**BookStore-API**

**Github Repo Link: -** <https://github.com/Imemyslf/Bookstore-API>

**API Endpoints for Testing (Postman): -**

|  |  |  |
| --- | --- | --- |
| Operation | Method | Endpoints |
| * Register User | POST | |  | | --- | |  |   /auth/register |
| * Login User | POST | /auth/login |
| * Add Book | POST | /books |
| * Update Book | PUT | |  | | --- | |  |   /books/{BookID} |
| * Get Book by ID | GET | /books/{BookID} |
| * Search By Genre | GET | /books/search?genre={genre} |
| * Delete Book | DELETE | /books/{BookID} |

**👤User API: -**

**1. Register User (POST): -** <http://localhost:3000/auth/register>

This endpoint is used to **create a new user account** by submitting valid credentials.

**➡️ Steps in Postman:**

* Go to the **Body** tab.
* Select the **raw** option.
* Set the format to **JSON**.

Enter the user credentials in the following format:

{

"email": "<your\_email>",

"password": "<your\_password>"

}

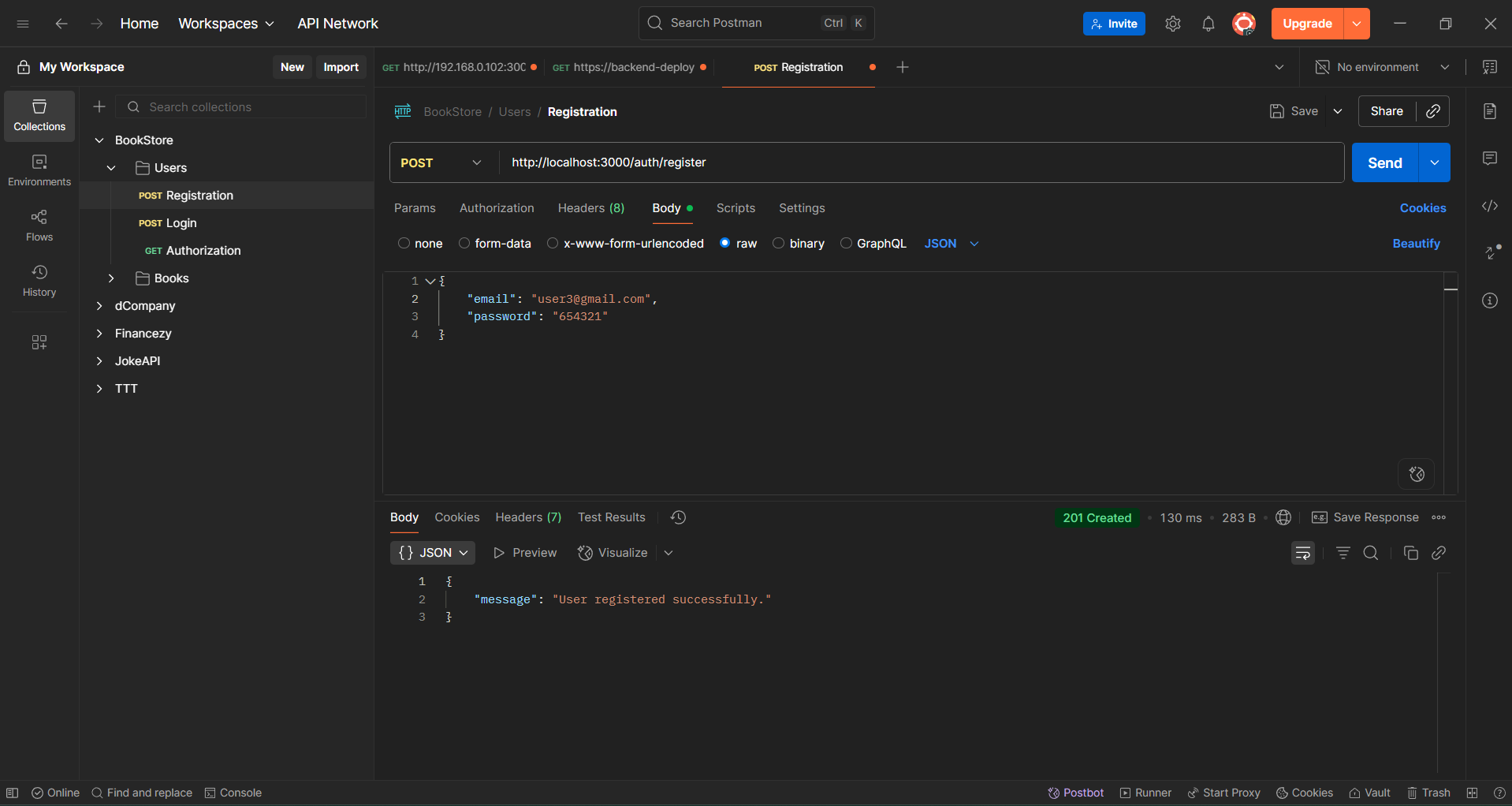
**📌 Example:**

{

"email": "user3@gmail.com",

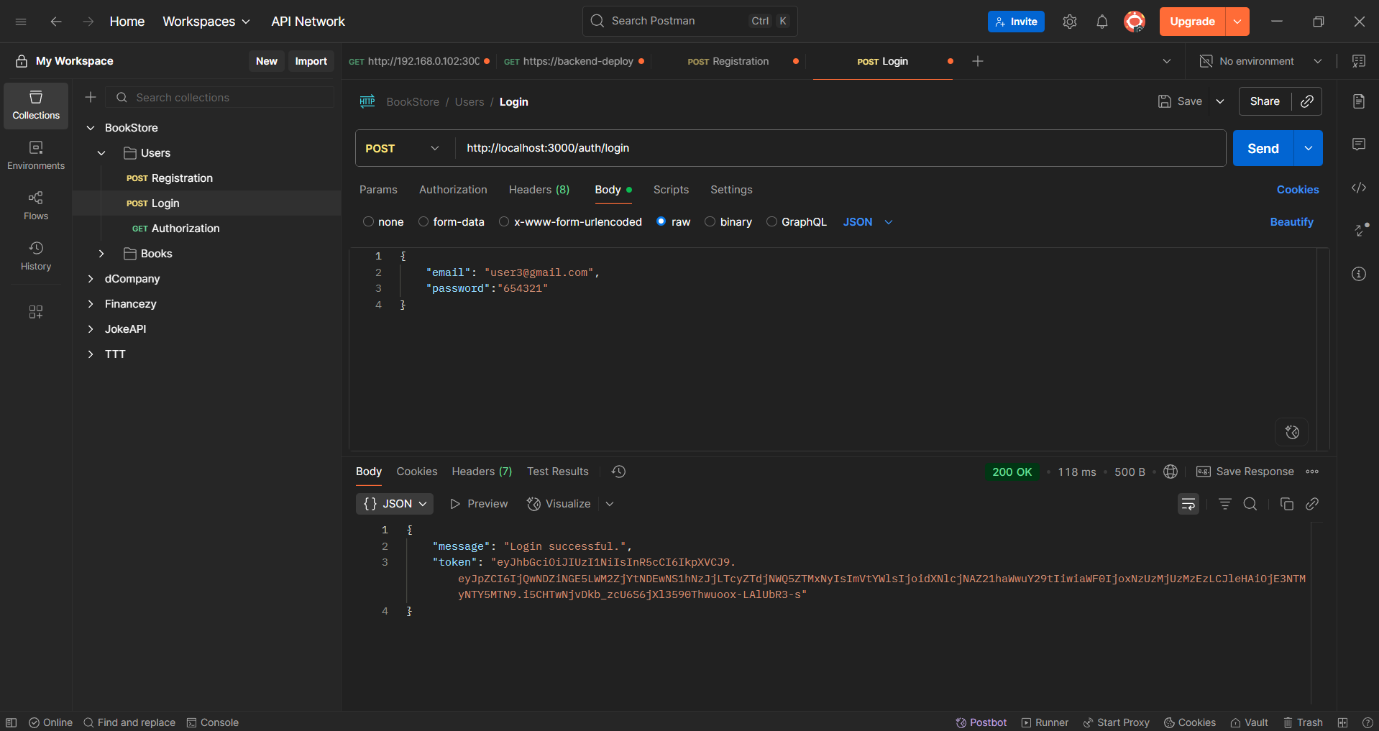
"password": "6454321"

}

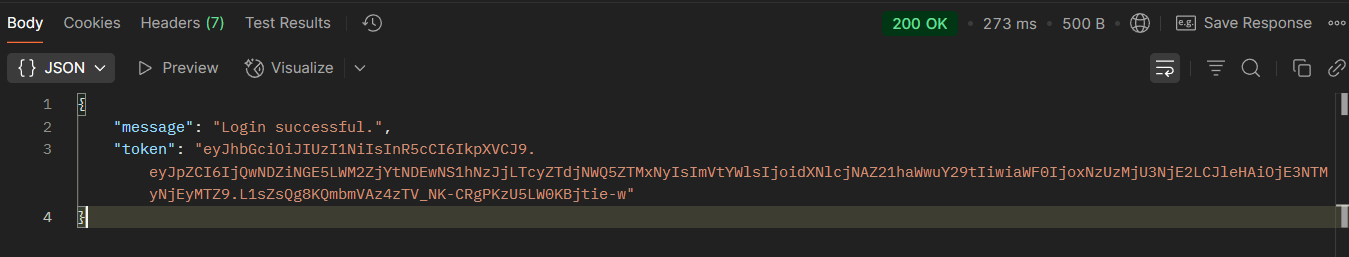


**2. Login User (POST): -** <http://localhost:3000/auth/login>

To log in, follow the same procedure as registration:  
Go to the **Body** tab in Postman, select **raw**, set the format to **JSON**, and enter the login credentials in the same structure.



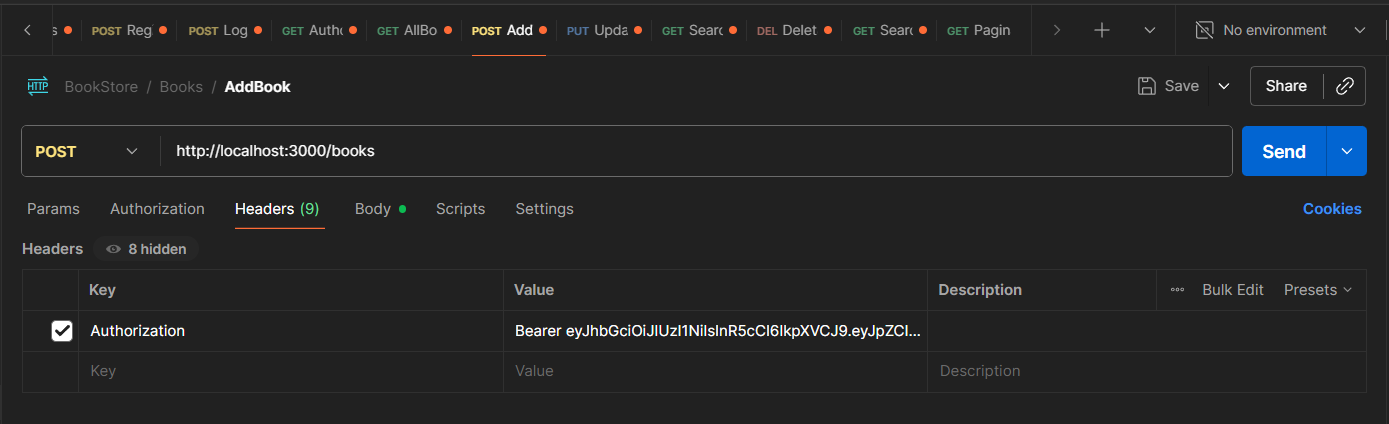
Upon making a successful login request, Postman returns a JSON response similar to the following:

****

The response includes a JWT token, which should be copied and used for authorization in subsequent API requests (e.g., for protected routes related to booking processes or user-specific operations).

This token is typically added to the request header as follows:

**Authorization: Bearer <your\_token\_here>**

****

**📖 Book API: -**

1. **➕ Add Book (POST): -** <http://localhost:3000/books>

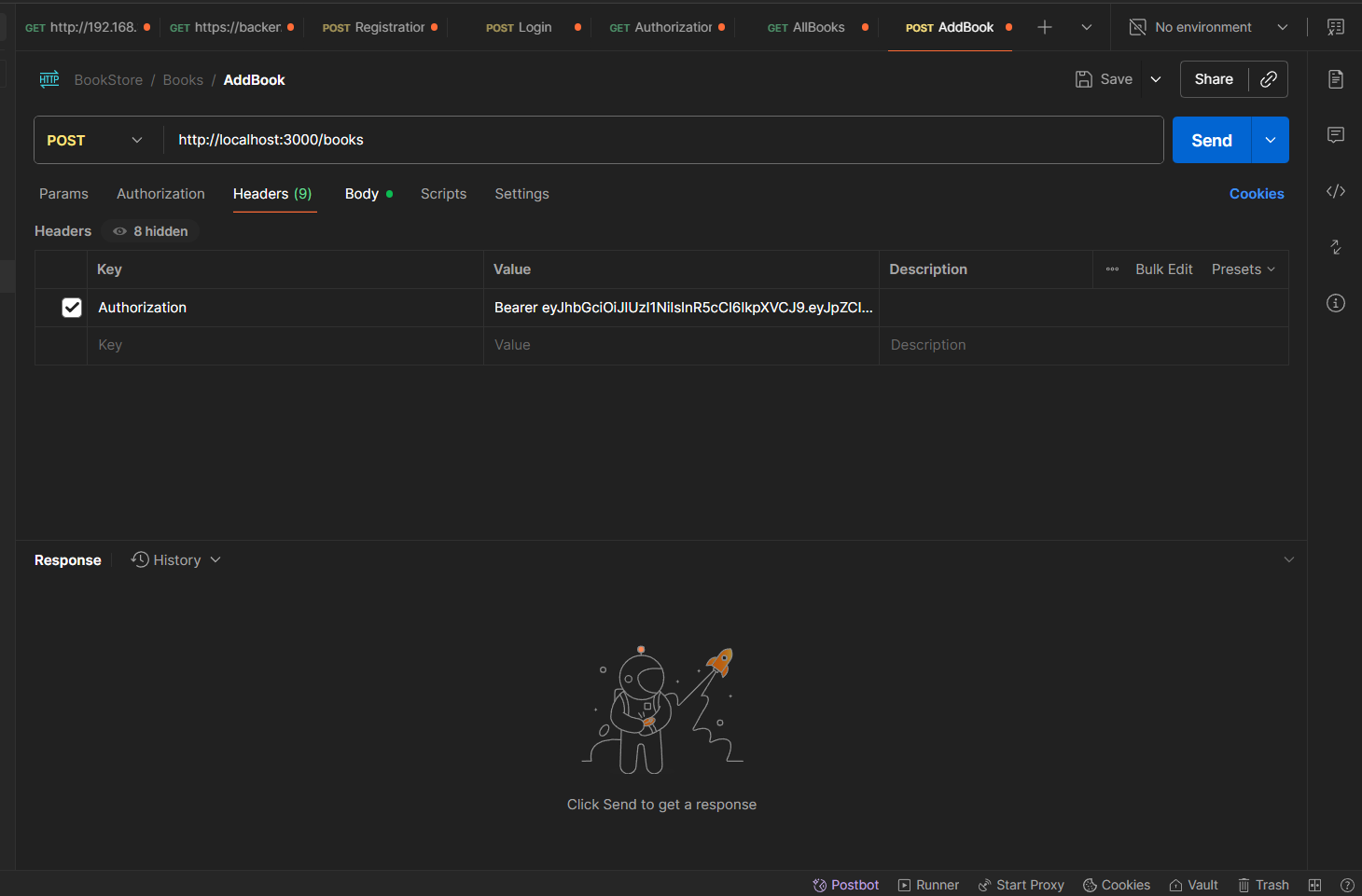
**Step 1:** - Copy the JWT token received after a successful login. In Postman, go to the "Headers" tab for the request and add:

📌 Example: -

**Key**: Authorization

**Value**:

Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjQwNDZiNGE5LWM2ZjYtNDEwNS1hNzJjLTcyZTdjNWQ5ZTMxNyIsImVtYWlsIjoidXNlcjNAZ21haWwuY29tIiwiaWF0IjoxNzUzMjUzMzEzLCJleHAiOjE3NTMyNTY5MTN9.i5CHTwNjvDkb\_zcU6S6jXl3590Thwuoox-LAlUbR3-s

****

**Step 2: -**

Add the details of the book in json format in Body section as: -

{

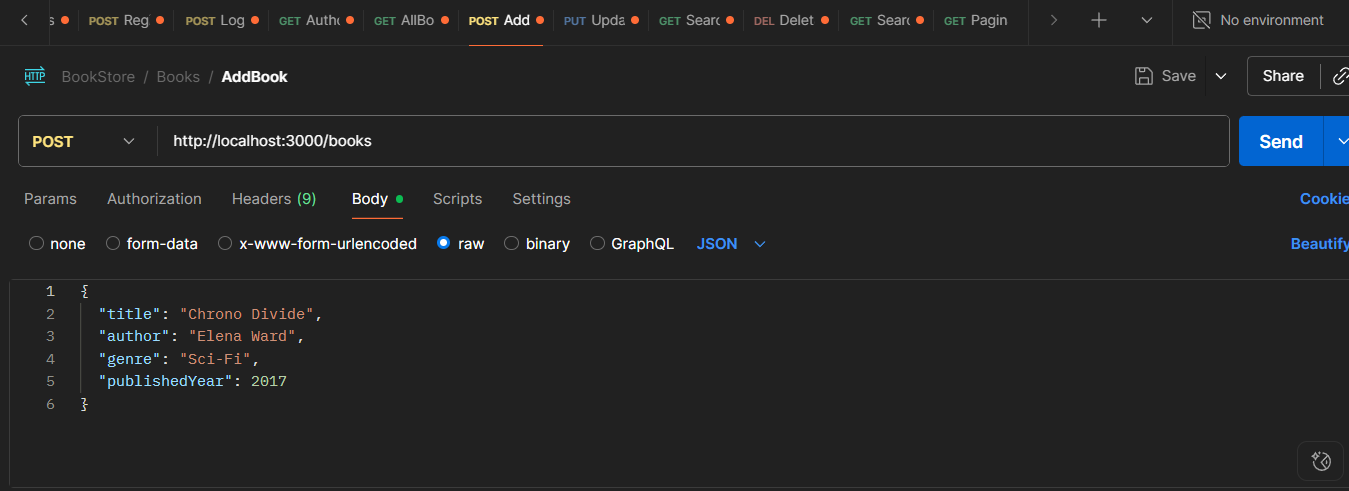
"title": "Chrono Divide",

  "author": "Elena Ward",

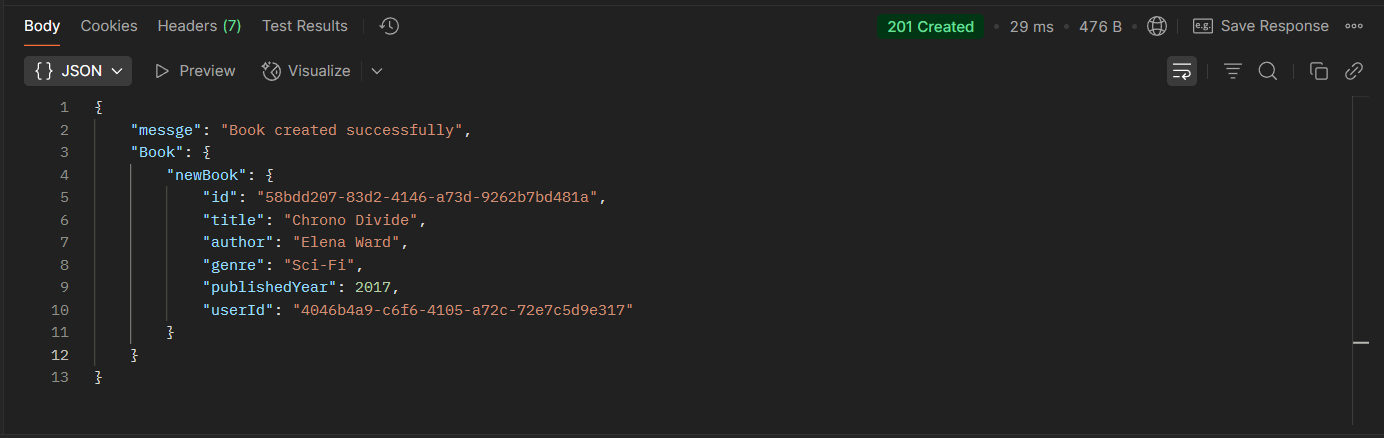
  "genre": "Sci-Fi",

  "publishedYear": 2017

}

****

**❔Output: -**

****

**📝 NOTE: -**

Authorization Requirement for Book Operations.

After successfully logging in and receiving the JWT token, it is essential to include this token in the Authorization header for all subsequent API requests related to Books operations.

**➡️ Step 1 (Mandatory for each request):**  
Set the Authorization header in Postman as:

**Authorization: Bearer <your\_token\_here>**

⚠️ **Important:** If this step is skipped, the server will respond with an error:

{

"error": "Missing token"

}

1. **🔁 Update Book (PUT): -** http://localhost:3000/books/ {**BookID**}

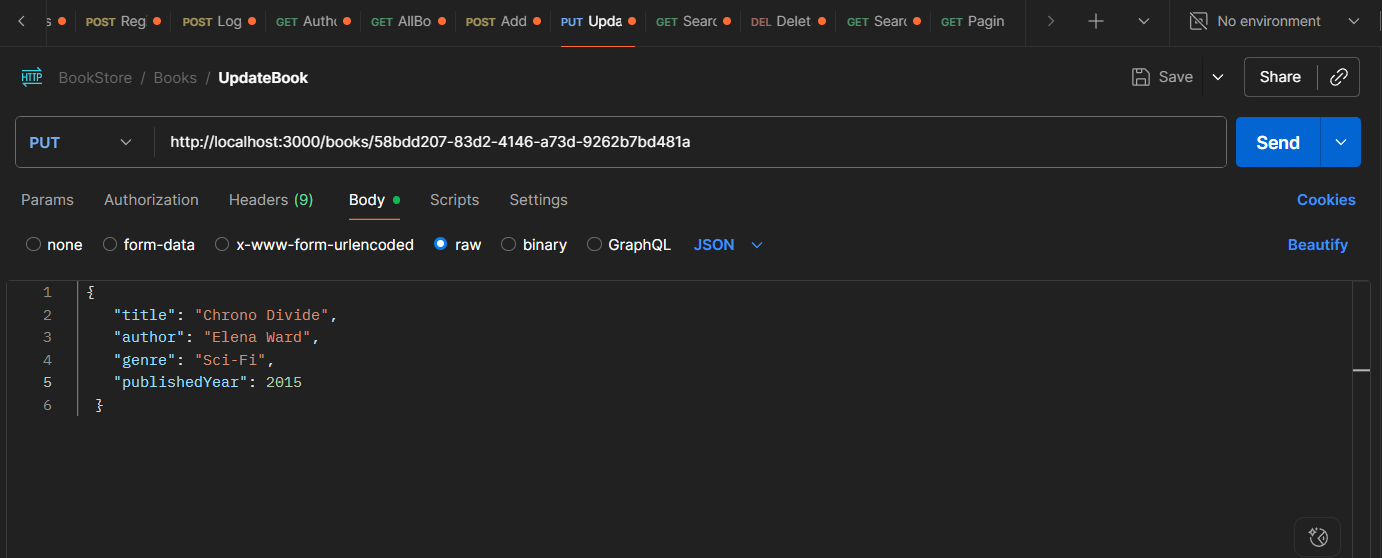
After successfully adding a book via the AddBook API, the server responds with a JSON object that includes a unique identifier for the book (usually labelled as **“id”**).

The BookID returned after adding a book is: `58bdd207-83d2-4146-a73d-9262b7bd481a`  
This ID should be used in all book-specific API operations (GET, PUT, DELETE).

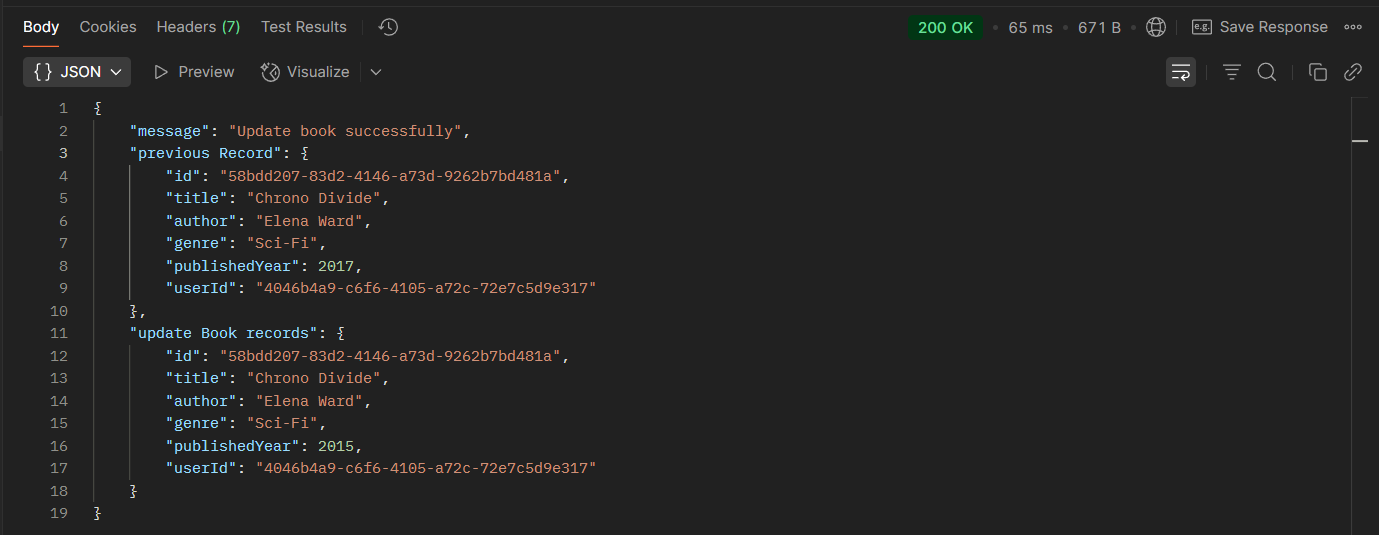
➡️ http://localhost:3000/books/ {**BookID**}

Example:

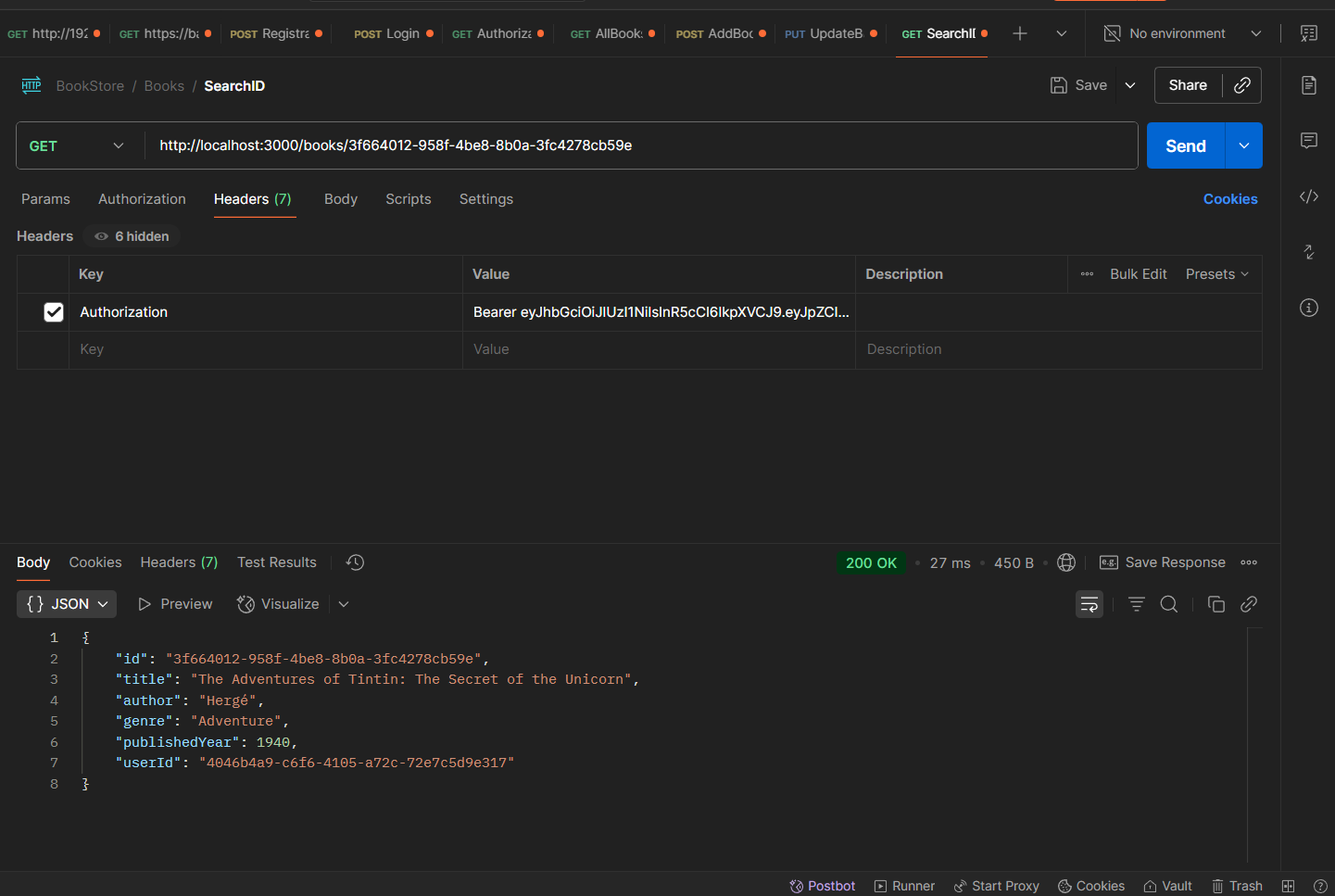
<http://localhost:3000/books/58bdd207-83d2-4146-a73d-9262b7bd481a>

****

**❔Output: -**

****

1. **🪪 Search By ID (GET): -** [http://localhost:3000/books/{BookID}](http://localhost:3000/books/%7bBookID%7d)

****

1. **🗃️ Search By Genre (GET): -** <http://localhost:3000/books/search?genre=Adventure>

This endpoint allows you to retrieve a list of books that match a specific genre.

🔍 How to Use:

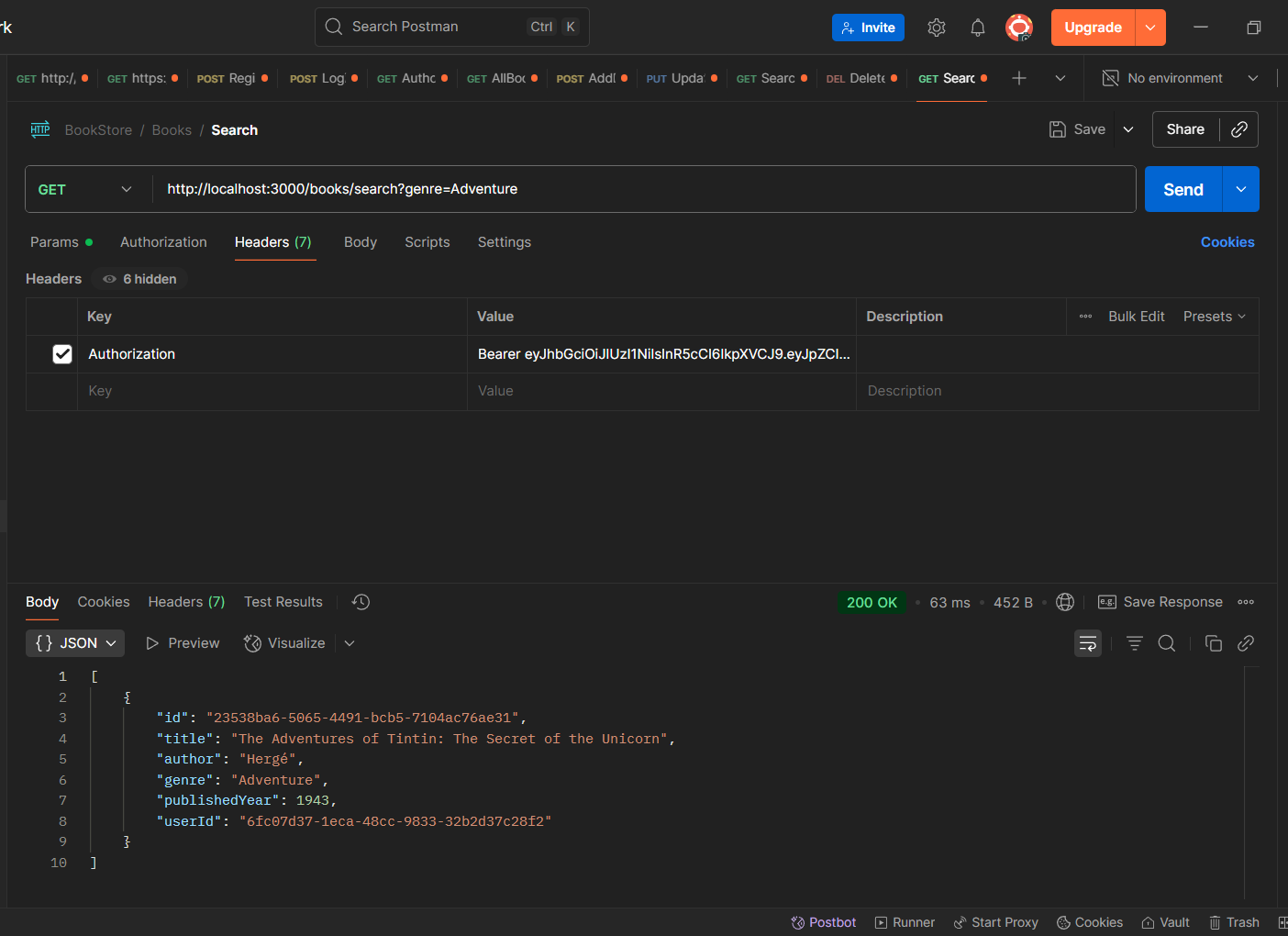
Pass the desired genre as a query parameter in the URL.  
Replace "Adventure" with any valid genre stored in the database.

**📌 Example:**

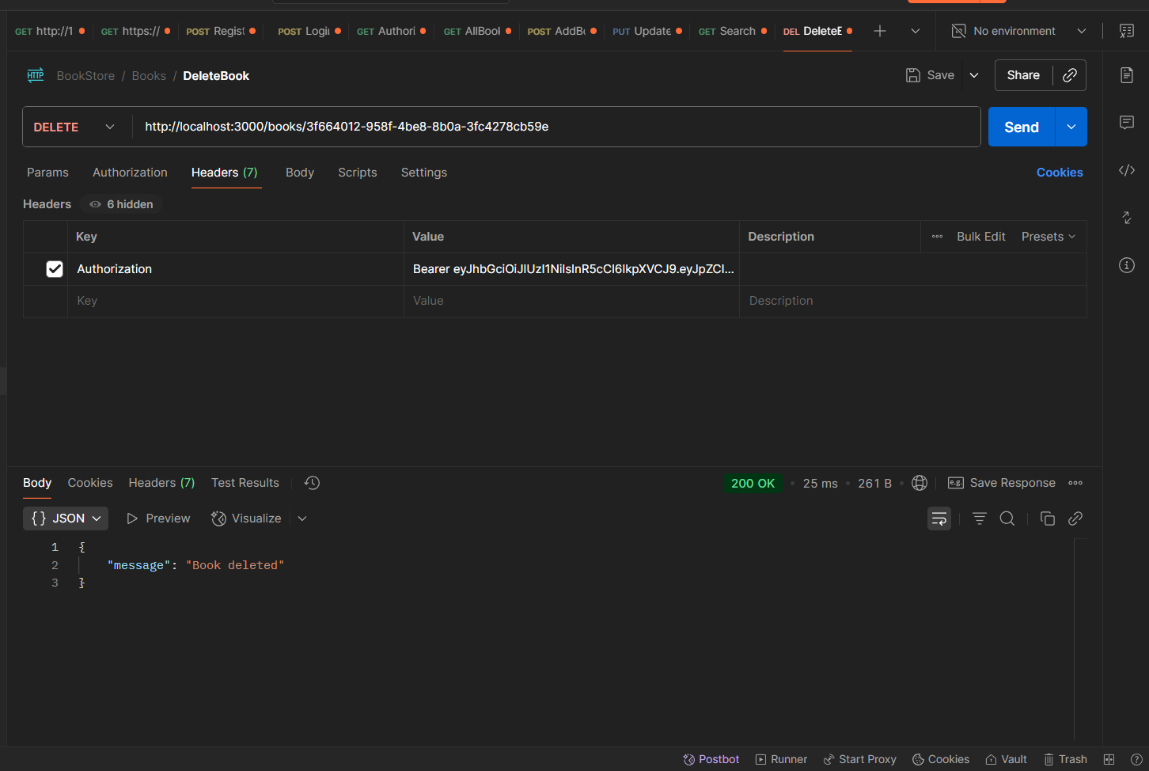
GET http://localhost:3000/books/search?genre=Fantasy

**✅ Response:**

Returns an array of book objects that belong to the specified genre.

****

1. **🗑️ Delete Book (DELETE): -** [http://localhost:3000/books/](http://localhost:3000/books/3f664012-958f-4be8-8b0a-3fc4278cb59e){BookID}

****