

# Postman Collection Documentation: Library Book Catalog API

Base URL: <http://localhost:8080>

---

## API Endpoints

Method	Endpoint	Description
POST	/books	Add a new book
GET	/books	Retrieve all books
GET	/books/{id}	Get book by ID
DELETE	/books/{id}	Delete book by ID
PATCH	/books/{id}/availability	Update book availability status

## Developer Info

Developed by: Darshan Chaudhari

Tools: Spring Boot, Postman, STS, GitHub

### Git repo -

[https://github.com/darshanchaudharii/Library\\_Management\\_Books\\_Catelog\\_Backend\\_Server-.git](https://github.com/darshanchaudharii/Library_Management_Books_Catelog_Backend_Server-.git)

### Postman Shared link -

<https://restless-rocket-398046.postman.co/workspace/My-Workspace~d611aeb1-7e53-402e-8557-fca0c9eaf520/collection/45915590-141ca865-08a1-49b2-8588-ac7aeda002e4?action=share&creator=45915590>

1. Title - POST /books – Add New Book (<http://localhost:8080/books>)

My Workspace

NewImport

collections

Library Catalog API

POST Add a New Book

GET Get All Books

GET Get Book by ID

DEL Delete Book by ID

PATCH Update Book Availability

New Collection

New Collection

TaskManager API

GET 1 - Register.

GET 2 - Login

POST Add a New Book

TaskManager API

Library Catalog API / Add a New Book

POSThttp://localhost:8080/api/books

Send

ParamsAuthorizationHeaders (9)BodyScriptsSettings

noneform-datax-www-form-urlencodedrawbinaryGraphQLJSON

```
1 {
2   "title": "To Kill a Mockingbird",
3   "author": "Harper Lee",
4   "isbn": "9780061120084",
5   "available": true
6 }
```

BodyCookiesHeaders (5)Test Results

200 OK10 ms266 B

Save Response

JSONPreviewVisualize

```
1 {
2   "title": "To Kill a Mockingbird",
3   "author": "Harper Lee",
4   "isbn": "9780061120084",
5   "id": 3,
6   "available": true
7 }
```

OnlineFind and replaceConsole

PostbotRunnerStart ProxyCookiesVaultTrash

## 2. GET /books – Retrieve All Books – Add New Book (http://localhost:8080/books)

The screenshot shows the Postman interface with a workspace named "My Workspace". The "Library Catalog API" collection is selected, and the "GET Get All Books" request is active. The request URL is "http://localhost:8080/api/books". The response is a 200 OK status with a 15 ms response time and 467 B of data. The response body is a JSON array of three book objects.

**Request:**

```
GET http://localhost:8080/api/books
```

**Response:**

```
200 OK • 15 ms • 467 B
```

```
{
  "1": {
    "title": "The Alchemist",
    "author": "Paulo Coelho",
    "isbn": "9780806231507",
    "id": 1,
    "available": true
  },
  "2": {
    "title": "1984",
    "author": "George Orwell",
    "isbn": "9780451524935",
    "id": 2,
    "available": false
  },
  "3": {
    "title": "To Kill a Mockingbird",
    "author": "Harper Lee",
    "isbn": "9780806112084",
    "id": 3,
    "available": true
  }
}
```

### 3. GET /books/{id} – Get Book by ID (http://localhost:8080/books/3)

The screenshot shows the Postman interface with a workspace named "My Workspace". The "Library Catalog API" collection is expanded, and the "GET Get Book by ID" request is selected. The request is a GET method to the URL "http://localhost:8080/api/books/3". The response is a 200 OK status with a response time of 28 ms and a body size of 266 B. The response body is a JSON object representing a book.

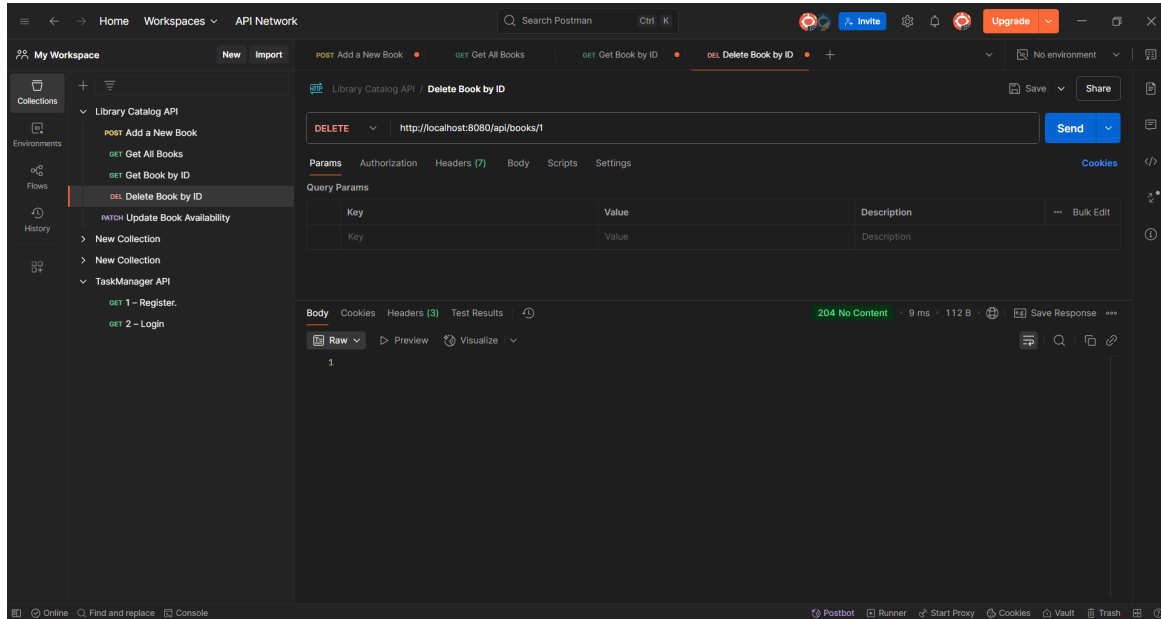
**Request:**

- Method: GET
- URL: http://localhost:8080/api/books/3

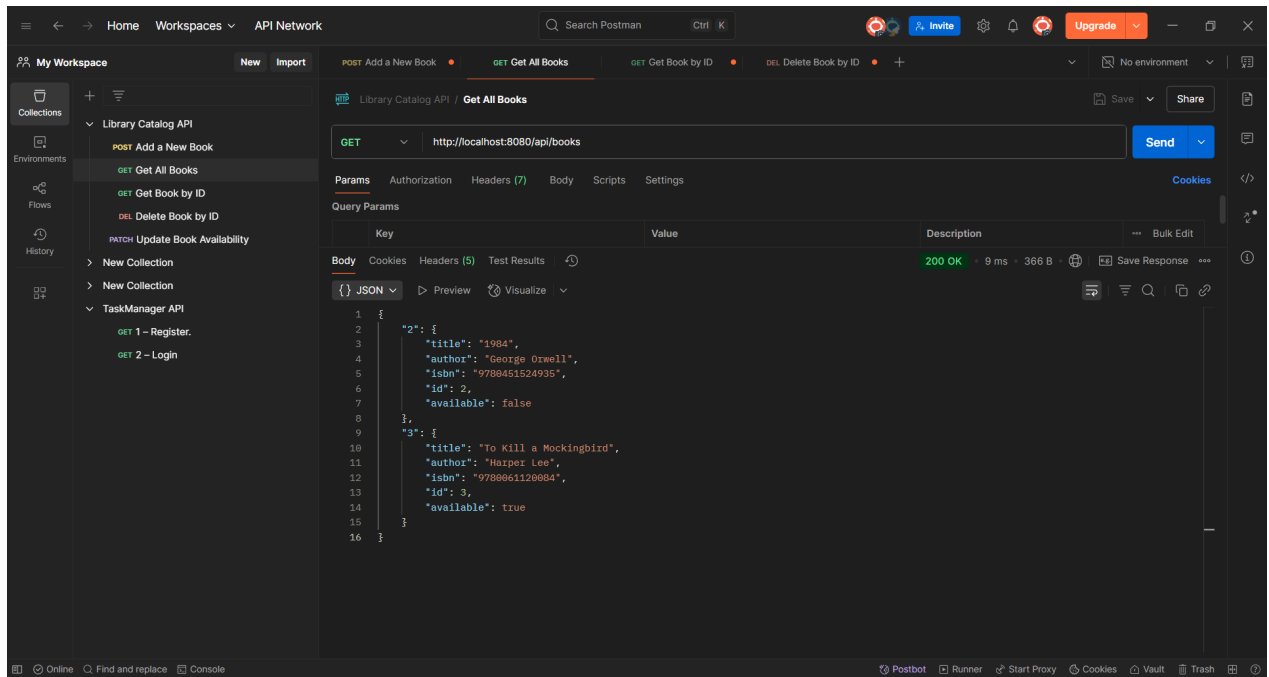
**Response:**

```
{  "title": "To Kill a Mockingbird",  "author": "Harper Lee",  "isbn": "9780061120084",  "id": 3,  "available": true}
```

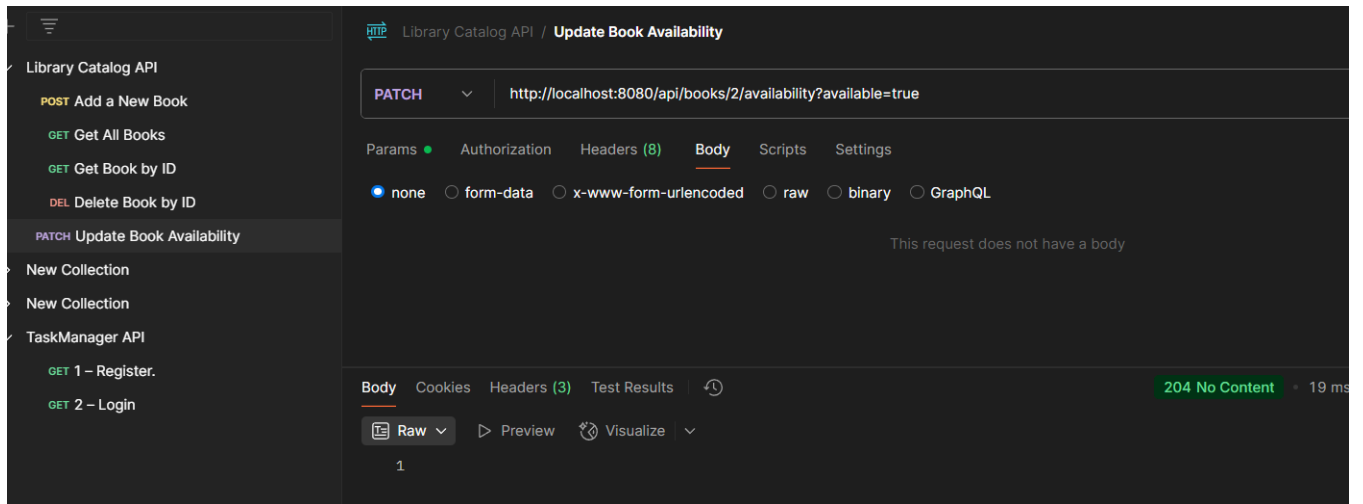
#### 4. DELETE /books/{id} (http://localhost:8080/books/1)



#### After Deletion -



## 5. PATCH /books/{id}/availability (http://localhost:8080/books/1/availability?available=true )



After Updating -

