**### Project Structure:**

```

/blog

├── src

| ├── main

| | ├── java

| | | ├── com.yourcompany.blog

| | | | ├── controller

| | | | | ├── AuthController.java

| | | | | ├── BlogPostController.java

| | | | | └── UserController.java

| | | ├── model

| | | | ├── User.java

| | | | ├── BlogPost.java

| | | | └── Comment.java

| | | ├── repository

| | | | ├── UserRepository.java

| | | | ├── BlogPostRepository.java

| | | | └── CommentRepository.java

| | | ├── service

| | | | ├── AuthService.java

| | | | ├── BlogPostService.java

| | | | ├── UserService.java

| | | | └── NotificationService.java

| | | ├── BlogApplication.java

| | └── resources

| | └── application.properties

└── lib

└── target

└── web

└── index.html

```

**### Class Descriptions:**

1. \*\*AuthController.java:\*\*

- Handles user registration, login, and email verification.

2. \*\*BlogPostController.java:\*\*

- Manages endpoints related to blog posts, such as viewing, searching, liking, sharing, commenting, writing, editing, and deleting.

3. \*\*UserController.java:\*\*

- Handles user-related operations, including managing user profiles and notifications.

4. \*\*User.java, BlogPost.java, Comment.java:\*\*

- Represents the entities with fields like username, email, password, blog post title, content, etc.

5. \*\*UserRepository.java, BlogPostRepository.java, CommentRepository.java:\*\*

- Interfaces defining database operations on User, BlogPost, and Comment entities.

6. \*\*AuthService.java, BlogPostService.java, UserService.java, NotificationService.java:\*\*

- Service classes implementing business logic for authentication, blog post management, user management, and notifications.

7. \*\*BlogApplication.java:\*\*

- Main class to run the application.

8. \*\*index.html:\*\*

- Frontend file to display the homepage and interact with the user.

### Technologies Used:

- \*\*Java:\*\* For backend logic.

- \*\*MySQL:\*\* As the relational database.

- \*\*HTML/CSS/JavaScript:\*\* For frontend development.

You'll need to manually handle HTTP requests and responses, implement your own routing, and manage database connections. This approach can be more complex compared to using a framework, but it provides greater flexibility and control over your application. Consider using libraries for tasks such as handling HTTP requests and managing database connections to simplify your code.