Certainly! Here's a detailed backend project idea for a Social Media Platform API using Java with Spring Boot, Hibernate, JPA, Swagger, JUnit, and Mockito:

Project Idea: Social Media Platform API

Project Description: Create a robust backend system for a Social Media Platform API that includes user management, posting, liking, commenting, following, and notifications. This project will incorporate Java technologies such as Spring Boot for RESTful APIs, Hibernate and JPA for database interaction, Swagger for API documentation, and thorough testing with JUnit and Mockito.

Key Features:

User Authentication and Authorization:

Implement user registration and login.

Use JWT (JSON Web Tokens) for user authentication and role-based access control.

Allow users to reset their passwords.

User Profiles:

Create and manage user profiles with information like username, email, profile picture, and bio.

Enable users to update their profiles.

Posting and Feeds:

Allow users to create, edit, and delete posts.

Implement a news feed that displays posts from followed users.

Support privacy settings for posts (e.g., public, private, friends-only).

Likes and Comments:

Enable users to like and comment on posts.

Display the number of likes and comments for each post.

Allow users to delete their own comments.

Following and Followers:

Implement the ability to follow and unfollow other users.

Provide lists of followers and users followed by each user.

Notifications:

Send notifications to users for likes, comments, and new followers.

Allow users to manage notification settings.

Search and Discovery:

Implement user search functionality.

Suggest friends or posts based on user activity and interests.

Security:

Implement security measures to protect against common vulnerabilities (e.g., input validation, XSS, CSRF).

Use password hashing for user authentication.

Testing:

Write unit tests for critical components of the application using JUnit.

Implement integration tests with Mockito for testing API endpoints.

Logging and Monitoring:

Implement logging to track application events and errors.

Set up monitoring for application performance and error detection.

API Documentation:

Create comprehensive API documentation using Swagger to document API endpoints and usage.

Deployment:

Deploy the API to an application server or cloud platform (e.g., AWS, Heroku).

Containerize the application using Docker.

Set up continuous integration and continuous deployment (CI/CD) pipelines.

List of APIs to Create:

User Authentication API:

/register: Register a new user.

/login: Log in a user and return a JWT token.

/reset-password: Initiate a password reset process.

User Profile API:

/user/{userId}: Retrieve user profile information.

/user/{userId}/update: Update user profile information.

/user/{userId}/followers: Get followers of a user.

/user/{userId}/following: Get users followed by a user.

Posting and Feeds API:

/posts: Get a list of posts.

/posts/{postId}: Get a specific post by ID.

/posts/create: Create a new post.

/posts/{postId}/update: Update a post.

/posts/{postId}/delete: Delete a post.

/feed: Get the user's news feed.

Likes and Comments API:

/posts/{postId}/like: Like a post.

/posts/{postId}/unlike: Unlike a post.

/posts/{postId}/comment: Add a comment to a post.

/comments/{commentId}/delete: Delete a comment.

Following and Followers API:

/user/{userId}/follow: Follow a user.

/user/{userId}/unfollow: Unfollow a user.

Notifications API:

/notifications: Get user notifications.

/notifications/settings: Manage notification settings.

Search and Discovery API:

/user/search: Search for users.

/suggestions: Get friend or post suggestions.

Privacy Settings API:

/user/{userId}/privacy: Set privacy preferences for a user's posts and profile.

By building this project, you'll demonstrate your proficiency in creating a full-fledged social media platform API with Spring Boot, Hibernate, JPA, Swagger, and thorough testing using JUnit and Mockito. This project can be a standout addition to your portfolio as a Java developer.

Database Model :

Designing a database model for a Social Media Platform API involves defining the tables and relationships that store the data required for the project. Here's a simplified database model for your project. Please note that you can expand or customize this model based on your specific requirements and business logic:

Entities and Tables:

User:

user\_id (Primary Key)

username

email

password\_hash

profile\_picture\_url

bio

created\_at

updated\_at

Post:

post\_id (Primary Key)

user\_id (Foreign Key referencing User)

content

privacy (e.g., public, private, friends-only)

created\_at

updated\_at

Like:

like\_id (Primary Key)

user\_id (Foreign Key referencing User)

post\_id (Foreign Key referencing Post)

created\_at

Comment:

comment\_id (Primary Key)

user\_id (Foreign Key referencing User)

post\_id (Foreign Key referencing Post)

content

created\_at

Follower:

follower\_id (Primary Key)

user\_id (Foreign Key referencing User)

follower\_user\_id (Foreign Key referencing User)

Notification:

notification\_id (Primary Key)

user\_id (Foreign Key referencing User)

content

created\_at

Tag:

tag\_id (Primary Key)

name

PostTag (Many-to-Many Relationship Table for Posts and Tags):

post\_tag\_id (Primary Key)

post\_id (Foreign Key referencing Post)

tag\_id (Foreign Key referencing Tag)

Relationships:

Each user can have multiple posts (One-to-Many relationship: User to Post).

Each user can have multiple likes (One-to-Many relationship: User to Like).

Each user can have multiple comments (One-to-Many relationship: User to Comment).

Each user can have multiple followers (Self-referencing Many-to-Many relationship: User to Follower).

Each user can receive multiple notifications (One-to-Many relationship: User to Notification).

Each post can have multiple tags (Many-to-Many relationship: Post to PostTag).

Each post can have multiple likes (One-to-Many relationship: Post to Like).

Each post can have multiple comments (One-to-Many relationship: Post to Comment).

Note:

The created\_at and updated\_at fields are used to track when records were created and last updated.

You may need to add additional fields or tables to meet specific project requirements, such as user settings, privacy settings, and search-related data.

This is a basic starting point for your database model. Depending on the complexity of your project and additional features you want to implement, you may need to further refine and expand the database schema. Always consider best practices for database design and normalization to ensure data integrity and performance.