posts.

**Database Schema (MySQL)**

**create database SMM;**

**use SMM;**

**CREATE TABLE posts (**

**id BIGINT AUTO\_INCREMENT PRIMARY KEY,**

**content TEXT NOT NULL,**

**created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,**

**updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP**

**);**

**CREATE TABLE comments (**

**id BIGINT AUTO\_INCREMENT PRIMARY KEY,**

**post\_id BIGINT NOT NULL,**

**content TEXT NOT NULL,**

**created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,**

**FOREIGN KEY (post\_id) REFERENCES posts(id) ON DELETE CASCADE**

**);**

**CREATE TABLE post\_likes (**

**post\_id BIGINT NOT NULL,**

**user\_id BIGINT NOT NULL,**

**PRIMARY KEY (post\_id, user\_id),**

**FOREIGN KEY (post\_id) REFERENCES posts(id) ON DELETE CASCADE );**

**Implementation Details**

**(i) User Management**

Users can register and log in using **email and password**.

**🔹 User Entity (User.java)**

**public class User {**

**private int id;**

**private String email;**

**private String password;**

**}**

**(ii) Post Management**

**Users can create, retrieve, update, and delete posts.**

**🔹 Post Entity (Post.java)**

**public class Post {**

**private int id;**

**private String title;**

**private String content;**

**private int userId;**

**}**

**(iii) Comment Management**

**Users can add comments on posts.**

**🔹 Comment Entity (Comment.java)**

**public class Comment {**

**private int id;**

**private int postId;**

**private int userId;**

**private String content;**

**}**

**(iv) Likes Management**

**Users can like posts and comments.**

**🔹 Like Entity (Like.java)**

**public class Like {**

**private int id;**

**private int userId;**

**private Integer postId;**

**private Integer commentId;**

**}**

* **Posts**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

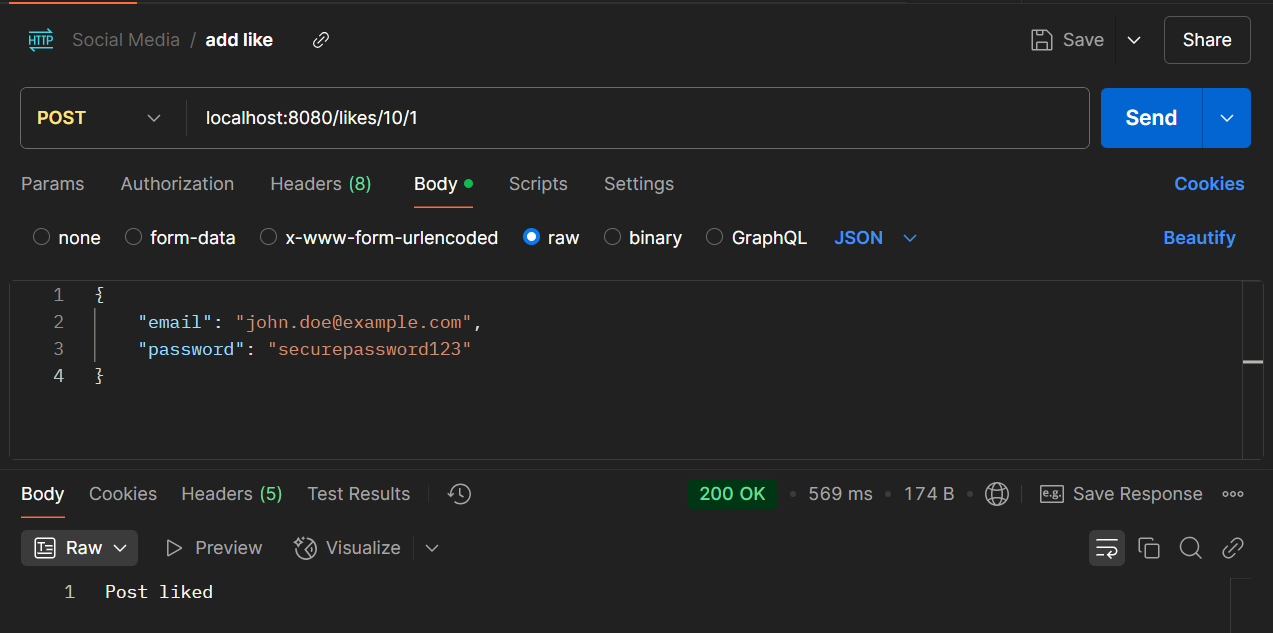
A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* **Likes**



A screenshot of a computer

AI-generated content may be incorrect.

* User

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* Comments

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

* **Swagger API Documentation**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* **Conclusion**

This project successfully implements a **Spring Boot-based Social Media Post Management System** with **user authentication, post creation, comments, and likes feature for both posts and comments**.