Logical Level Design (LLD) - User Management and Discussion Application

1. Components Overview

- **Entities**:
  - `User`: Represents a registered user with attributes like name, email, and discussions.
  - `Discussion`: Represents a discussion post with attributes like text, image, hashtags, and created timestamp.
- **Repositories**:
  - `UserRepository`: Handles database operations related to users.
  - `DiscussionRepository`: Handles database operations related to discussions.
- **Services**:
  - `UserService`: Provides business logic related to users (e.g., CRUD operations).
  - `DiscussionService`: Provides business logic related to discussions (e.g., CRUD operations).
- **Controllers**:
  - `UserController`: Handles HTTP requests related to users.
  - `DiscussionController`: Handles HTTP requests related to discussions.
- **Security**:
  - `JwtAuthenticationFilter`: Filter to handle JWT authentication and authorization.
  - `SecurityConfig`: Configuration class for Spring Security.

2. Flow of Operations

User Management Flow

1. **Create User**:
   - **Request**: HTTP POST to `/api/users`
   - **Controller**: `UserController` receives the request.
   - **Service**: `UserService` validates and processes the request.
   - **Repository**: `UserRepository` saves the user to the database.
2. **Update User**:
   - **Request**: HTTP PUT to `/api/users/{id}`
   - **Controller**: `UserController` receives the request.
   - **Service**: `UserService` retrieves and updates the user.
   - **Repository**: `UserRepository` updates the user in the database.
3. **Delete User**:
   - **Request**: HTTP DELETE to `/api/users/{id}`

    - **Controller**: `UserController` receives the request.
    - **Service**: `UserService` deletes the user and associated discussions.
    - **Repository**: `UserRepository` deletes the user from the database.
4. **Get All Users**:
    - **Request**: HTTP GET to `/api/users`
    - **Controller**: `UserController` retrieves all users.
    - **Service**: `UserService` retrieves users from `UserRepository`.
    - **Response**: List of users serialized to JSON.

Discussion Management Flow

1. **Create Discussion**:
    - **Request**: HTTP POST to `/api/discussions`
    - **Controller**: `DiscussionController` receives the request.
    - **Service**: `DiscussionService` validates and processes the request.
    - **Repository**: `DiscussionRepository` saves the discussion to the database.
2. **Update Discussion**:
    - **Request**: HTTP PUT to `/api/discussions/{id}`
    - **Controller**: `DiscussionController` receives the request.
    - **Service**: `DiscussionService` retrieves and updates the discussion.
    - **Repository**: `DiscussionRepository` updates the discussion in the database.
3. **Delete Discussion**:
    - **Request**: HTTP DELETE to `/api/discussions/{id}`
    - **Controller**: `DiscussionController` receives the request.
    - **Service**: `DiscussionService` deletes the discussion.
    - **Repository**: `DiscussionRepository` deletes the discussion from the database.
4. **Get Discussions by User**:
    - **Request**: HTTP GET to `/api/discussions/user/{userId}`
    - **Controller**: `DiscussionController` retrieves discussions by user ID.
    - **Service**: `DiscussionService` retrieves discussions from `DiscussionRepository`.
    - **Response**: List of discussions serialized to JSON.
5. **Search Discussions by Hashtag**:
    - **Request**: HTTP GET to `/api/discussions/tags?hashtag={hashtag}`
    - **Controller**: `DiscussionController` retrieves discussions by hashtag.
    - **Service**: `DiscussionService` retrieves discussions from `DiscussionRepository`.
    - **Response**: List of discussions serialized to JSON.

3. Security Considerations

- **JWT Authentication**: Implemented using `JwtAuthenticationFilter` and `SecurityConfig` to secure API endpoints.
- **Authorization**: Ensure appropriate roles and permissions are enforced for sensitive operations.

4. Error Handling

- **Global Exception Handling**: Implement `@ControllerAdvice` to handle exceptions uniformly across the application.