**Project Structure and Setup**

1. **Maven Configuration**: Java 17, Spring Boot 3.2.0, Spring Security, JPA, PostgreSQL, and Swagger.
2. **Database Configuration**: Using PostgreSQL create a database name bookdb
3. **Swagger Integration**: API documentation accessible at /swagger-ui.html.

**Key Components**

**Entity Classes**

* **User**: Stores username, password, and roles (ADMIN or USER)
* **Book**: Contains book name, file path, upload date, active status, and a reference to the user who uploaded it

**Authentication & Security**

* **Basic Authentication**: Implemented via Spring Security
* **User Registration & Login**: Through /api/auth/register and /api/auth/login endpoints
* **Role-Based Access Control**: Controls access to different API endpoints

**User Management**

* **User Registry**: New users can register with a username and password
* **Admin Controls**: Admins can view, update roles, and delete users

**Book Management**

* **Book Upload**: Users can upload books with a name and associated file
* **Book Listing**: Users can get a list of all active books
* **Book Search**: Search functionality by book name
* **Book Reading**: Ability to download and read a specific book
* **Book Status Management**: Admins can activate/deactivate books

**Additional Features**

* **Data Initialization**: Creates default admin and user accounts on startup
* **File Upload Configuration**: Handles file storage for uploaded books
* **Global Exception Handling**: Provides proper error responses for various exceptions

**API Endpoints Summary**

**Auth**

* POST /api/auth/register - Register a new user
* POST /api/auth/login - User login (handled by Spring Security)

**User Management (Admin)**

* GET /api/admin/users - Get all users
* GET /api/admin/users/{id} - Get user by ID
* DELETE /api/admin/users/{id} - Delete a user
* PUT /api/admin/users/{id}/roles - Update user roles

**Book Management**

* GET /api/books - Get all active books
* GET /api/books/search?keyword=... - Search for books by name
* GET /api/books/{id} - Get book details by ID
* GET /api/books/{id}/read - Read/download a book
* POST /api/books - Upload a new book (multipart form data)

**Book Management (Admin)**

* PUT /api/admin/books/{id}/status?active=true|false - Activate or deactivate a book

**Running the Application**

1. Set up PostgreSQL with a database named bookdb
2. Build and run the application using Maven: mvn spring-boot:run
3. Access Swagger UI at: http://localhost:8080/swagger-ui.html
4. Default user credentials:
   * Admin: username=admin, password=admin
   * User: username=user, password=user