**Library Management System**

**Overview**

The Library Management System is a Spring Boot application designed to manage books, patrons, and borrowing records. This document provides instructions on how to run the application and interact with its API endpoints.

**Running the Application**

1. **Clone the Repository**

git clone https://github.com/your-repo/library-management.git

cd library-management

1. **Configure the Application**

Update the src/main/resources/application.properties file with your database and server settings:

spring.application.name=library-management

spring.datasource.url=jdbc:mysql://localhost:3306/library\_db

spring.datasource.username=your-username

spring.datasource.password=your-password

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect

server.port=8085

spring.cache.type=simple

1. **Build the Application**

Using Maven:

mvn clean install

1. **Run the Application**

mvn spring-boot:run

The application will start on http://localhost:8085.

**Testing**

To run the tests, use the following Maven command:

mvn test

**Books API**

1. **Get All Books**
   * **Method**: GET
   * **URL**: http://localhost:8085/api/books
   * **Description**: Retrieves all books.
2. **Get Book by ID**
   * **Method**: GET
   * **URL**: http://localhost:8085/api/books/{id}
   * **Description**: Retrieves a book by its ID.
3. **Add a New Book**
   * **Method**: POST
   * **URL**: http://localhost:8085/api/books
   * **Body**:

{

"title": "New Book",

"author": "New Author",

"publicationYear": 2024,

"count": 10,

"isbn": "1234567890123"

}

* + **Description**: Adds a new book to the library.

1. **Update a Book**
   * **Method**: PUT
   * **URL**: http://localhost:8085/api/books/{id}
   * **Body**:

{

"title": "Updated Title",

"author": "Updated Author",

"publicationYear": 2025,

"count": 15,

"isbn": "1234567890123"

}

* + **Description**: Updates an existing book by its ID.

1. **Delete a Book**
   * **Method**: DELETE
   * **URL**: http://localhost:8085/api/books/{id}
   * **Description**: Deletes a book by its ID.

**Patrons API**

1. **Get All Patrons**
   * **Method**: GET
   * **URL**: http://localhost:8085/api/patrons
   * **Description**: Retrieves all patrons.
2. **Get Patron by ID**
   * **Method**: GET
   * **URL**: http://localhost:8085/api/patrons/{id}
   * **Description**: Retrieves a patron by their ID.
3. **Add a New Patron**
   * **Method**: POST
   * **URL**: http://localhost:8085/api/patrons
   * **Body**:

{

"name": "New Patron",

"contactInfo": "newpatron@example.com"

}

* + **Description**: Adds a new patron to the system.

1. **Update a Patron**
   * **Method**: PUT
   * **URL**: http://localhost:8085/api/patrons/{id}
   * **Body**:

{

"name": "Updated Patron",

"contactInfo": "updatedpatron@example.com"

}

* + **Description**: Updates an existing patron by their ID.

1. **Delete a Patron**
   * **Method**: DELETE
   * **URL**: http://localhost:8085/api/patrons/{id}
   * **Description**: Deletes a patron by their ID.

**Borrowing Records API**

1. **Get All Borrowing Records**
   * **Method**: GET
   * **URL**: http://localhost:8085/api/borrowingRecords
   * **Description**: Retrieves all borrowing records.
2. **Borrow a Book**
   * **Method**: POST
   * **URL**: http://localhost:8085/api/borrow/{bookId}/patron/{patronId}
   * **Description**: Creates a new borrowing record for a book and a patron.

=>Ensure you replace {id}, {bookId}, and {patronId} with actual IDs when making requests.