Secure Library Management System using Spring Boot

Objective

Your task is to develop a **secure REST API** for a **Library Management System** using **Spring Boot**. The system should allow users to:

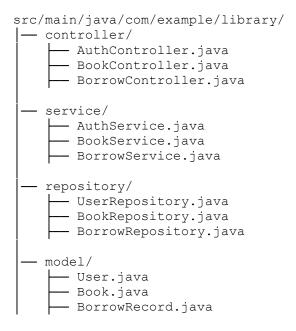
- 1. Register and log in using Basic Authentication.
- 2. Receive a JWT Bearer Token upon login.
- 3. Manage books (Admin only).
- 4. Borrow and return books (Users only).

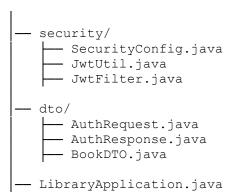
The project should follow a **layered architecture** including:

- Model Layer (Entities representing database tables)
- Repository Layer (Data access layer using Spring Data JPA)
- Service Layer (Business logic for authentication, book management, and borrowing)
- **Controller Layer** (RESTful APIs to expose functionality)
- **Security Configuration** (Handles authentication and authorization)

Project Structure

Your project must follow the structure below:





1. Model Layer

The **model layer** contains entity classes that represent database tables.

1.1. User Class

- Represents a system user (Admin or User).
- Attributes:
 - o id: Auto-incremented primary key.
 - o username: Unique and required for authentication.
 - o password: Encrypted using BCrypt hashing.
 - o role: Enum (ADMIN or USER) to differentiate privileges.

Constraints:

- o Admins can manage books (Add, Update, Delete).
- **o** Users can borrow and return books.
- o Only one user can borrow a book at a time.

1.2. Book Class

- Represents a book in the library.
- Attributes:
 - o id: Auto-incremented primary key.
 - o title: Name of the book.
 - o author: Author of the book.
 - o isBorrowed: Boolean flag indicating if the book is currently borrowed.
- Constraints:
 - o Admins can add, update, or delete books.
 - Users can only view books.

1.3. BorrowRecord Class

- Represents a transaction of borrowing and returning books.
- Attributes:
 - o id: Auto-incremented primary key.
 - o user: Reference to the User who borrowed the book.
 - o book: Reference to the Book that was borrowed.
 - o borrowedAt: Timestamp of when the book was borrowed.
 - o returnedAt: Timestamp of when the book was returned.
- Constraints:
 - o A book cannot be borrowed if it is already borrowed.
 - Users can return books after reading.

2. Repository Layer

The **repository layer** handles database operations.

2.1. UserRepository Interface

- Fetch users by username for authentication.
- Ensure username is unique.

2.2. BookRepository Interface

- Provides methods to:
 - Fetch all books.
 - o Add, update, delete books (**Admin only**).

2.3. BorrowRepository Interface

- Provides methods to:
 - Check if a **book is currently borrowed**.
 - o Fetch all borrowed books by a user.

3. Service Layer

The **service layer** contains business logic for user authentication, book management, and borrowing.

3.1. AuthService Class

- Handles user registration and authentication.
- Methods:
 - o register (User user): Stores a new user in the database with an encrypted password.
 - o login(String username, String password):
 - Verifies credentials using Basic Authentication.
 - Generates and returns a JWT token for further API access.

3.2. BookService Class

- Handles book-related business logic.
- Methods:
 - o addBook (Book book): Adds a new book (Admin only).
 - o listBooks(): Returns a list of all books (Accessible to all).

3.3. BorrowService Class

- Handles borrowing and returning books.
- Methods:
 - o borrowBook(Long userId, Long bookId):
 - Checks if the book is already borrowed.
 - Creates a new borrow record.
 - o returnBook(Long userId, Long bookId):
 - Updates the **return date** for a borrowed book.
 - Marks the book as available.

4. Controller Layer

The **controller layer** exposes REST APIs.

4.1. AuthController Class

- Exposes authentication endpoints.
- Endpoints:

- o POST /auth/register \rightarrow Registers a new user.
- o POST /auth/login → Authenticates a user and returns a JWT token.

4.2. BookController Class

- Exposes book management endpoints.
- Endpoints:
 - o POST /books/add \rightarrow Admin-only \rightarrow Adds a new book.
 - o GET /books/list → All users → Lists all books.

4.3. BorrowController Class

- Exposes borrow and return endpoints.
- Endpoints:
 - o POST /borrow/{bookId} \rightarrow User-only \rightarrow Borrows a book.
 - o POST /return/{bookId} \rightarrow User-only \rightarrow Returns a borrowed book.

5. Security Layer

The **security layer** handles authentication.

5.1. SecurityConfig Class

- Implements **Basic Authentication for login**.
- Implements JWT-based Bearer Token authentication for protected routes.

5.2. JwtUtil Class

- Generates **JWT tokens**.
- Validates **JWT tokens**.

5.3. JwtFilter Class

- Intercepts requests.
- Extracts and validates **JWT tokens**.

Testing Endpoints

User Registration

```
POST /auth/register
{
    "username": "admin",
    "password": "admin123",
    "role": "ADMIN"
}
```

Login to get JWT Token

```
POST /auth/login
{
   "username": "admin",
    "password": "admin123"
}
```

Add Book (Admin)

```
POST /books/add
Authorization: Bearer <TOKEN>
{
   "title": "Spring Boot Guide",
   "author": "John Doe"
}
```

List Books (User)

```
GET /books/list
Authorization: Bearer <TOKEN>
```

Borrow a Book

```
POST /borrow/1
Authorization: Bearer <TOKEN>
```

Return a Book

```
POST /return/1
Authorization: Bearer <TOKEN>
```

Final Notes

- 1. Use BCrypt to encrypt passwords.
- 2. Ensure JWT tokens are required for protected routes.
- 3. Restrict book management to ADMINs.

4. Users can borrow but cannot borrow an already borrowed book.

This is a fully detailed Spring Boot final exam question that tests authentication, authorization, REST APIs, and role-based access control.