Project Title: Interactive Social Media Forum

Functional and Non-Functional Requirements

Functional Requirements:

1. **User Authentication and Profiles**:
    - Users can register an account.
    - Users can log in and log out.
    - Users can view and edit their profile information.
    - Users can view other users' profiles.
2. **Post Creation and Management**:
    - Users can create a new post with text (optional: add images or videos).
    - Users can edit or delete their own posts.
    - Users can view a list of all posts.
    - Posts can be organized by tags and categories.
3. **Comments and Interactions**:
    - Users can comment on posts.
    - Users can reply to comments (nested comments).
    - Users can like and dislike posts and comments.
4. **User Interaction and Networking**:
    - Users can follow/unfollow other users.
    - Users can send direct messages to other users.
    - Users receive notifications for likes, comments, follows, and messages.
5. **Moderation and Administration**:
    - Admins can manage users and content.
    - Users can report inappropriate content.
    - Role-based access control for admin, moderator, and user roles.

Non-Functional Requirements:

1. **Performance**:
    - The system should handle up to 1000 concurrent users without significant performance degradation.
    - Response time for any action should be less than 2 seconds.
2. **Scalability**:
    - The system should be scalable to accommodate future growth in the number of users and posts.
3. **Security**:
    - User data should be encrypted.
    - Authentication should be handled securely using Spring Security.
    - Input validation should be enforced to prevent SQL injection and other attacks.
4. **Usability**:
    - The user interface should be intuitive and easy to navigate.
    - The system should be accessible on both desktop and mobile devices.
5. **Maintainability**:

    - The codebase should follow best practices for readability and maintainability.
    - Documentation should be provided for the code and API.
6. **Reliability**:
    - The system should have an uptime of 99.9%.
    - Data integrity should be maintained, ensuring no data loss during transactions.

Use Cases

Use Case 1: User Registration

Actor: User
Description: A user registers for a new account.
Steps:

1. User navigates to the registration page.
2. User fills out the registration form with username, email, and password.
3. User submits the registration form.
4. System validates the input.
5. System creates a new user account and stores it in the database.
6. System sends a confirmation email to the user.
7. User confirms their email.

Use Case 2: Create a Post

Actor: Authenticated User
Description: A user creates a new post.
Steps:

1. User logs in.
2. User navigates to the create post page.
3. User enters the text for the post .
4. User submits the post.
5. System validates the input.
6. System saves the post to the database.
7. System displays the post in the user's feed.

Use Case 3: Comment on a Post

Actor: Authenticated User
Description: A user comments on an existing post.
Steps:

1. User logs in.
2. User navigates to the post they want to comment on.
3. User enters a comment.
4. User submits the comment.
5. System validates the input.
6. System saves the comment to the database.

7. System displays the comment under the post.

Use Case 4: Like a Post

Actor: Authenticated User
Description: A user likes an existing post.
Steps:

1. User logs in.
2. User navigates to the post they want to like.
3. User clicks the like button.
4. System updates the like count for the post.
5. System records the like in the database.

Use Case 5: View User Profile

Actor: Any User
Description: A user views another user's profile.
Steps:

1. User navigates to the profile page of another user.
2. System retrieves the profile information from the database.
3. System displays the profile information, including posts and comments.

Objects, Classes, and Relationships

Entities:

User:

- `id` (serial)
- `username` (varchar)
- `email` (varchar)
- `password` (varchar)

Post:

- `id` (serial)
- `author_id` (foreign key referencing User)
- `text` (varchar)
- `created_at` (timestamp)

Comment:

- `id` (serial)
- `post_id` (foreign key referencing Post)
- `author_id` (foreign key referencing User)
- `text` (varchar)

- `created_at` (timestamp)

Like:

- `id` (serial)
- `liker_id` (foreign key referencing User)
- `post_id` (foreign key referencing Post)

Class Diagram (Textual Representation):

```
User
+ id: Long
+ username: String
+ email: String
+ password: String
+ posts: List<Post>
+ comments: List<Comment>
+ likes: List<Like>

Post
+ id: Long
+ author: User
+ text: String
+ createdAt: LocalDateTime
+ comments: List<Comment>
+ likes: List<Like>

Comment
+ id: Long
+ post: Post
+ author: User
+ text: String
+ createdAt: LocalDateTime

Like
+ id: Long
+ liker: User
+ post: Post
```

CRC Cards (Class-Responsibility-Collaborator)

User:

- Responsibilities:
    - Register and log in.
    - Manage profile information.
    - Create posts and comments.
    - Like posts and comments.
- Collaborators:
    - Post
    - Comment
    - Like

Post:

- Responsibilities:
    - Store post content.
    - Link to the author.
    - Manage comments and likes.
- Collaborators:
    - User
    - Comment
    - Like

Comment:

- Responsibilities:
    - Store comment content.
    - Link to the post and author.
- Collaborators:
    - Post
    - User

Like:

- Responsibilities:
    - Store like information.
    - Link to the liker and post.
- Collaborators:
    - User
    - Post

Summary

This document outlines the high-level overview, functional and non-functional requirements, use cases, and class structure for the Interactive Social Media Forum project. The design ensures a structured approach to building a user-friendly and secure platform for social interactions, supporting various features such as user authentication, post management, commenting, liking, and user networking.