Library Management System

Introduction

The Library Management System is a web application for managing books, patrons, and borrowing records. It provides functionality to add, update, and delete books and patrons, as well as to borrow and return books. The application uses Spring Boot and H2 Database for simplicity and ease of testing.

Database Configuration

The application uses an H2 in-memory database. The database is pre-configured and does not require any external setup.

Accessing H2 Console

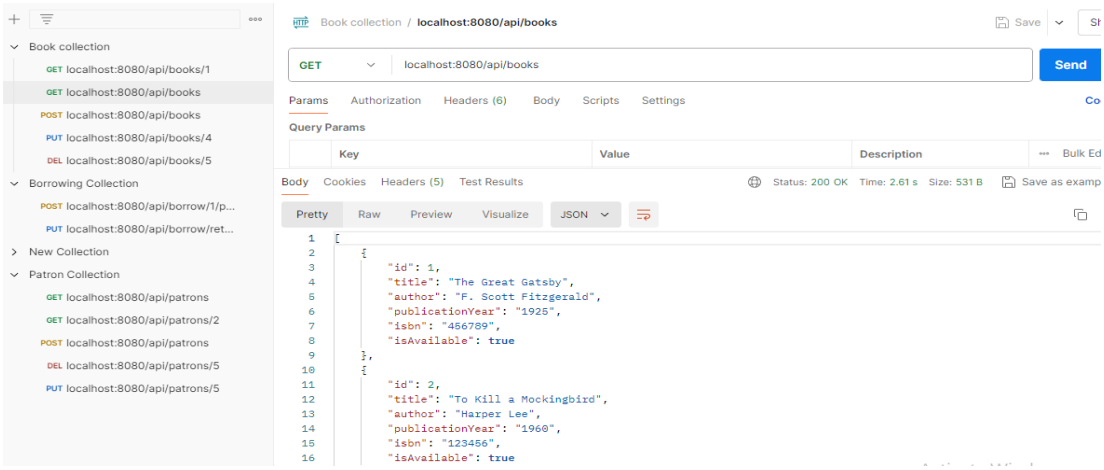
The H2 console is available at http://localhost:8080/h2-console. The default JDBC URL is jdbc:h2:mem:testdb, and the default username and password are sa and empty, respectively.

API Endpoints

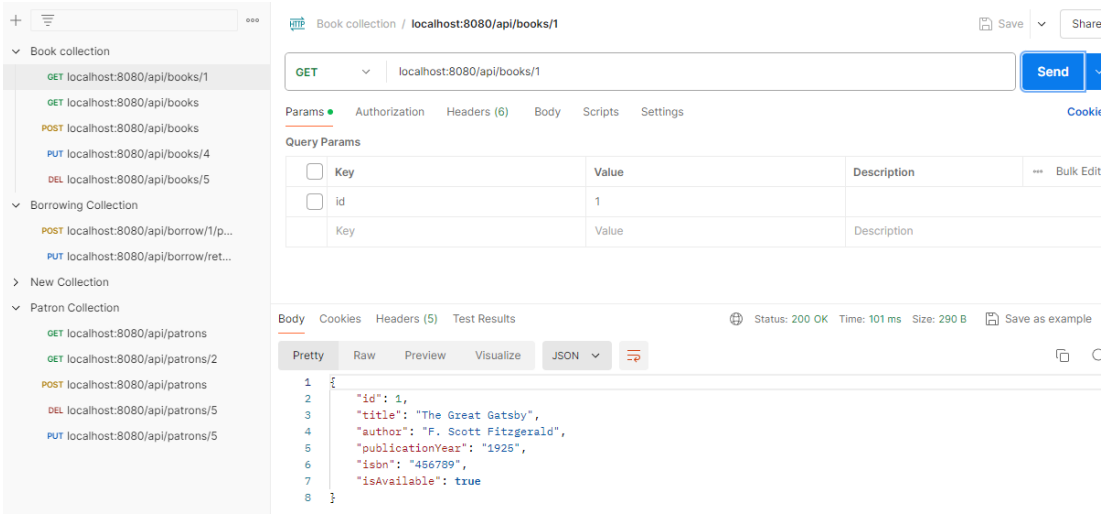
The application provides the following API endpoints:

Books

- **Get All Books**
  - **URL**: /api/books
  - **Method**: GET
  - **Response**: A list of all books in the system.

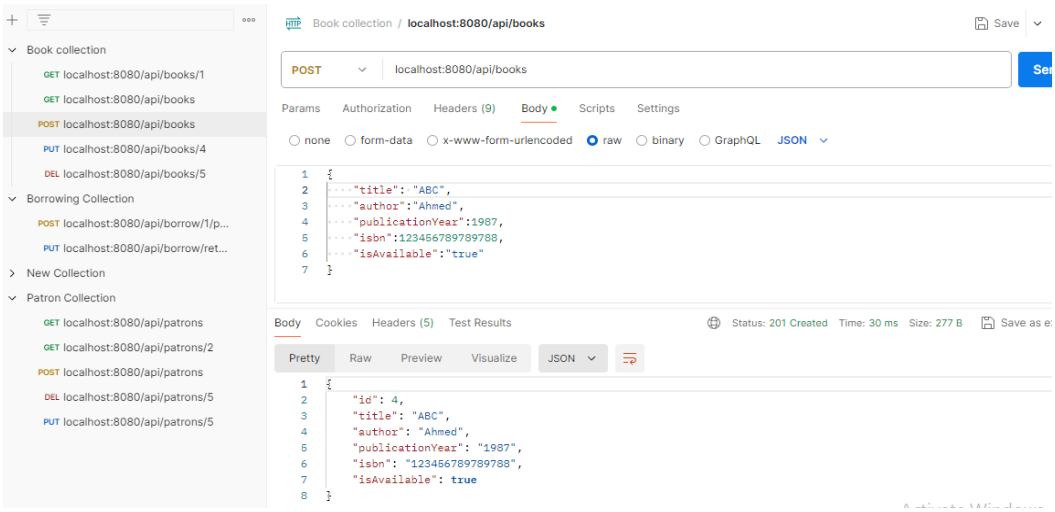


- **Get Book by ID**
    - **URL**: /api/books/{id}
    - **Method**: GET
    - **URL Parameters**: id (Book ID)
    - **Response**: Details of the book with the specified ID.

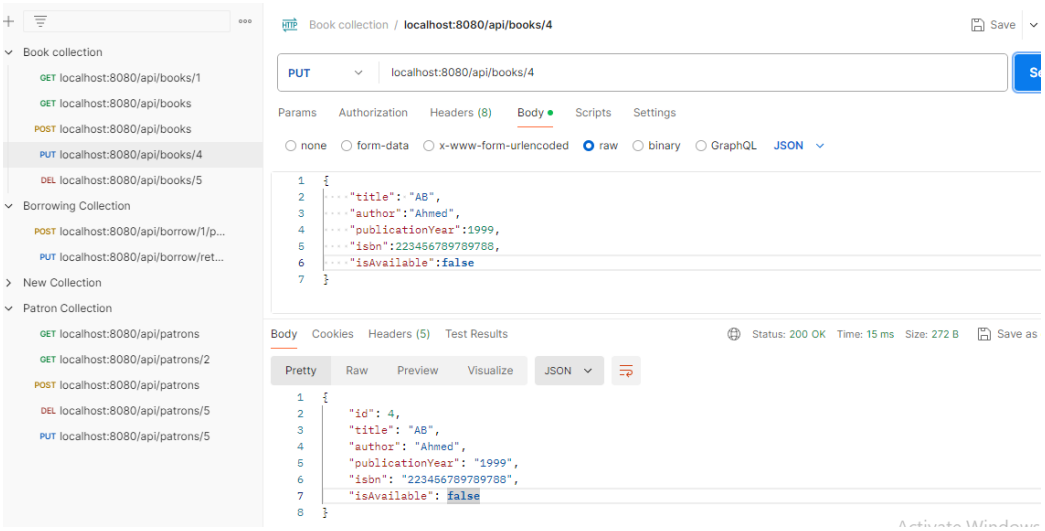


- **Add a New Book**
    - **URL**: /api/books
    - **Method**: POST
    - **Request Body**: Book object (

      { "title": "new Title", "author": "new Author", "publicationYear": "1925", "isbn": "9780743273565", "isAvailable": true })
    - **Response**: Details of the added book.

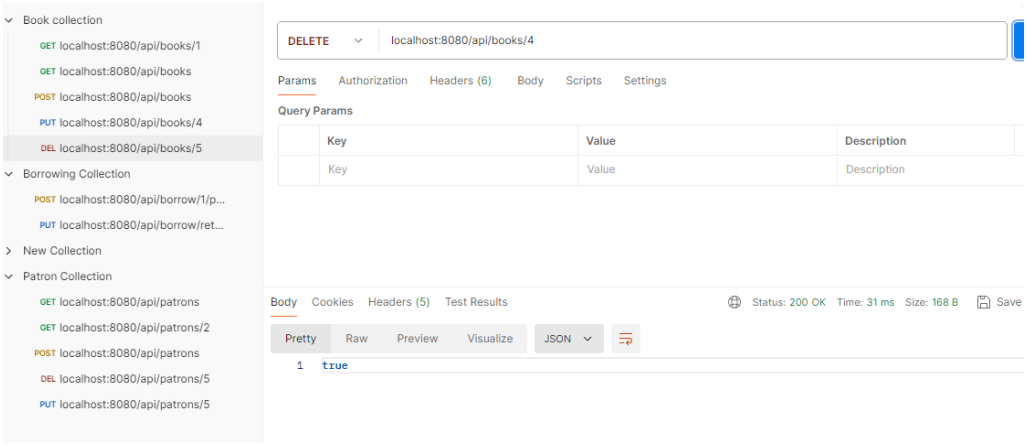


- **Update a Book**
  - **URL**: /api/books/{id}
  - **Method**: PUT
  - **URL Parameters**: id (Book ID)
  - **Request Body**: Updated Book object
  - **Response**: Details of the updated book.



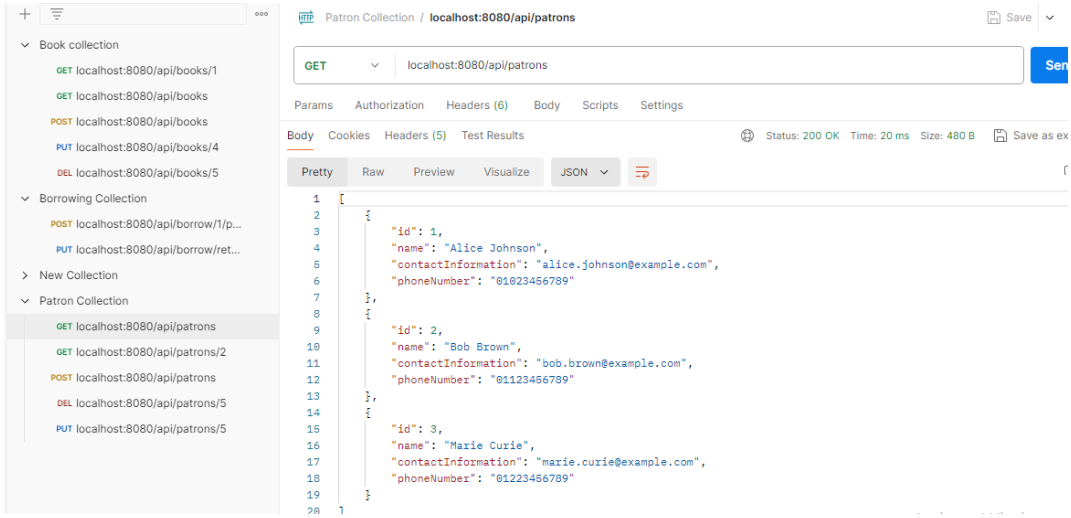
- **Delete a Book**
  - **URL**: /api/books/{id}
  - **Method**: DELETE
  - **URL Parameters**: id (Book ID)

- **Response**: Success or failure message.

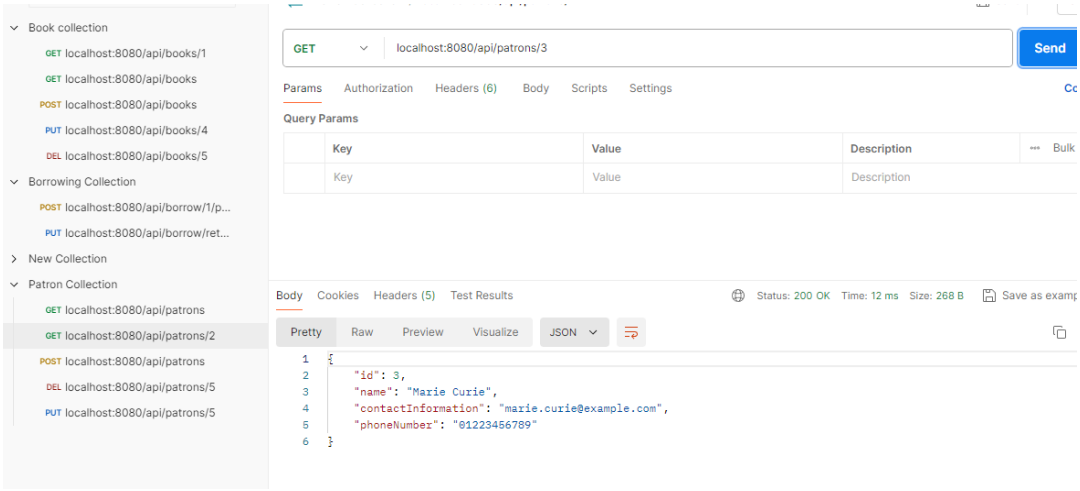


Patrons

- **Get All Patrons**
  - **URL**: /api/patrons
  - **Method**: GET
  - **Response**: A list of all patrons in the system.

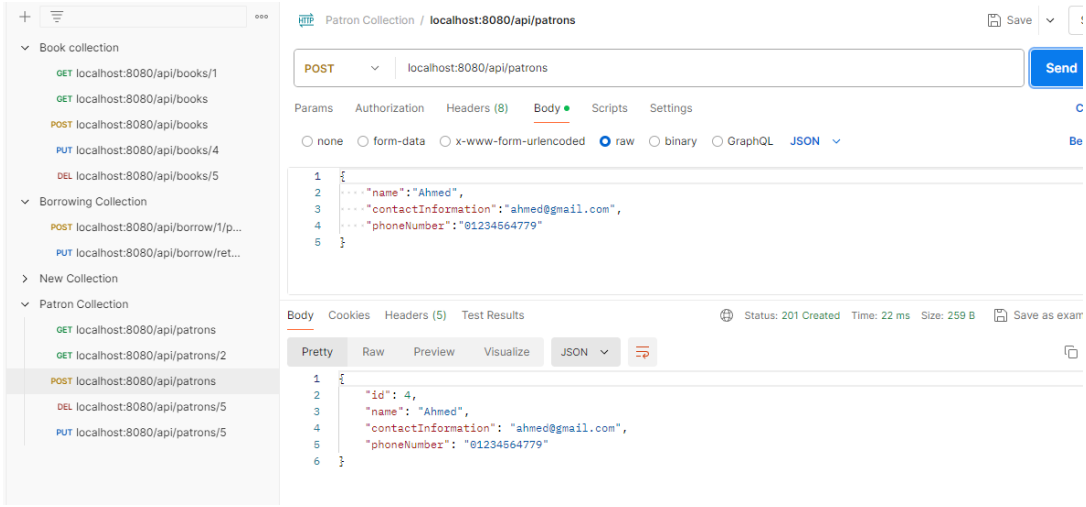


- **Get Patron by ID**
  - **URL**: /api/patrons/{id}
  - **Method**: GET
  - **URL Parameters**: id (Patron ID)
  - **Response**: Details of the patron with the specified ID.



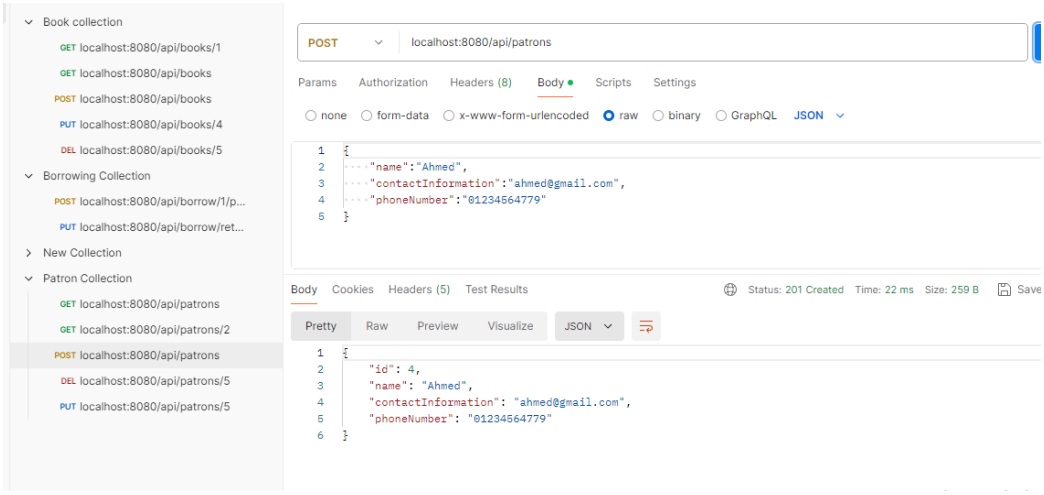
- **Add a New Patron**
  - **URL**: /api/patrons
  - **Method**: POST
  - **Request Body**: Patron object (

    { "name": "new Name", "contactInformation": "newname@example.com","phoneNumber": "10234567890" })
  - **Response**: Details of the added patron.

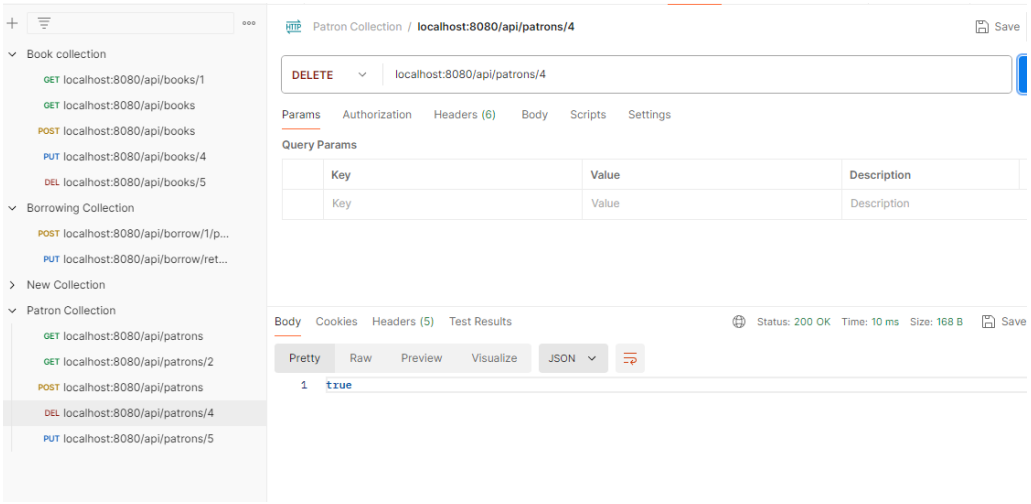


- **Update a Patron**
  - **URL**: /api/patrons/{id}
  - **Method**: PUT
  - **URL Parameters**: id (Patron ID)
  - **Request Body**: Updated Patron object

- **Response**: Details of the updated patron.



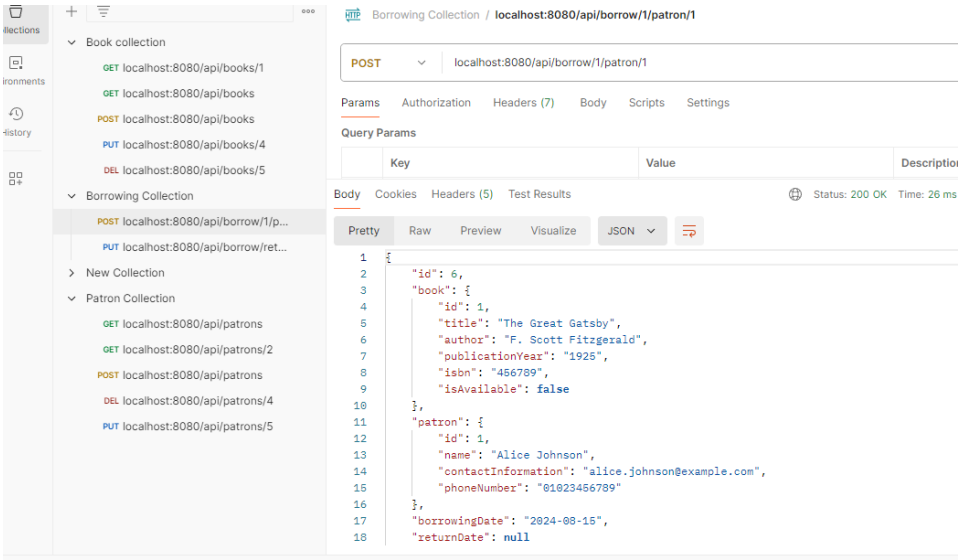
- **Delete a Patron**
  - **URL**: /api/patrons/{id}
  - **Method**: DELETE
  - **URL Parameters**: id (Patron ID)
  - **Response**: Success or failure message.



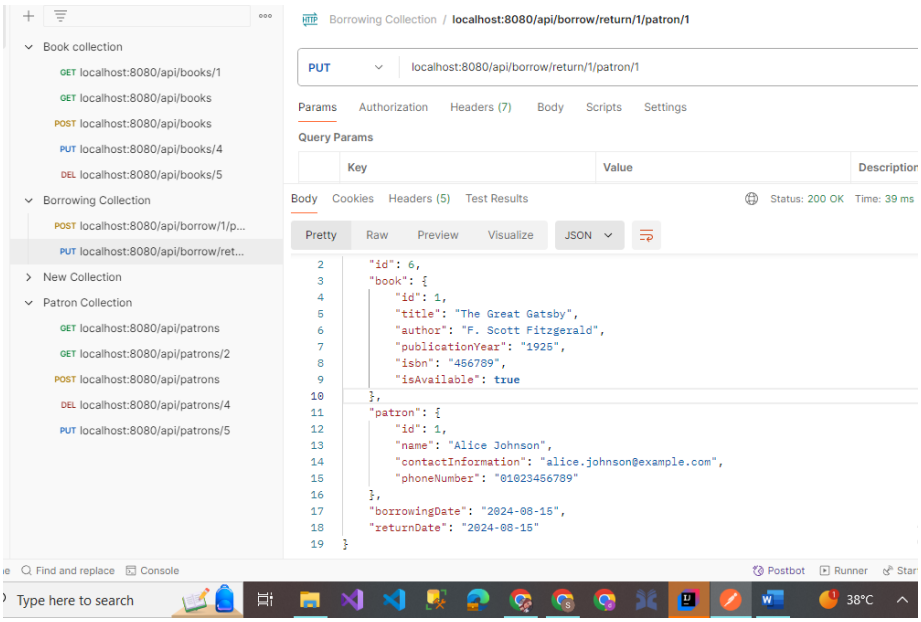
Borrowing Records

- **Borrow a Book**
  - **URL**: /api/borrowings/borrow
  - **Method**: POST
  - **Request Body**: Borrowing record object ({ "bookId": 1, "patronId": 1 })

- **Response**: Details of the borrowing record created.



- **Return a Book**
  - **URL**: /api/borrowings/return
  - **Method**: POST
  - **Request Body**: Return record object ({ "bookId": 1, "patronId": 1 })
  - **Response**: Details of the returned borrowing record.



Testing

- Tests for the application are provided using JUnit and Mockito.
- Unit Tests

- The application includes unit tests for all API endpoints, ensuring that the functionality is validated. The tests are in the src/test/java/com/library/library directory.