TOPIC :
LIBRARY BORROWING AND BOOK TRACKING SYSTEM

Lecturer: Mr. ROEUN Pacharoth

Presented by: I4-Group-B

Team members

Huy Chanchhinghoir

Thoung Soket

Im Chheangngim

Thou Thearo

Sophal Mengchhiv

TABLE OF CONTENT

PROJECT OVERVIEW

The Library Management System is a web-based application that helps manage books, users, and borrowing records. It replaces manual library work with a digital system, making book management easier and more organized.

PROJECT OBJECTIVE

- Develop a basic Library Management System using Spring Boot
- Manage books, users, and borrowing records
- Apply MVC architecture for better code organization
- Design a clear and structured database
- Provide a simple and user-friendly interface

PLAN OF PROJECT

Requirement Analysis

- Identify user roles and system features
- Define borrowing rules and system scope

System Design

- Design database and UI layout
- Plan system architecture

Implementation

- Develop frontend and backend
- Integrate database and APIs

Testing

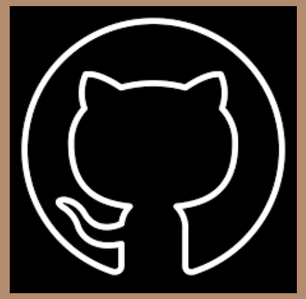
- Test system functions
- Fix bugs

TECHNOLOGY TOOLS

Java

Springboot

Gradle



Github

MySQL

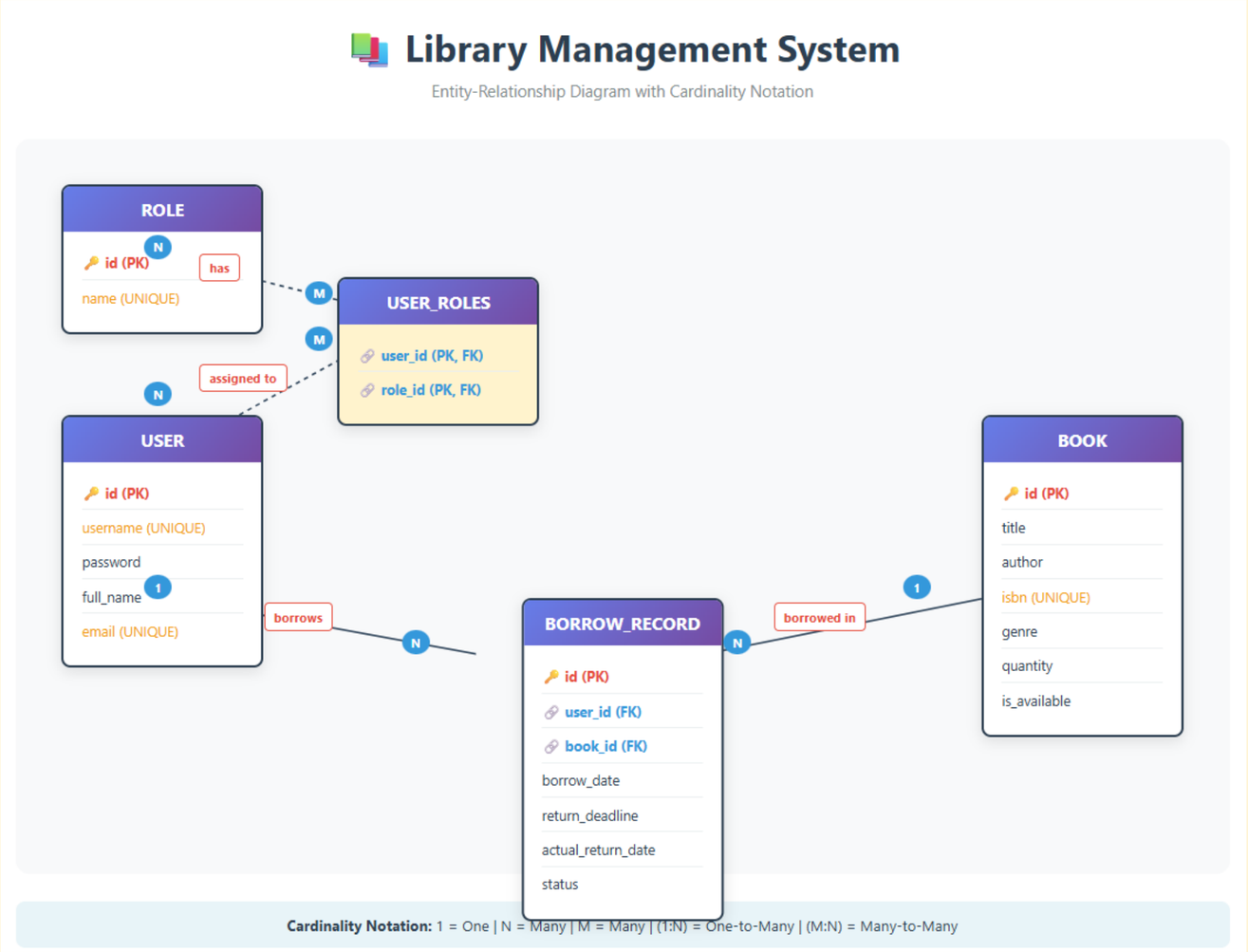
Git

DEPENDENCIES

```
spring.application.name=library-management-system

# Database Configuration
spring.datasource.url=jdbc:mysql://localhost:3306/library_db?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=Thouthearo168168168
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# JPA Configuration
spring.jpa.hibernate.ddl-auto=validate
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

# Flyway Configuration
spring.flyway.enabled=true
spring.flyway.locations=classpath:static/migration
spring.flyway.baseline-on-migrate=true

# Thymeleaf Configuration
spring.thymeleaf.cache=false
```

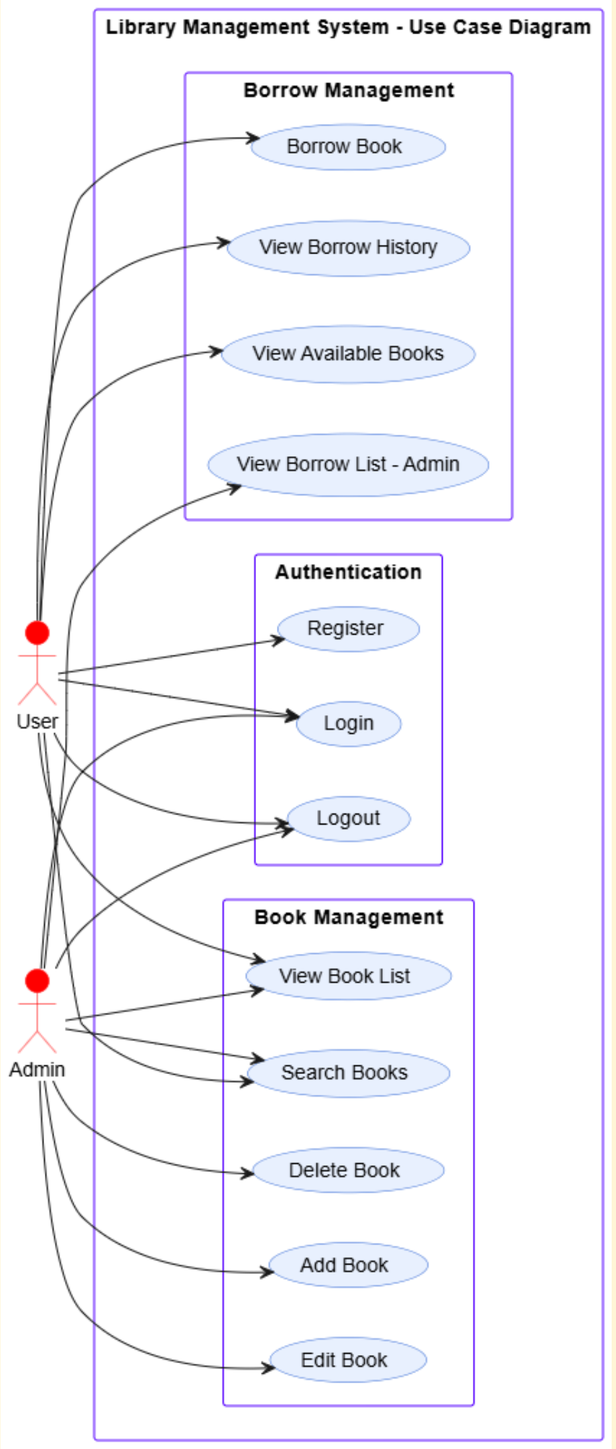
DIAGRAM

ER Diagram



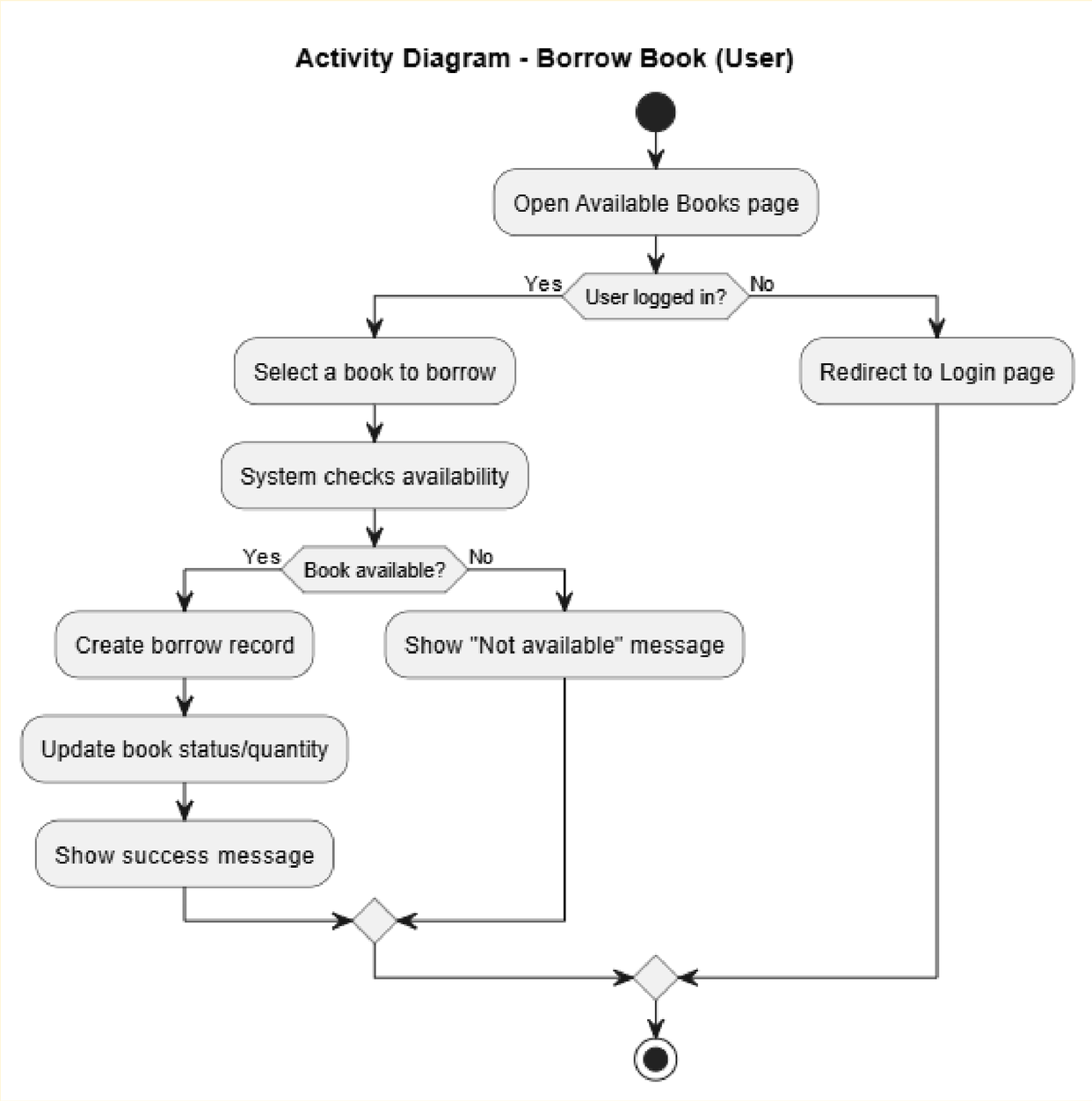
DIAGRAM

Use case diagram



DIAGRAM

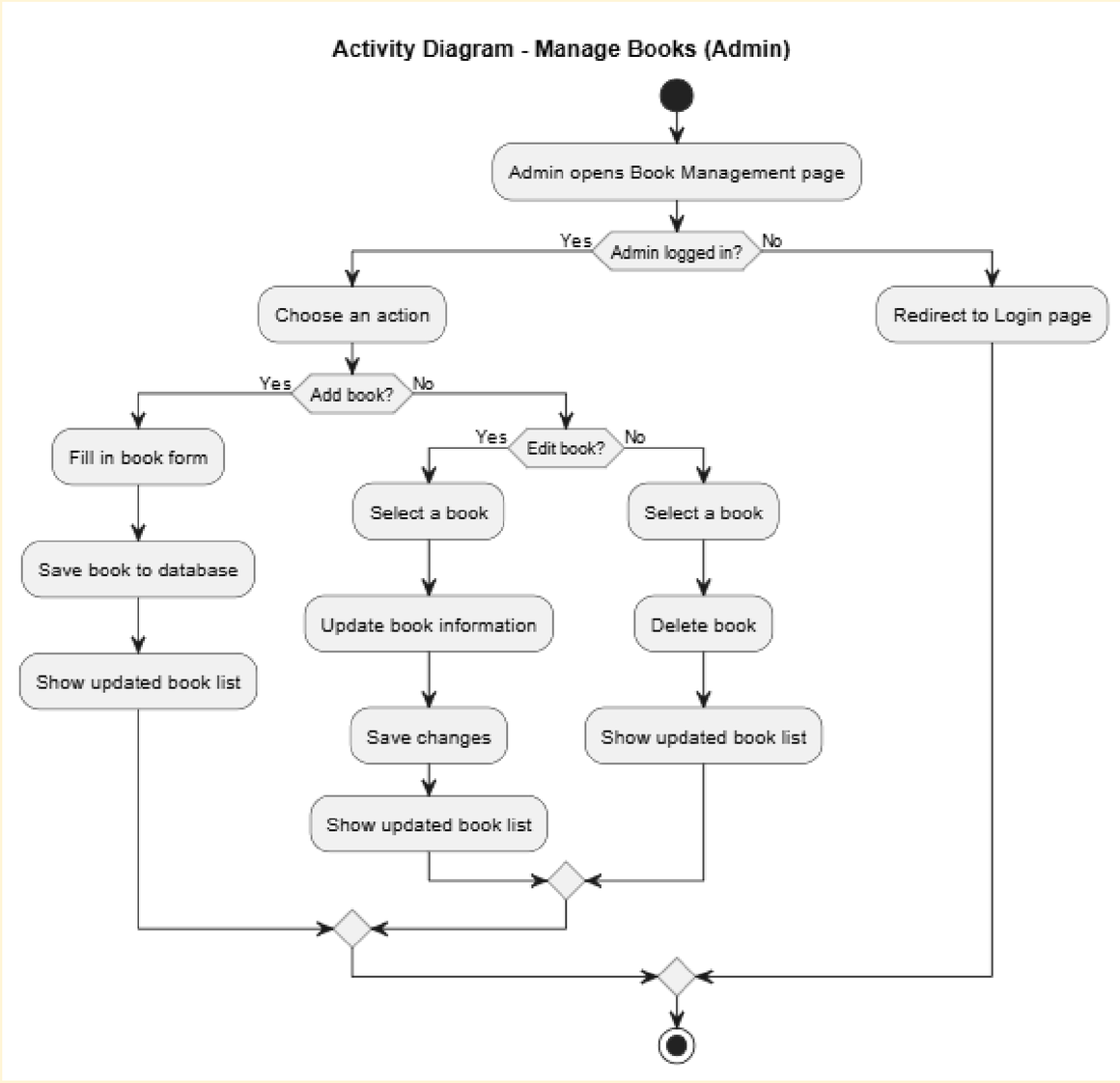
Activity diagram
Borrow Book (User Flow)



DIAGRAM

Activity diagram

Manage Books (Admin Flow)



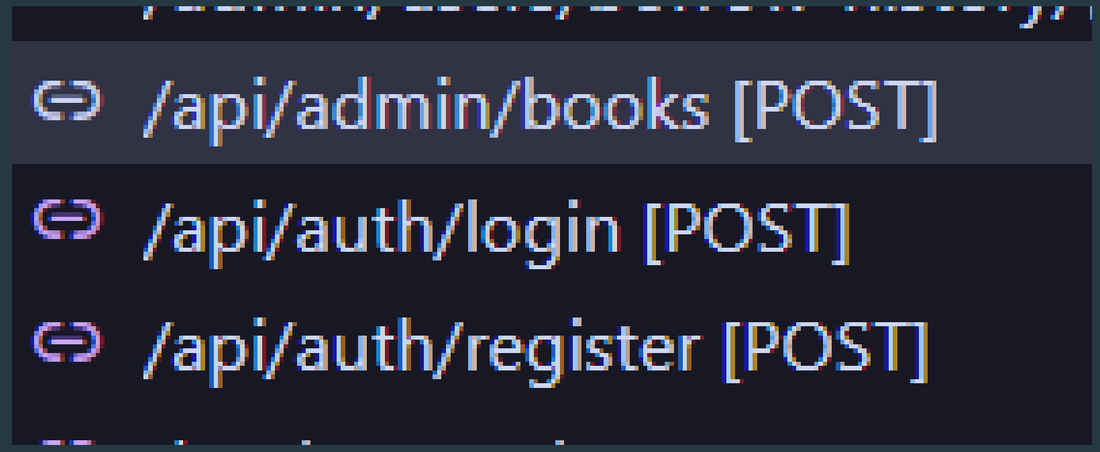
API Endpoints

Purpose

- Provide REST APIs for authentication and book management
    - Used for communication between client tools (Postman) and backend

Main Endpoints

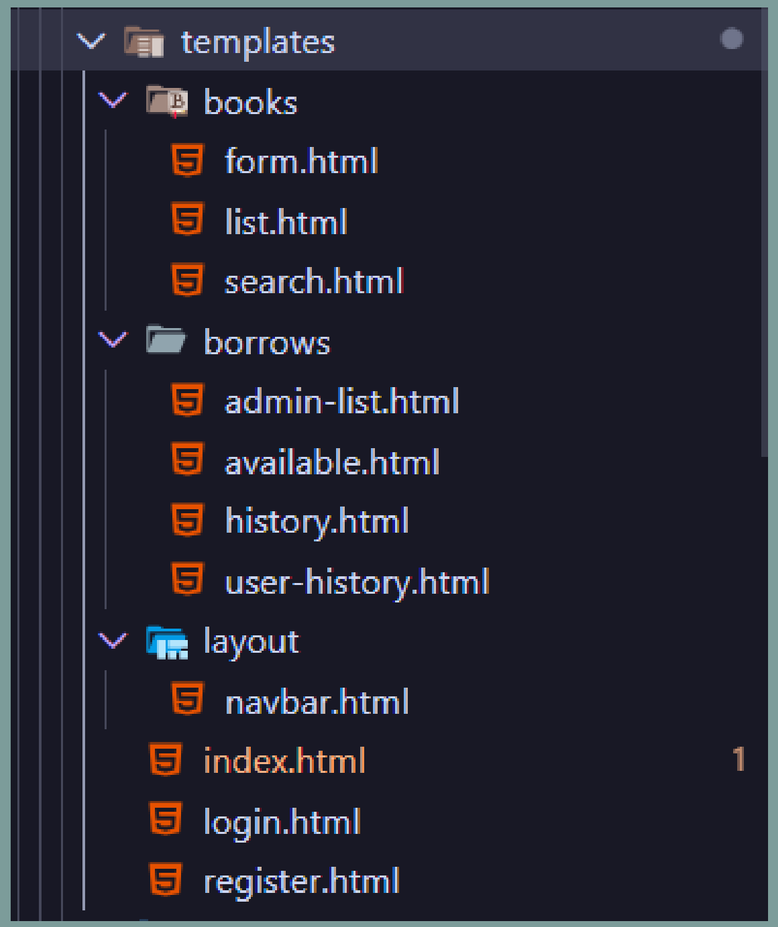
- POST /api/auth/register – register new users
- POST /api/auth/login – authenticate users
- POST /api/admin/books – add new books (admin only)



IMPLEMENTATION

- FRONTEND

Folder structure



- **BACKEND**

Folder structure :

- **Controller : Handle HTTP requests and responses .**

  Example: AuthController (AuthController.java)

  - **GET /login - Display login form**
  - **GET /register - Display registration form**
  - **POST /register - Process user registration**

- **Config : set up how application works and related to security**

  Example: SecurityConfig.java

  - **Encrypting passwords**
  - **Validating JWT tokens**
  - **Controlling access by role**
  - **Managing login/logout**
  - **Blocking unauthorized requests**

- **Security : JWT token-based authentication and authorization**

Example: JwtTokenProvider (JwtTokenProvider.java)

- **generateToken(authentication) - Create JWT after login**
- **validateToken(token) - Verify token signature and expiration**

- **Entity : represented table in databases;**

Example : User Entity (User.java)

Attributes:

- id: Long (Primary Key)
- username: String (Unique)
- password: String (BCrypt Encrypted)
- fullName: String
- email: String (Unique)
- roles: Set<Role> (Many-to-Many)

- **Service : Contain business logic and transaction managemen.**

  Example:BookService (BookService.java)

  - **getAllBooks() - Retrieve all books**
  - **getBookById(id) - Find specific book**
  - **getAvailableBooks() - Filter available books**
  - **saveBook(book) - Create new book record**
  - **updateBook(id, details) - Modify book information**
  - **deleteBook(id) - Remove book from inventory**
  - **searchBooks(keyword) - Search by title or author**
  - **updateAvailability(id, status) - Toggle availability**

- **Repository (Data Access Layer) : Perform database operations using Spring Data JPA**

Example BookRepository.java

- **findByIsbn(isbn) - ISBN lookup**
- **findByIsAvailable(status) - Filter available books**
- **findByTitleContainingIgnoreCaseOrAuthorContainingIgnoreCase() - Search**

DTOs: Data Transfer Objects for request/response handling

Example :BookRequest.java

```java
 * DTO for creating and updating books in the library system.
 * Includes comprehensive validation for security and data integrity.
 */
@Data
@NoArgsConstructor
@AllArgsConstructor
public class BookRequest {

    @NotBlank(message = "Title is required")
    @Size(min = 1, max = 70, message = "Title must be between 1 and 70 characters")
    @Pattern(regexp = "^[a-zA-Z0-9\\s\\-._':()&]+$", message = "Title contains invalid characters")
    private String title;

    @NotBlank(message = "Author is required")
    @Size(min = 1, max = 70, message = "Author must be between 1 and 70 characters")
    @Pattern(regexp = "^[a-zA-Z\\s\\-._',()&]+$", message = "Author contains invalid characters")
    private String author;

    @NotBlank(message = "ISBN is required")
    @Size(min = 10, max = 17, message = "ISBN must be between 10 and 17 characters")
    @Pattern(regexp = "^[0-9\\-]+$", message = "ISBN must contain only numbers and hyphens")
    private String isbn;

    @NotBlank(message = "Genre is required")
    @Size(min = 1, max = 50, message = "Genre must be between 1 and 50 characters")
    @Pattern(regexp = "^[a-zA-Z\\s\\-._',()&]+$", message = "Genre contains invalid characters")
    private String genre;

    @NotNull(message = "Quantity is required")
    @Min(value = 1, message = "Quantity must be at least 1")
    @Max(value = 3, message = "Quantity cannot exceed 10000")
    private Integer quantity;
}
```

DEMONSTRATION

CONCLUSION

This Library Management System simplifies book handling, borrowing, and user management through automation. It improves efficiency, accuracy, and user experience while strengthening our teamwork and system design skills.

THANK YOU FOR LISTENING!

Don't hesitate to ask any questions!