**Library Management System**

**Overview:**

The **Library Management System** is a Spring Boot application designed to manage the operations of a library, including book management, patron management, and borrowing records. This system provides RESTful APIs for interaction with the backend services.

**Prerequisites:**

* Java Development Kit (JDK) 8 or higher
* Maven 3.6 or higher
* MySQL database
* Git

**Installation**

1. **Clone the repository:**

git clone https://github.com/EslamA99/Library-Management-System.git

cd Library-Management-System

1. Update the application.properties file located in src/main/resources to match your MySQL database configuration:

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  
spring.datasource.url=jdbc:mysql://localhost:3306/library\_management  
spring.datasource.username=root  
spring.datasource.password=root  
spring.jpa.show-sql=true  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect  
spring.jpa.hibernate.ddl-auto=create

# spring.jpa.hibernate.ddl-auto=update

1. Build the project:

The application should now be running on <http://localhost:8080>.

mvn spring-boot:run

**API Endpoints:**

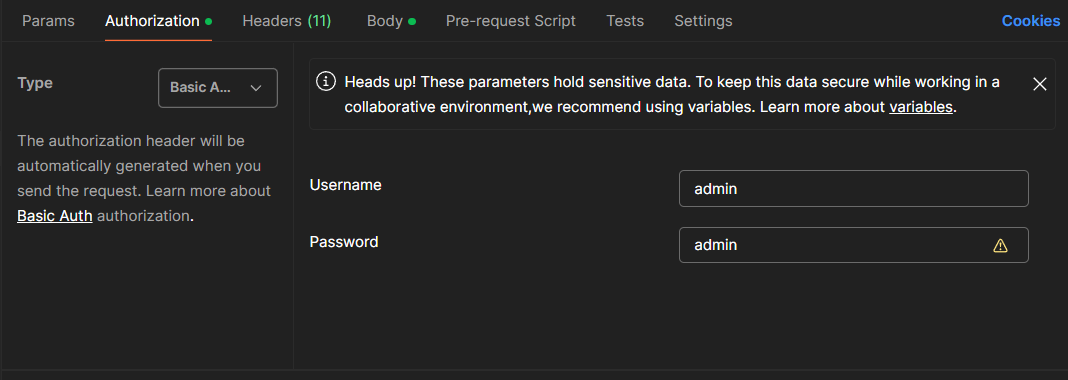
**Before try the APIs please use this to users to authenticate to run all APIs.**

**Basic Auth section in postman:-**

**Username: admin**

**Password:admin**

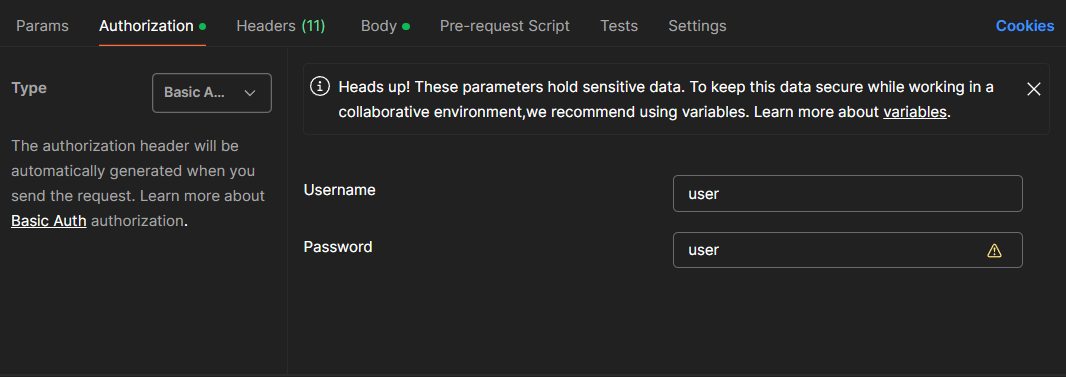
**Admin can access all APIs (books, Borrowing)**

****

**Username:user**

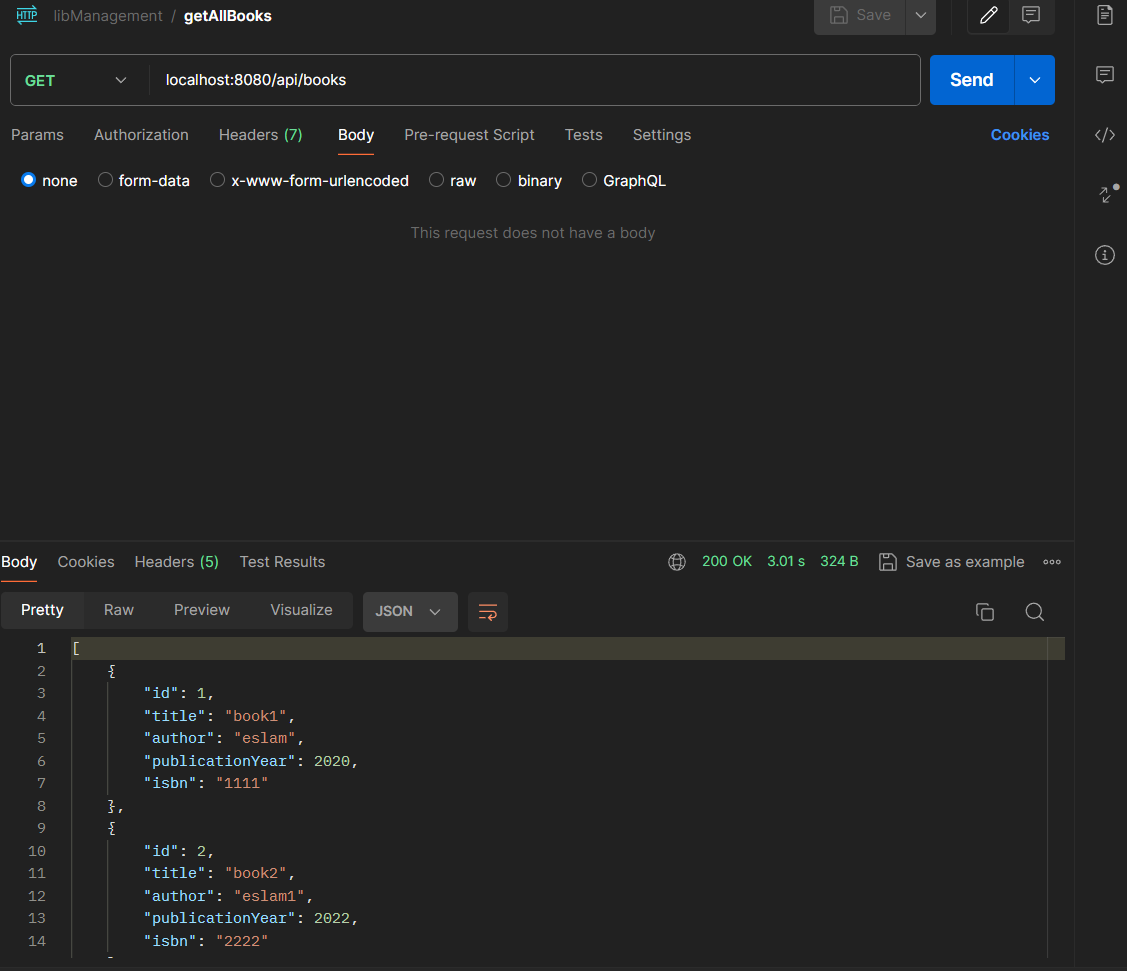
**Password:user**

**This user can only access (patrons ,Borrowing)**

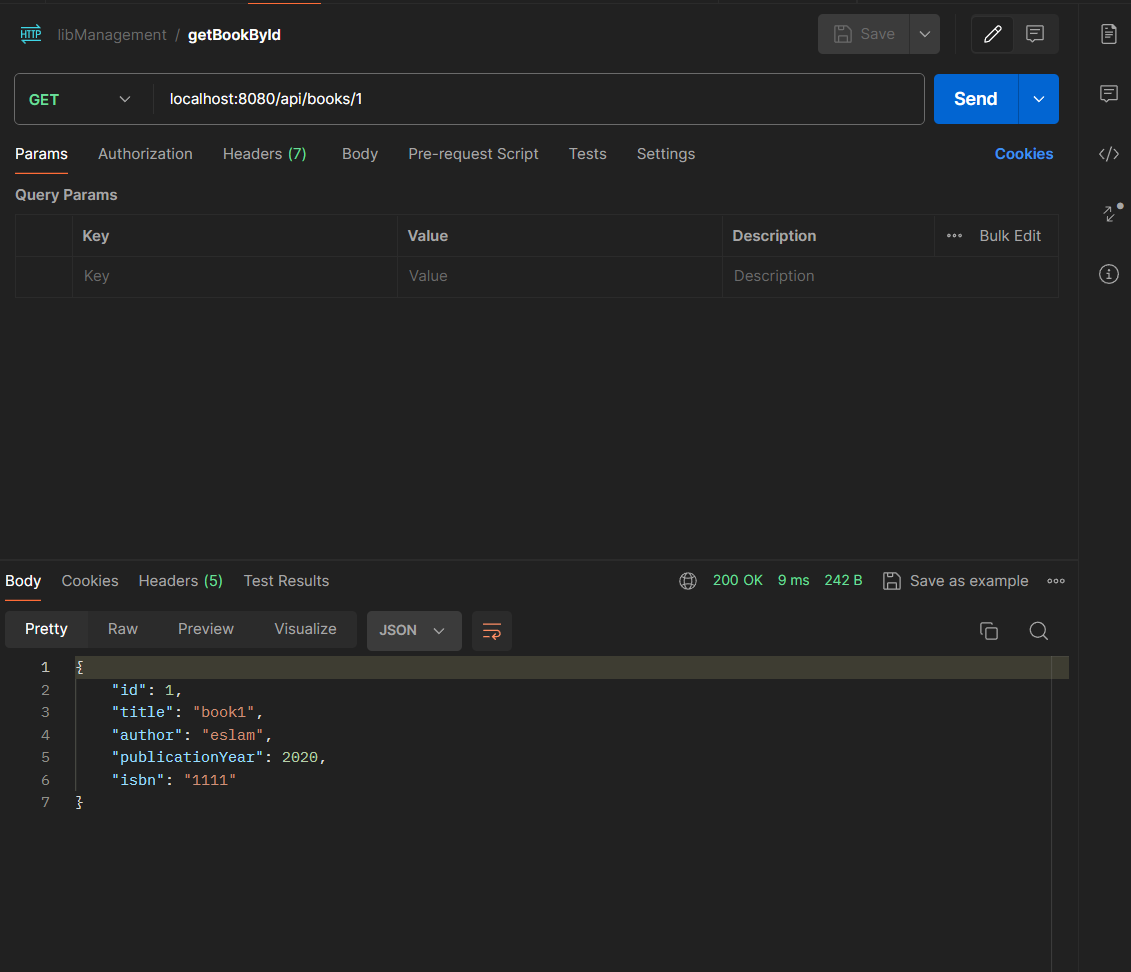
****

**Book Management Endpoints:**

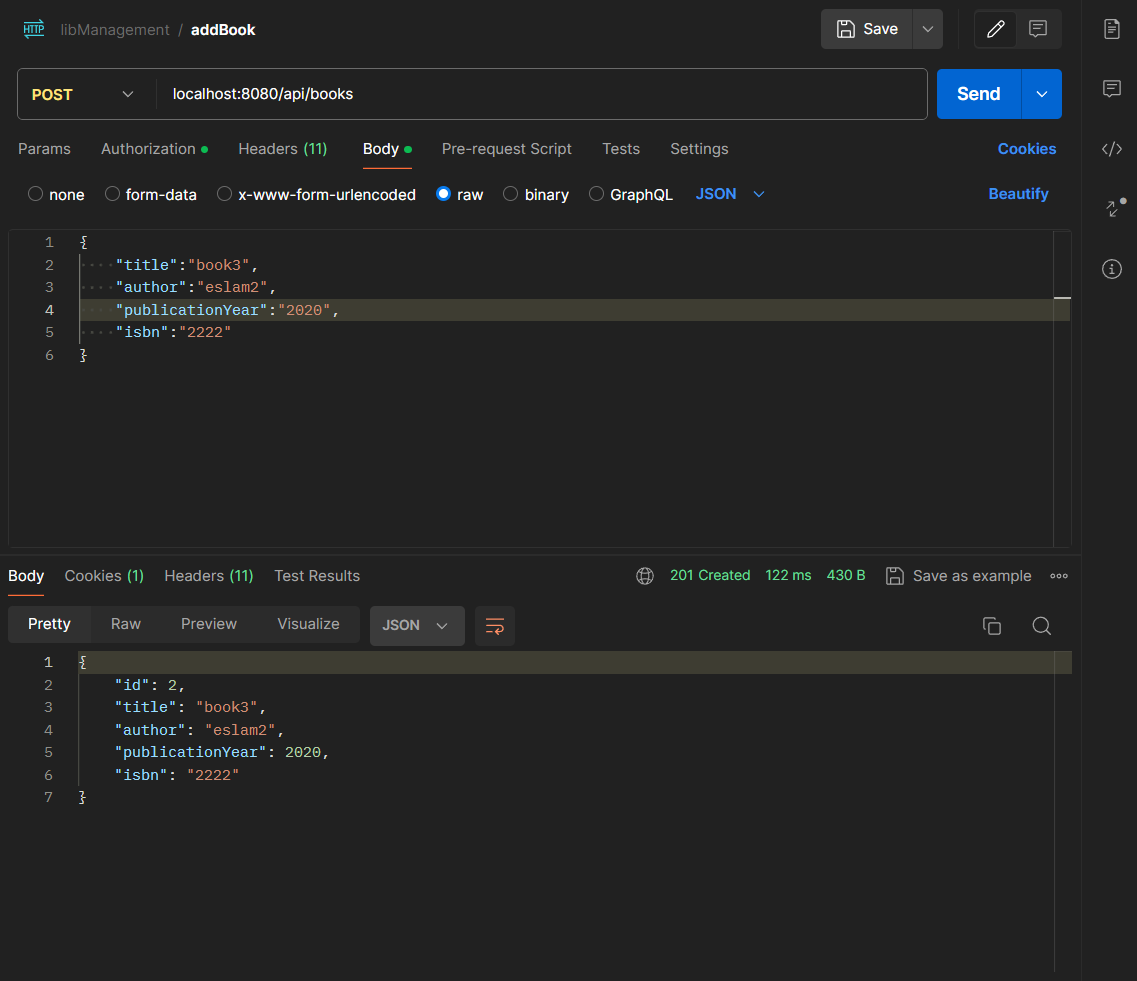
* **GET /api/books**: Retrieve a list of all books.



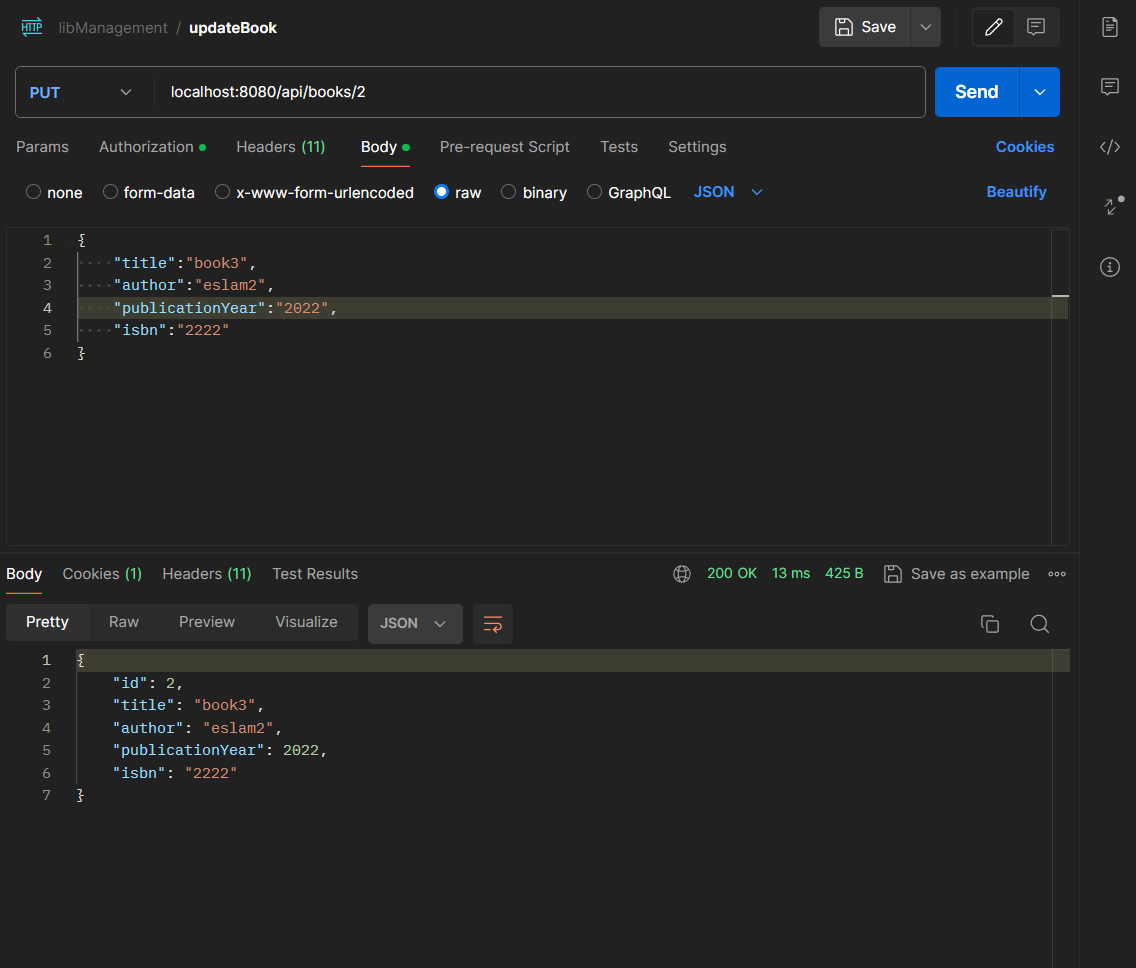
* **GET /api/books/{id}**: Retrieve details of a specific book by ID.



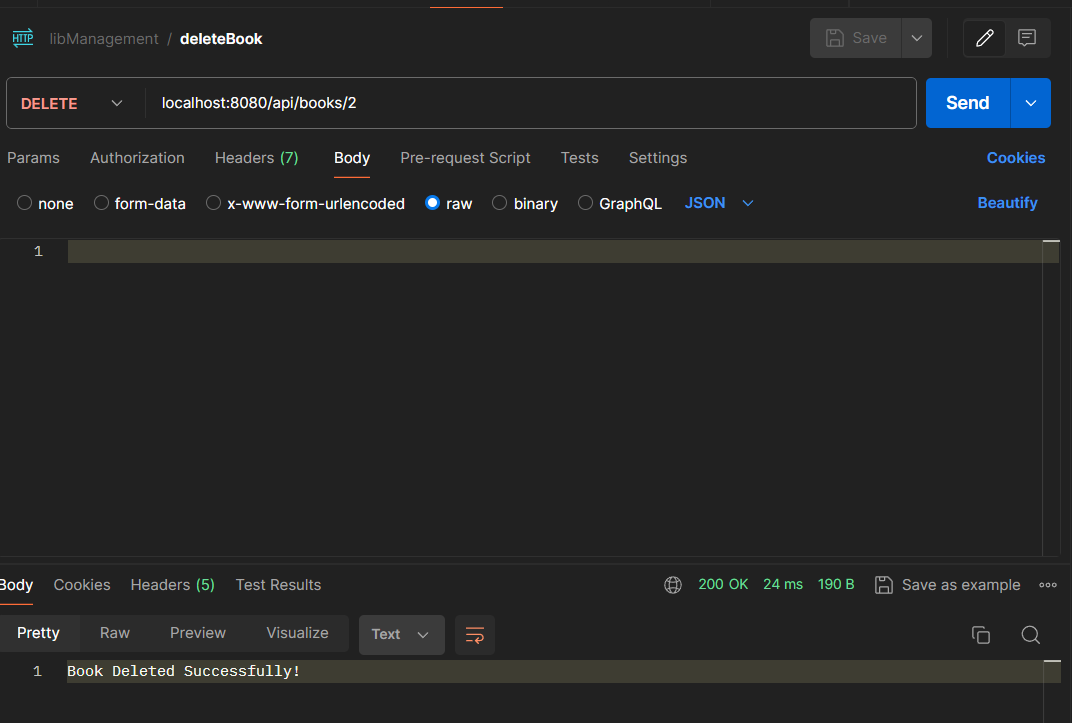
* **POST /api/books**: Add a new book to the library.



* **PUT /api/books/{id}**: Update an existing book's information.

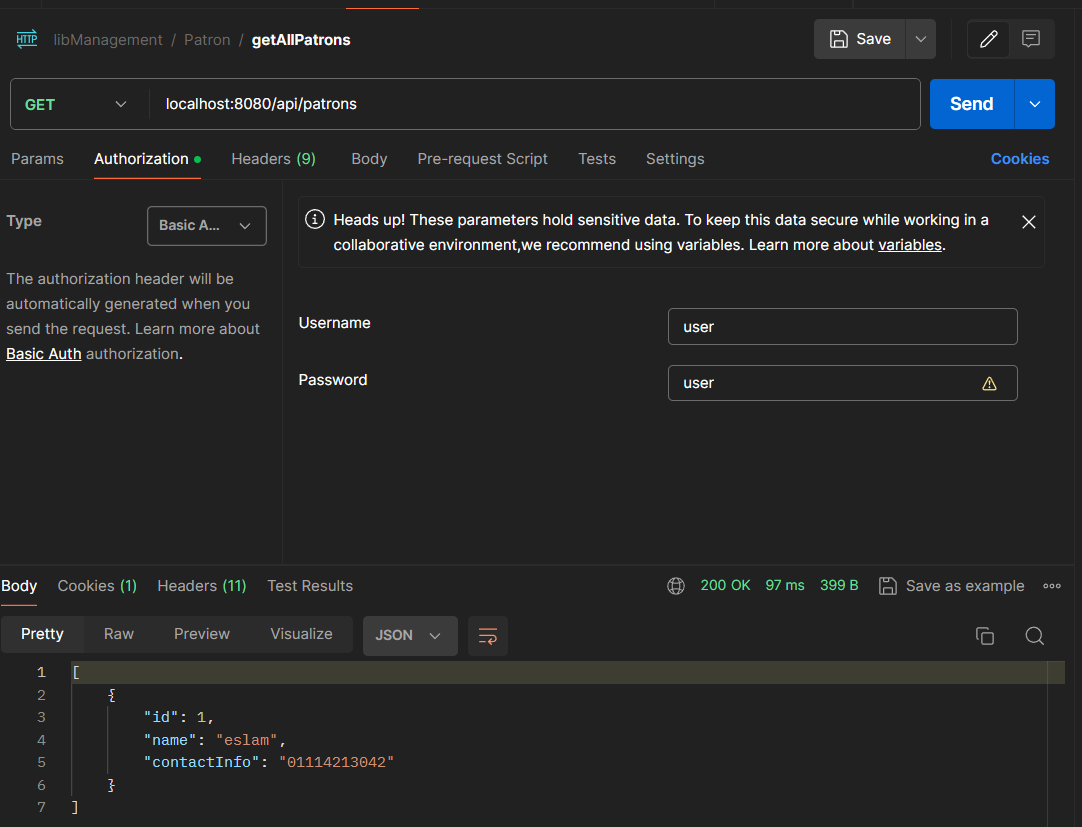


* **DELETE /api/books/{id}**: Remove a book from the library.

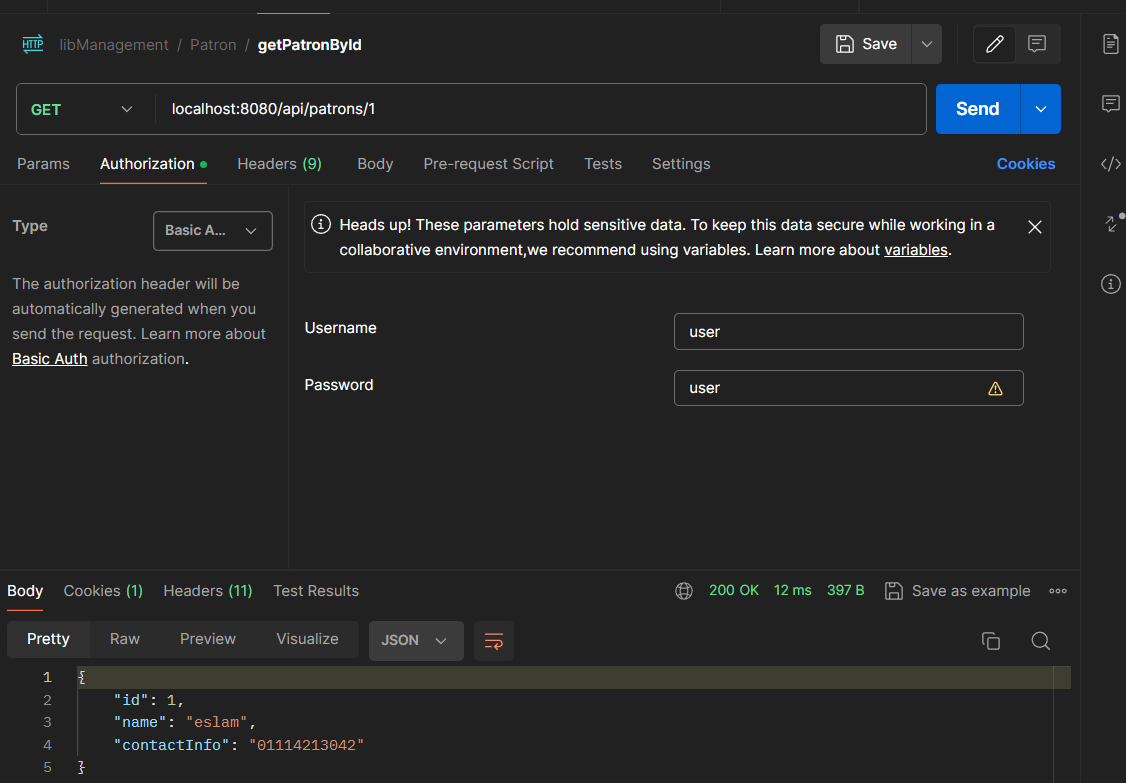


**Patron Management Endpoints:**

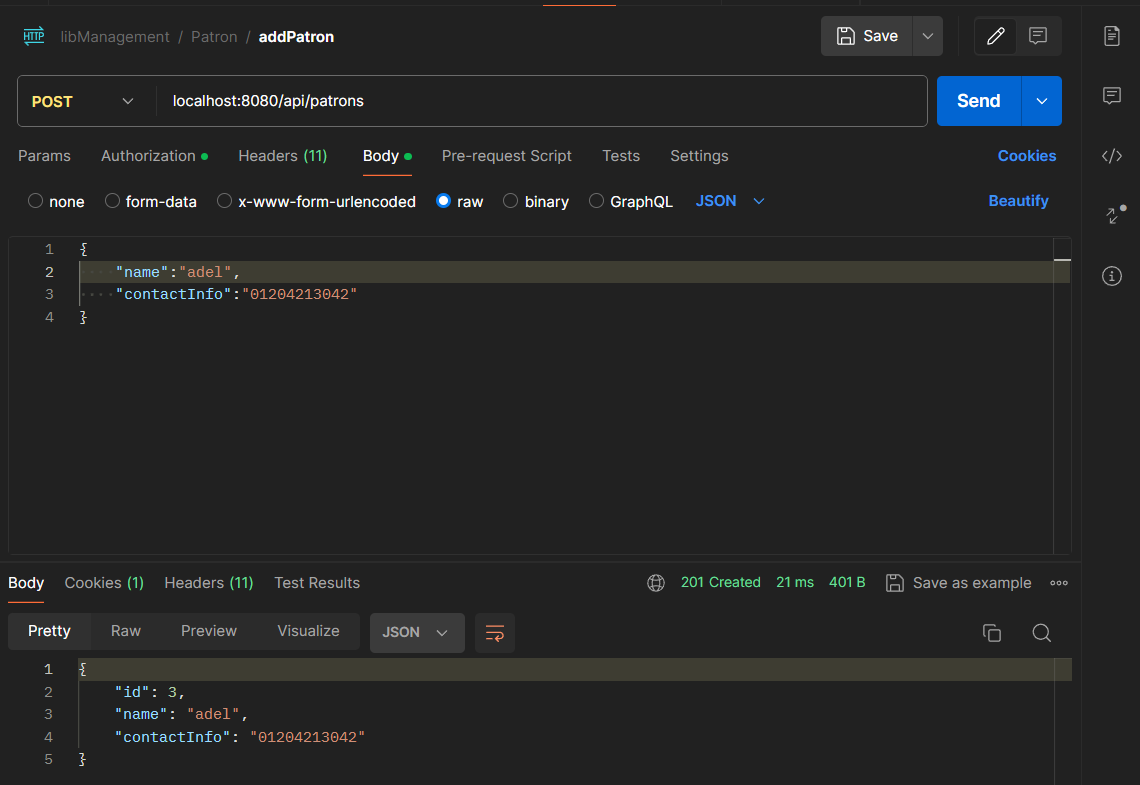
* **GET /api/patrons**: Retrieve a list of all patrons.



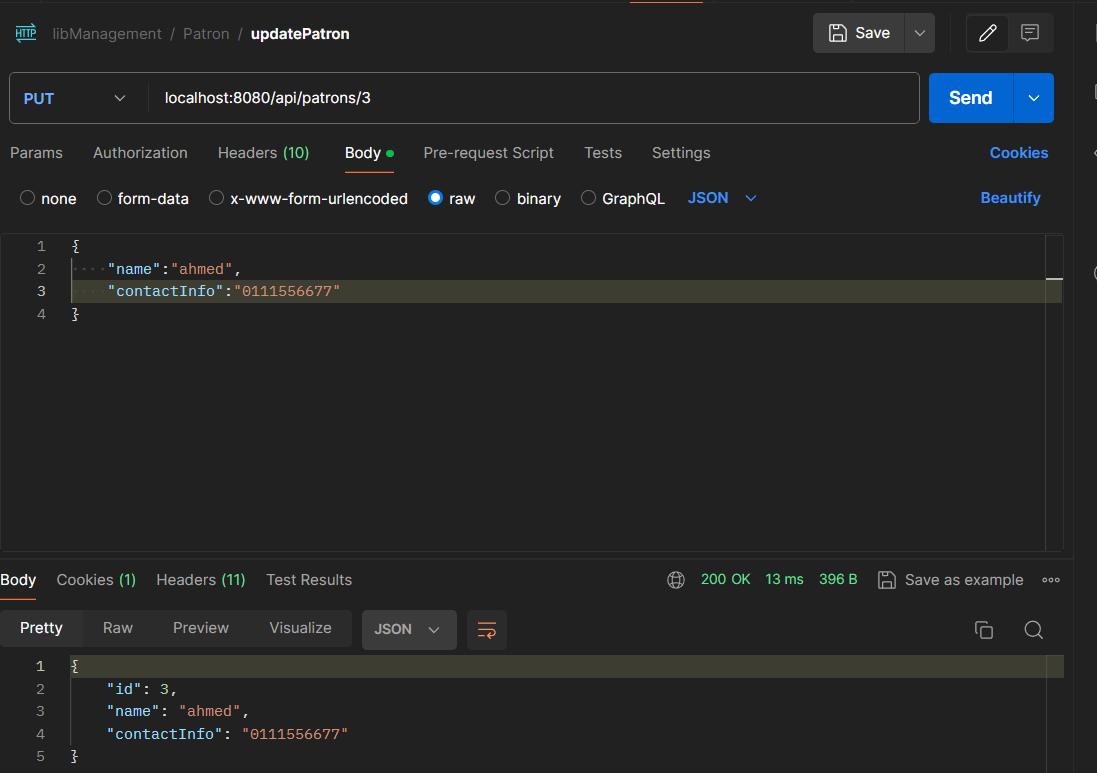
* **GET /api/patrons/{id}**: Retrieve details of a specific patron by ID.



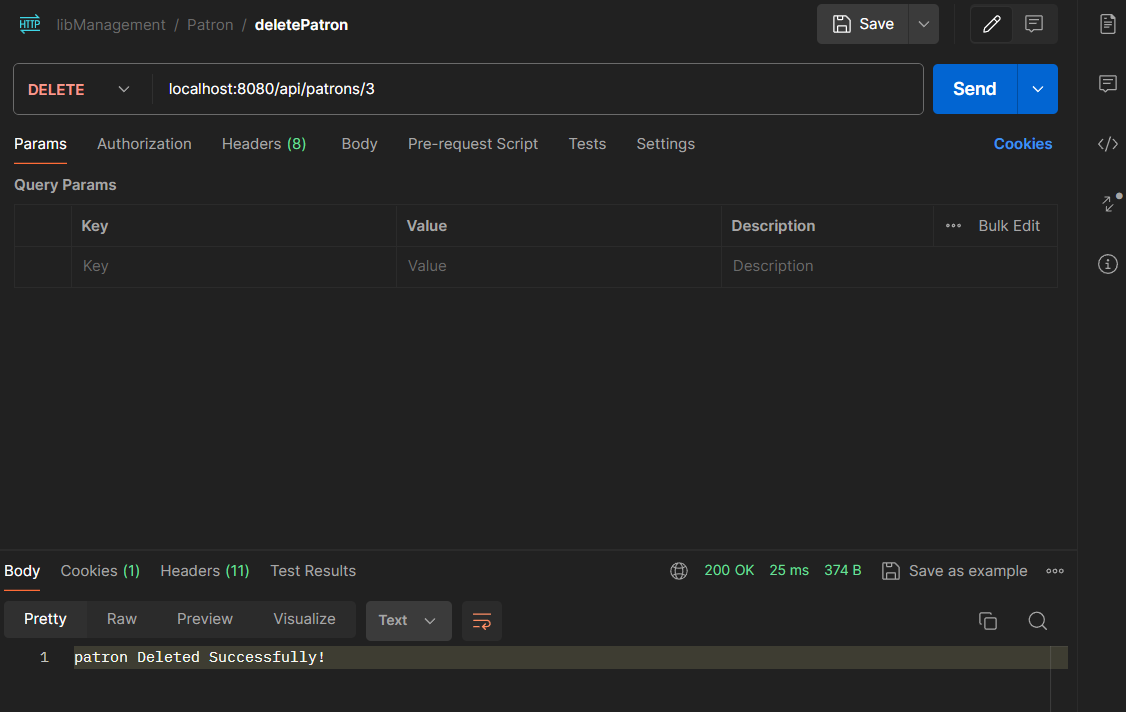
* **POST /api/patrons**: Add a new patron to the system.



* **PUT /api/patrons/{id}**: Update an existing patron's information.

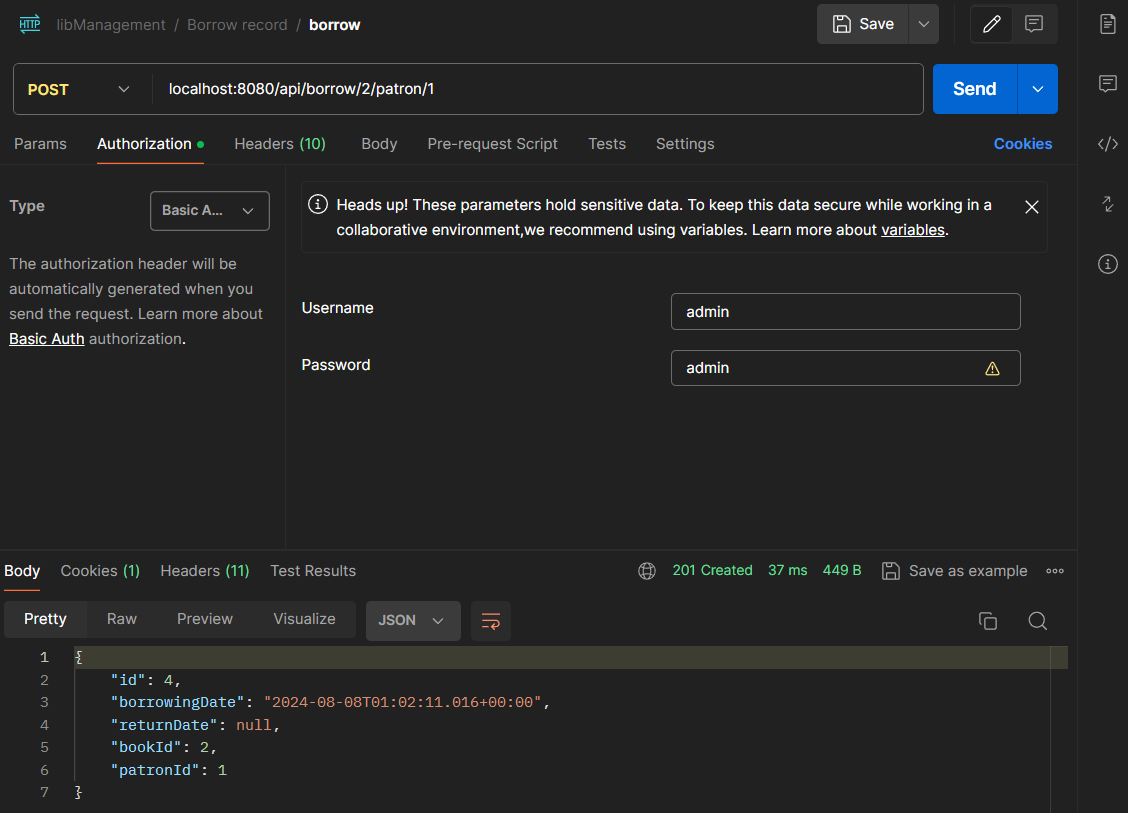


* **DELETE /api/patrons/{id}**: Remove a patron from the system.

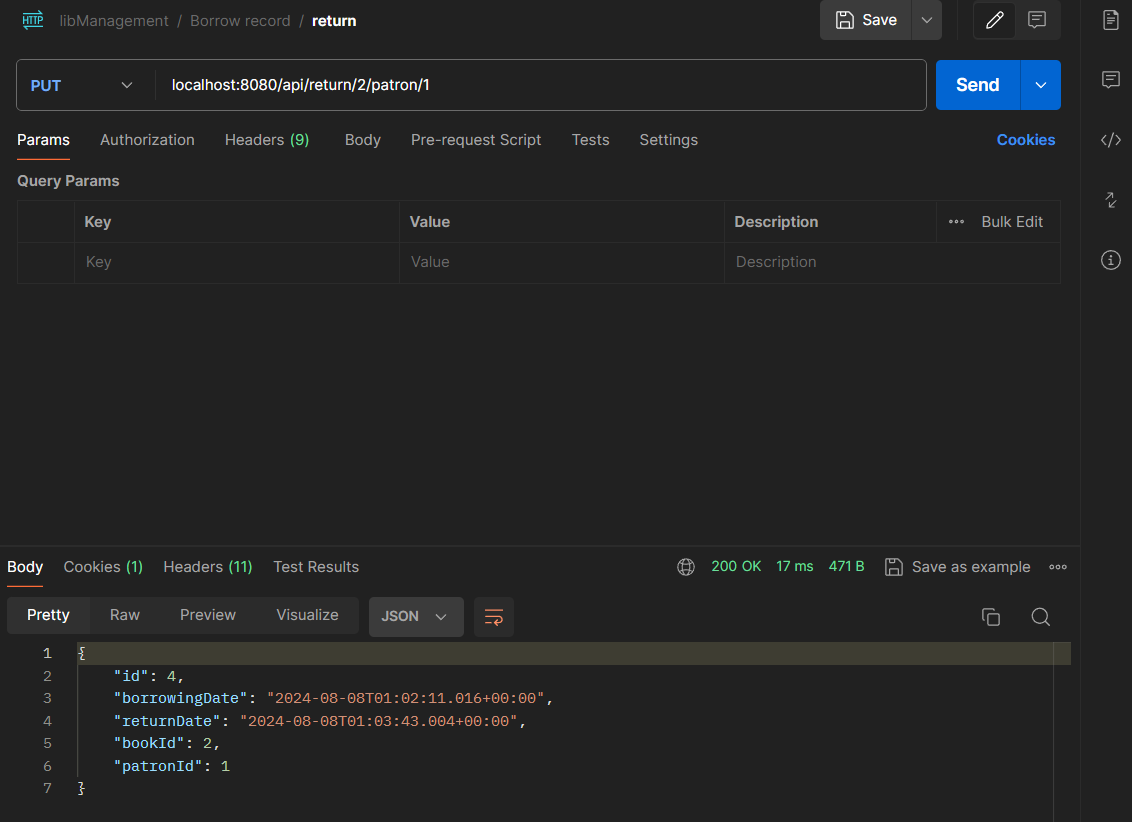


**Borrowing Endpoints:**

* **POST /api/borrow/{bookId}/patron/{patronId}**: Allow a patron to borrow a book.



* **PUT /api/return/{bookId}/patron/{patronId}**: Record the return of a borrowed book by a patron.



**Database Schema**

The database schema includes tables for books, patrons, and borrowing records. Here is a basic schema description:

* **Books**:
  + id (Primary Key)
  + title
  + author
  + publicationYear
  + isbn
* **Patrons**:
  + id (Primary Key)
  + name
  + contactInformation
* **Borrowing Records**:
  + id (Primary Key)
  + book\_id (Foreign Key to Books)
  + patron\_id (Foreign Key to Patrons)
  + borrowDate
  + returnDate