LIBRARY MANAGEMENT SYSTEM

Create a database in postgre SQL:

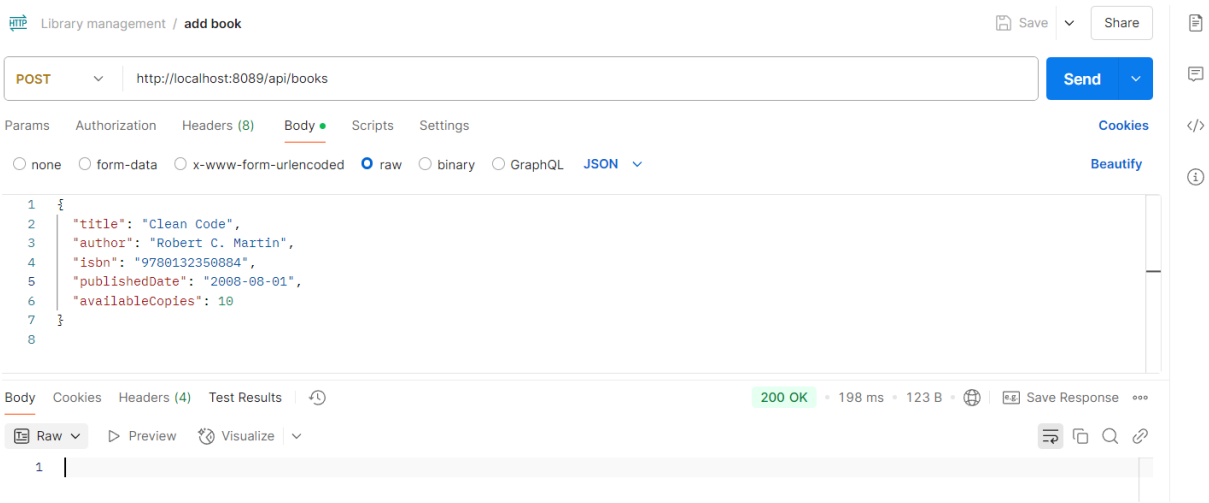
```
C:\Users\thirishaa>psql -U postgres -d postgres
Password for user postgres:

psql (17.2)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=# CREATE DATABASE library_db;
CREATE DATABASE
postgres=# GRANT ALL PRIVILEGES ON DATABASE library_db TO thirishaa;
GRANT
postgres=# psql -U thirishaa -d library_db
postgres-#
```

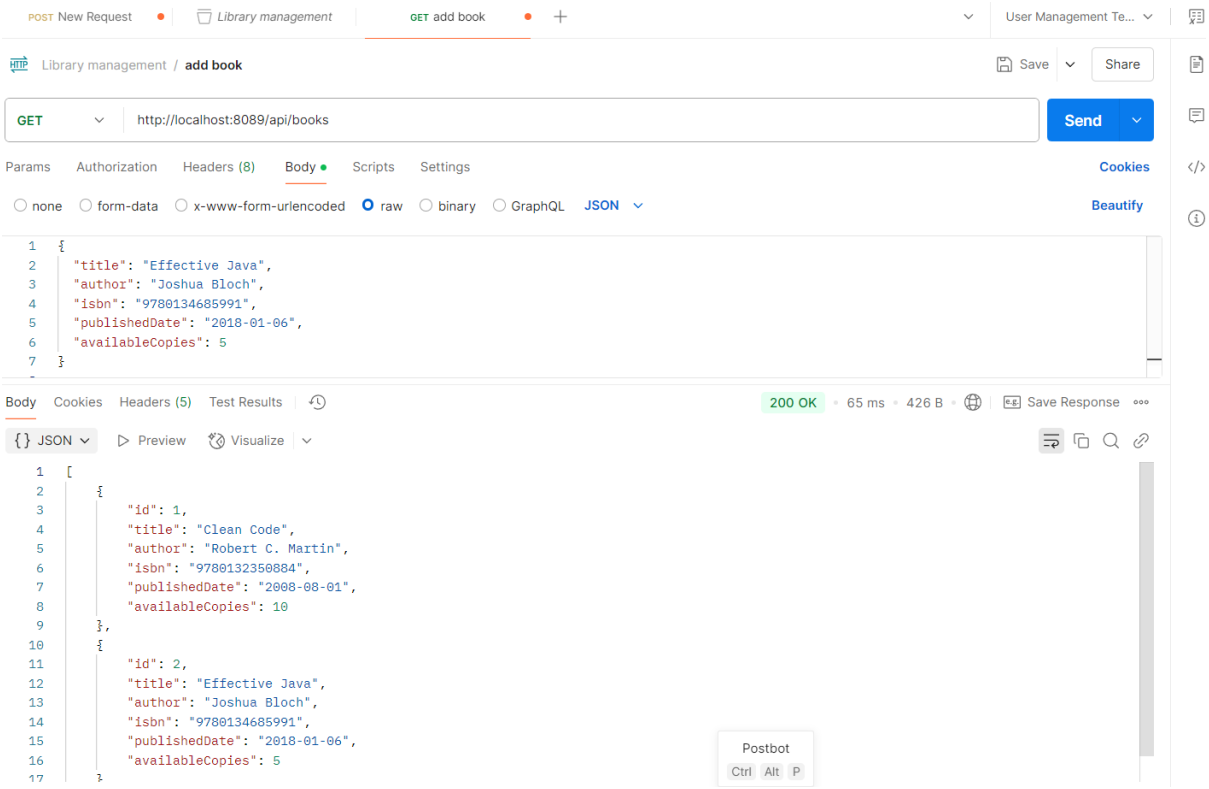
ADD A BOOK :

Description:
Adds a book to the catalog with title, author, ISBN, and available copies.
Expected Result: Status 200 with book ID in response.



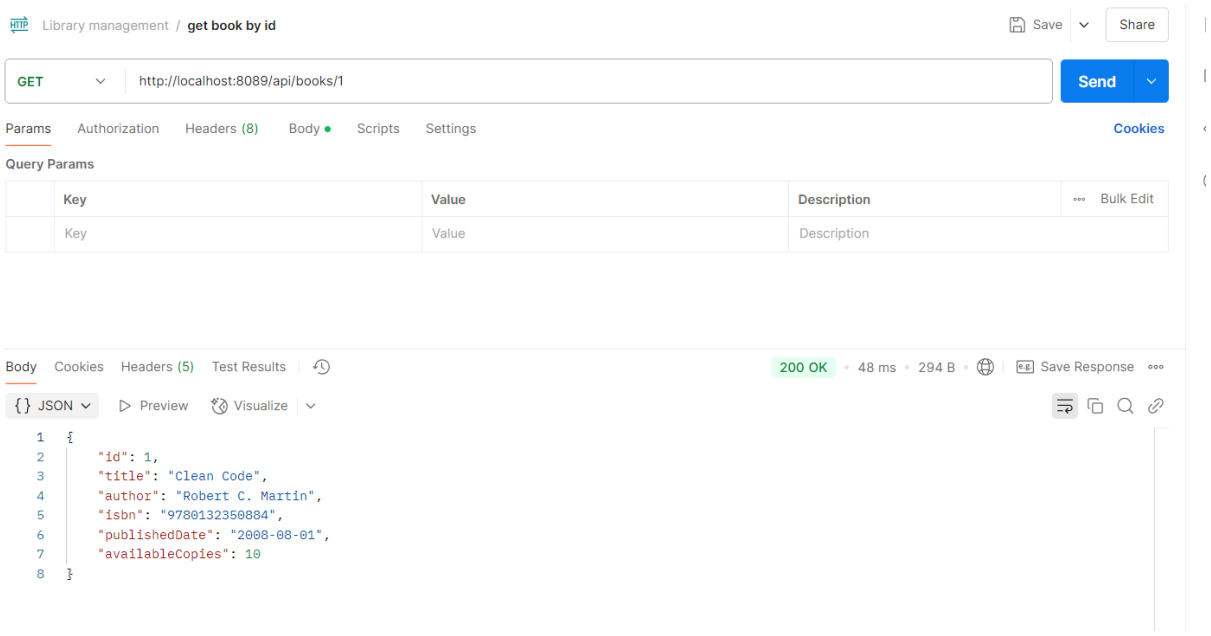
GET ALL BOOKS:

Description:
Returns a list of all books present in the catalog.
Expected Result: JSON array of books.



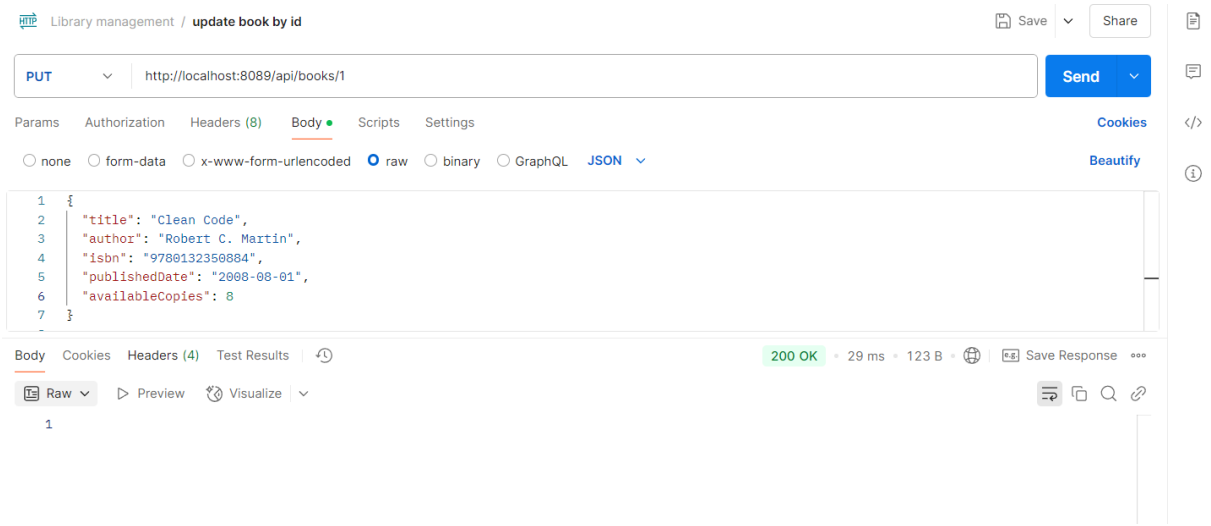
Get book details by ID:

Description:
Fetches a single book based on ID.
Expected Result: Returns book details if exists, else triggers 404.

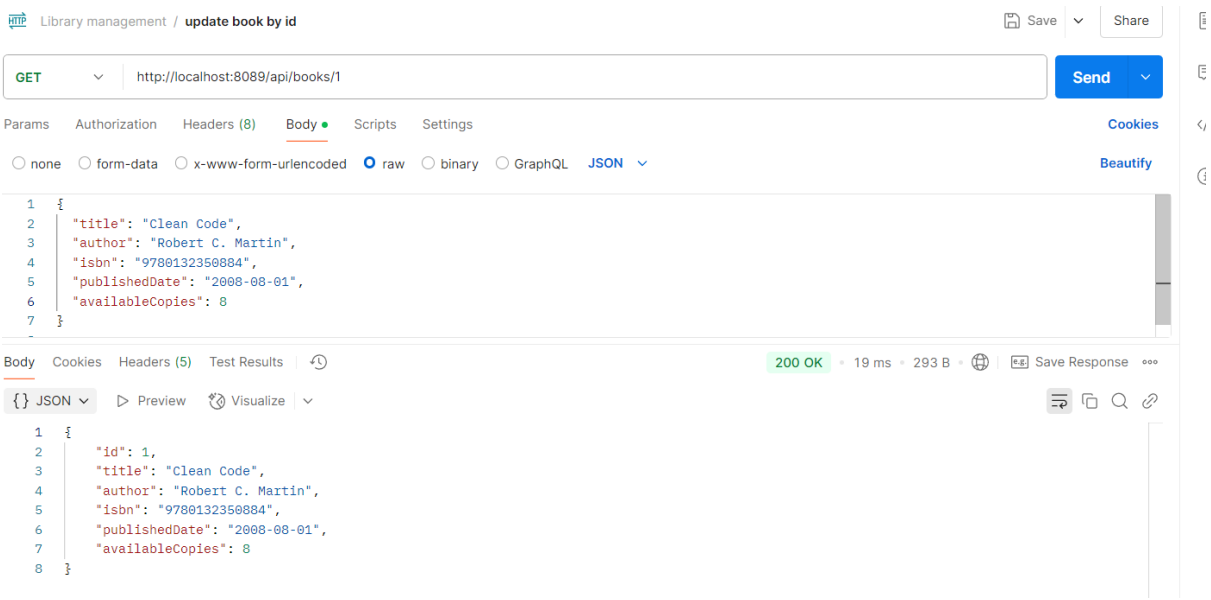


Update book by ID :

Description:
Updates a book's title or available count.
Note: Here available copies changed from 10 to 8.

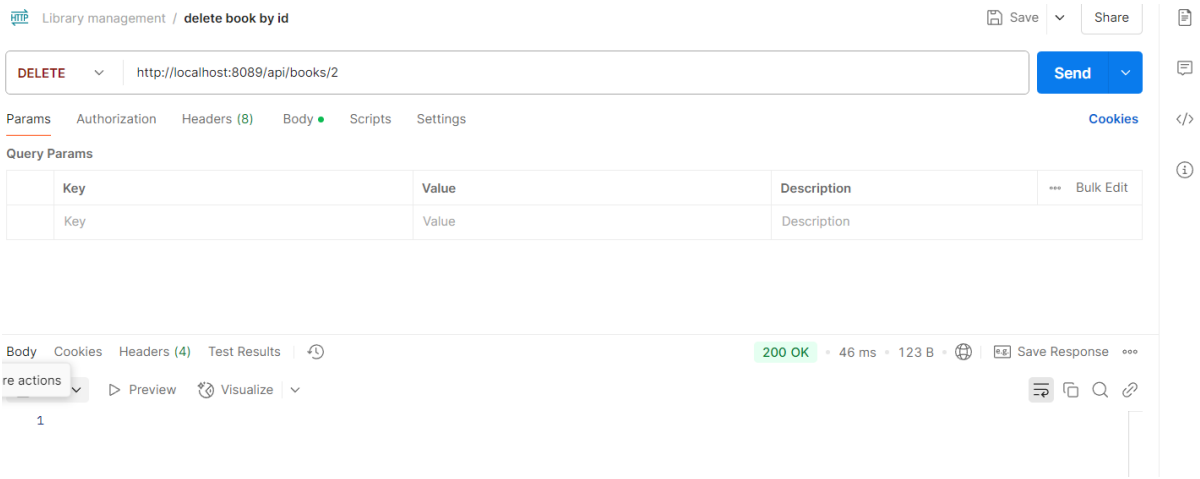


Availabe_count updated from 10 to 8



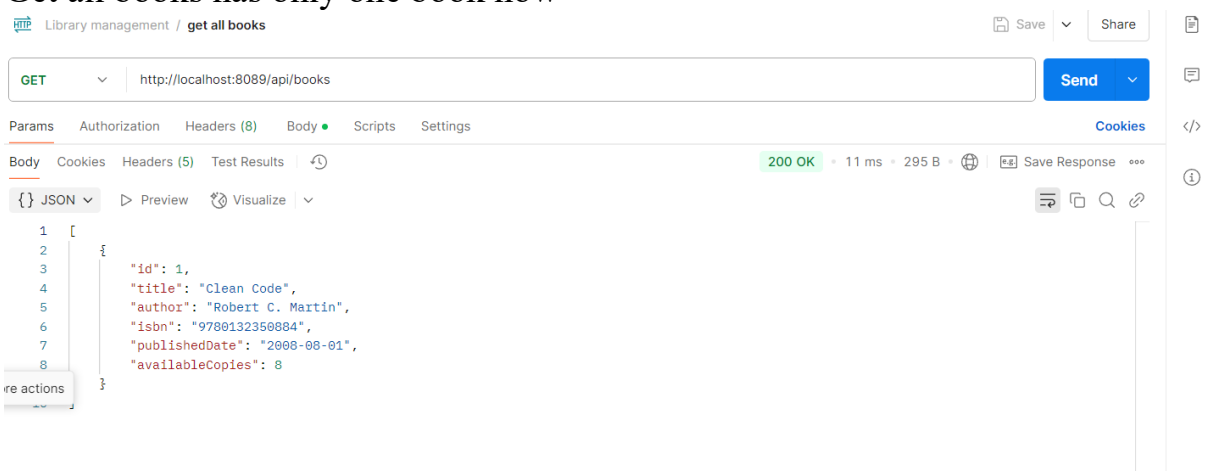
Delete book by id :

Description:
Deletes the book with the specified ID.
Note: After deletion, the number of books decreased.



Deleted the book with id=2:

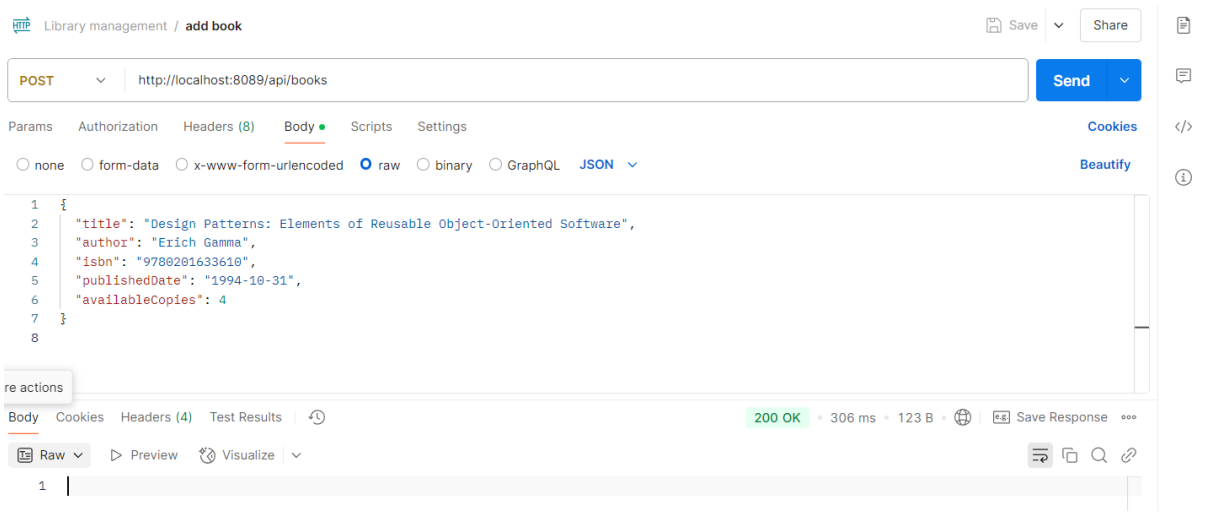
Get all books has only one book now



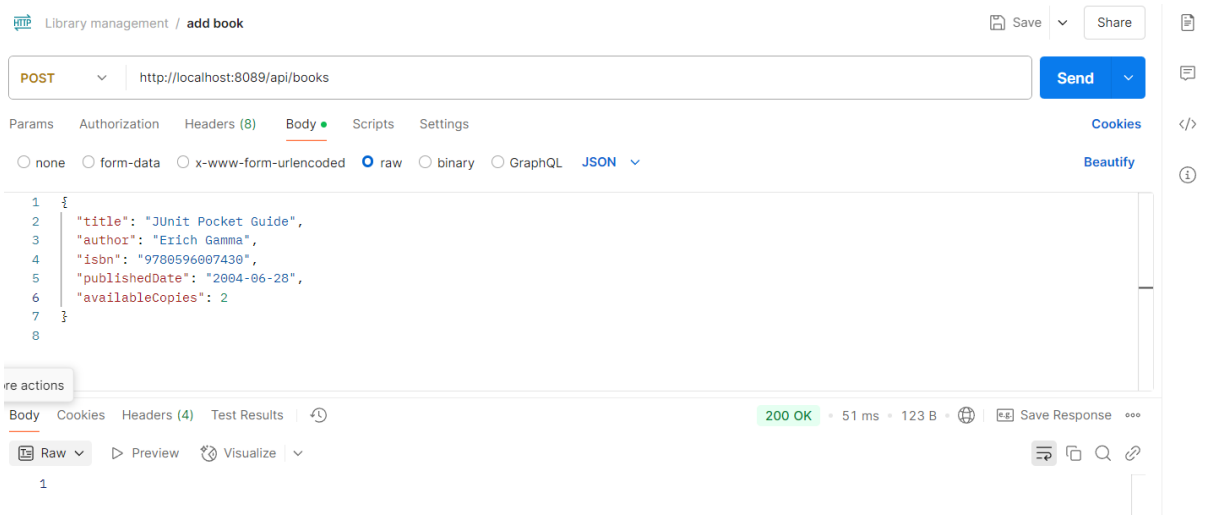

```
library_db=# select * from book;
 id |   title    |      author      |     isbn      | published_date | available_copies
----+------------+------------------+---------------+----------------+------------------
  1 | Clean Code | Robert C. Martin | 9780132350884 | 2008-08-01     |                8
(1 row)
```

Add Multiple Books

Description:
Adds more than one book consecutively.
Expected Result: New books visible in GET /books.



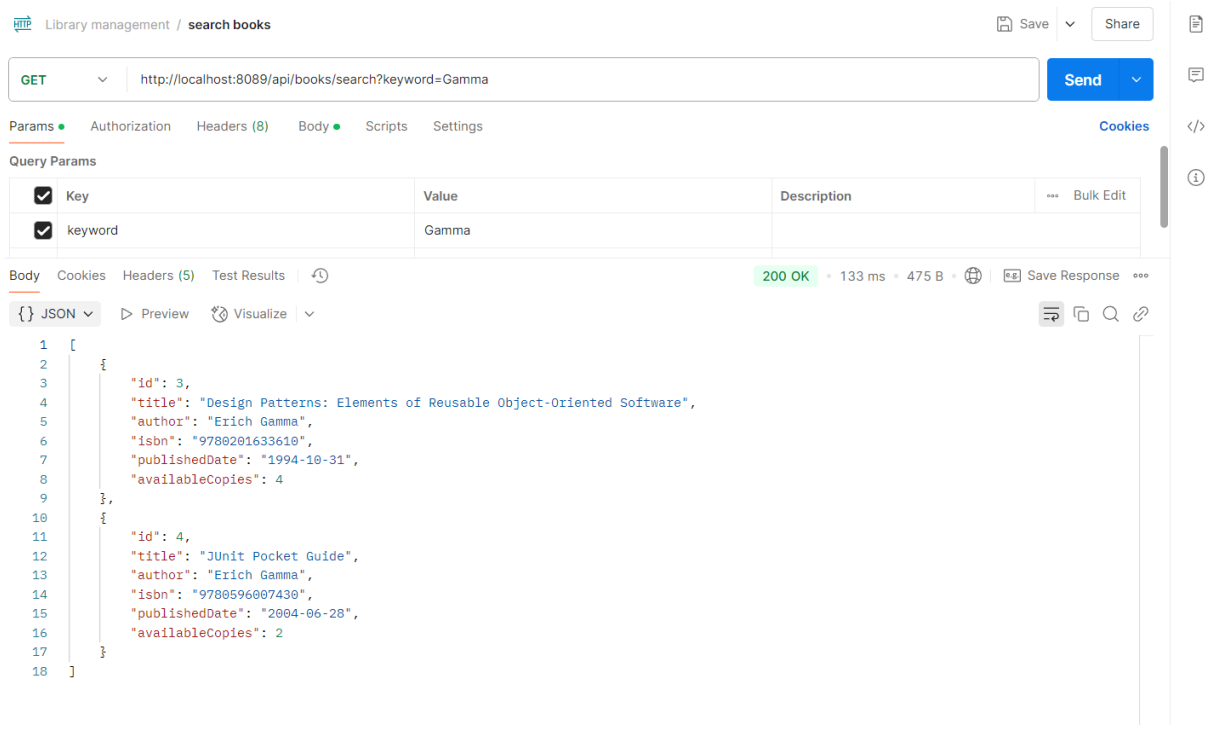

```
library_db=# select * from book;
 id |                          title                           |      author      |     isbn      | published_date | available_copies
----+----------------------------------------------------------+------------------+---------------+----------------+------------------
  1 | Clean Code                                               | Robert C. Martin | 9780132350884 | 2008-08-01     |                8
  3 | Design Patterns: Elements of Reusable Object-Oriented Software | Erich Gamma      | 9780201633610 | 1994-10-31     |                4
(2 rows)
```




```
library_db=# select * from book;
 id |                            title                             |      author      |     isbn      | published_date | available_copies
----+--------------------------------------------------------------+------------------+---------------+----------------+------------------
  1 | Clean Code                                                   | Robert C. Martin | 9780132350884 | 2008-08-01     |                8
  3 | Design Patterns: Elements of Reusable Object-Oriented Software | Erich Gamma    | 9780201633610 | 1994-10-31     |                4
  4 | JUnit Pocket Guide                                           | Erich Gamma      | 9780596007430 | 2004-06-28     |                2
```

🔎 Search Books

Description:
Search by title, author, or ISBN.
Expected Result: Books matching query string.

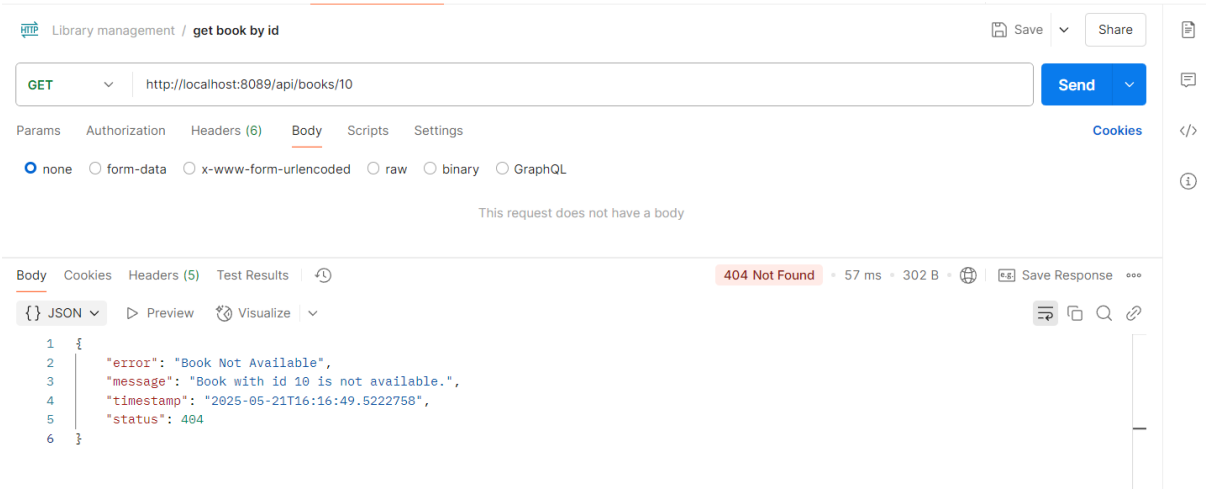


⚠️ Exception Handling: Book Not Found

Trigger:
GET request to non-existing book ID.

Flow:

- Service layer throws BookNotAvailableException
- Global Exception Handler formats 404 response



Error Handling: Book Not Found

When a client sends a GET request to fetch a book by ID (e.g., GET /api/books/10), the system behaves as follows if the book does **not** exist:

1. **Service Layer**
   The service method getBookById(Long id) calls the DAO to find the book. If no book is found (null), it throws a custom exception:

throw new BookNotAvailableException("Book with id " + id + " is not available.");

2. **Global Exception Handler**
   The exception is caught by the global exception handler class (@ControllerAdvice), which formats the error response with HTTP status 404 Not Found and a JSON body containing:
   - timestamp
   - status code (404)
   - error message ("Book Not Available")
   - detailed message from the exception
3. **API Response**
   The client receives a structured error response like this:

{

"timestamp": "2025-05-21T15:56:42.2495785",

"status": 404,

"error": "Book Not Available",

"message": "Book with id 10 is not available."}

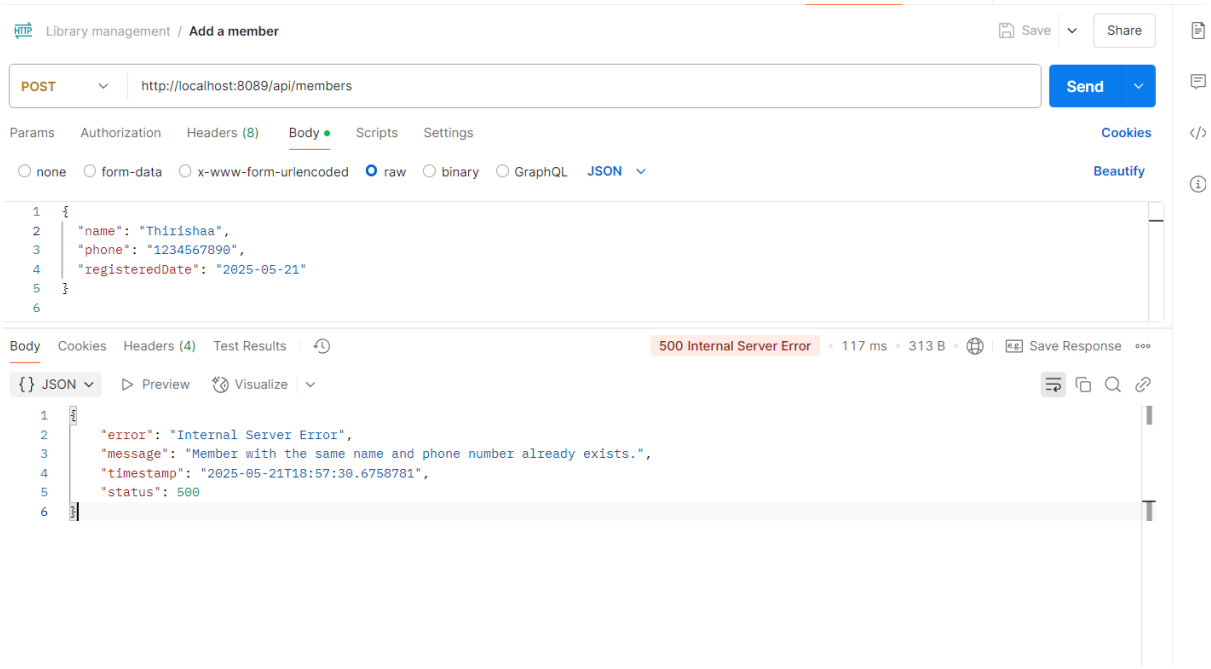
This ensures that missing resources are clearly communicated to API consumers with meaningful HTTP status codes and messages.

👤 Member Operations

✅ Add a Member

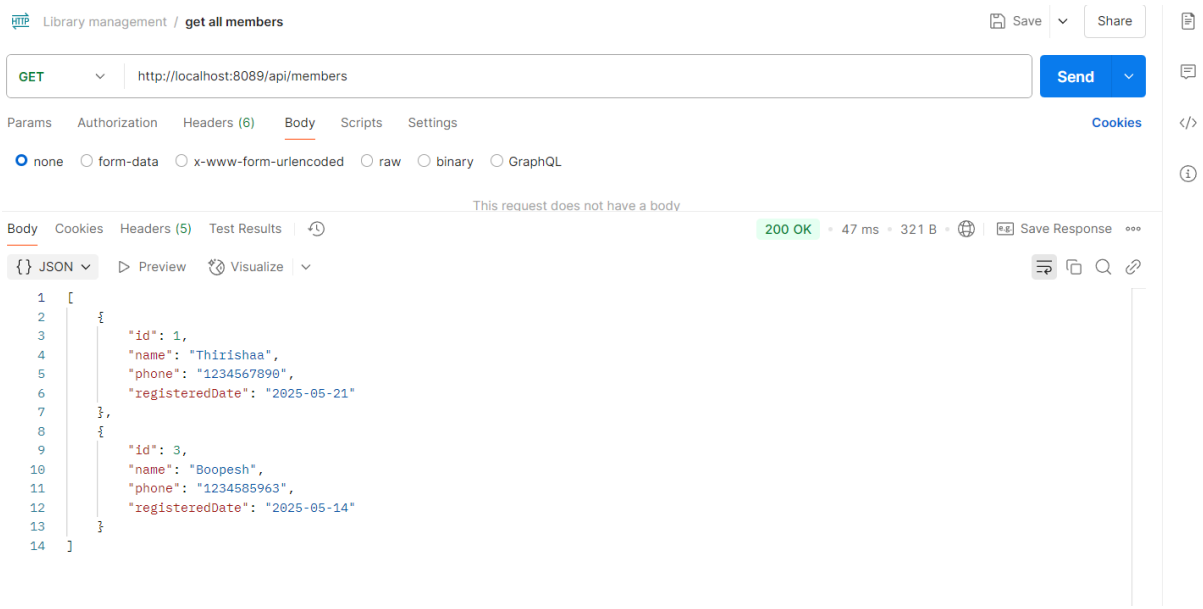
Description:
Adds a member with name, phone, and date.
Expected Result: Status 200 with new member ID.

```
library_db=# select * from member;
 id |    name    |   phone    | registered_date
----+------------+------------+-----------------
  1 | Thirishaa | 1234567890 | 2025-05-21
(1 row)
```



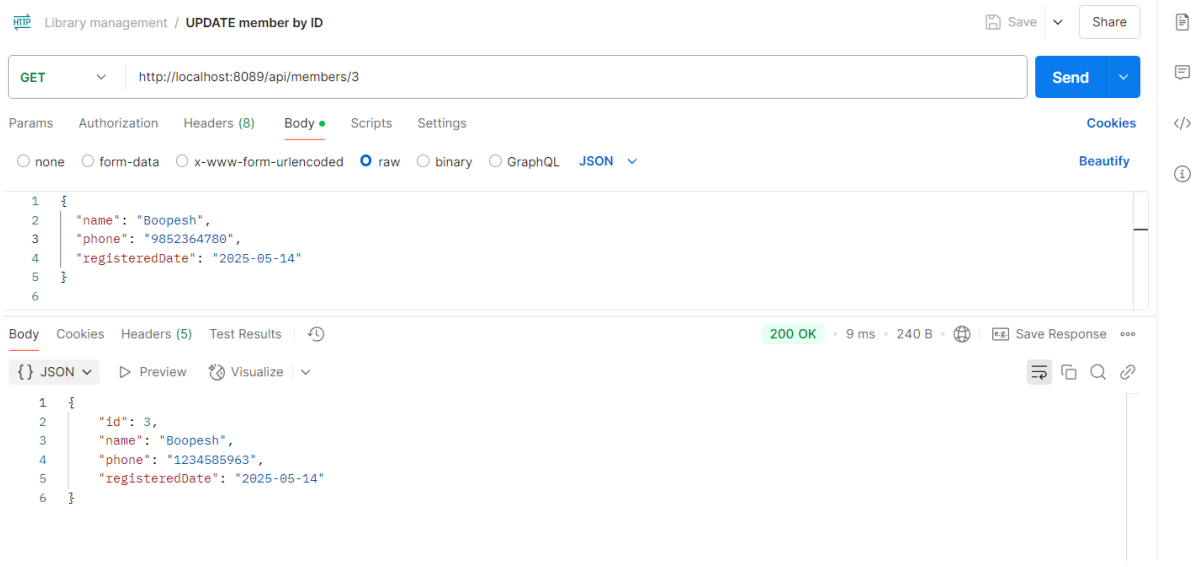
📋 Get All Members

Returns all registered members.



🖍️ Update Member by ID

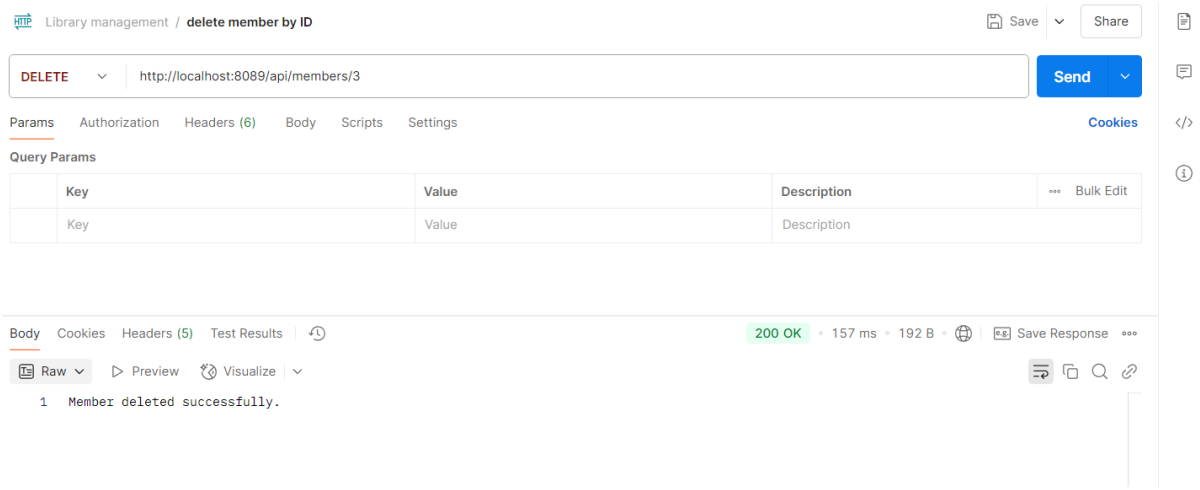
Updates phone or name.




```
library_db=# select * from member;
 id |   name    |   phone    | registered_date
----+-----------+------------+-----------------
  1 | Thirishaa | 1234567890 | 2025-05-21
  3 | Boopesh   | 1234585963 | 2025-05-14
(2 rows)
```

❌ Delete Member by ID

Deletes the member with given ID.




```
library_db=# select * from member;
 id |    name     |    phone     | registered_date
----+-------------+--------------+-----------------
  1 | Thirishaa | 1234567890 | 2025-05-21
(1 row)
```

🔄 Borrow Operations

```
library_db=# select * from boOK;
 id |                                   title                                    |      author      |     isbn      | published_date | available_copies
----+----------------------------------------------------------------------------+------------------+---------------+----------------+------------------
  1 | Clean Code                                                                 | Robert C. Martin | 9780132350884 | 2008-08-01     |                8
  3 | Design Patterns: Elements of Reusable Object-Oriented Software             | Erich Gamma      | 9780201633610 | 1994-10-31     |                4
  4 | JUnit Pocket Guide                                                         | Erich Gamma      | 9780596007430 | 2004-06-28     |                2
(3 rows)


library_db=# select * from member;
 id |    name     |    phone     | registered_date
----+-------------+--------------+-----------------
  1 | Thirishaa | 1234567890 | 2025-05-21
  4 | Dhanvik     | 9876543210 | 2025-05-21
  5 | Dia         | 9876543210 | 2025-05-21
(3 rows)


library_db=# select * from borrow;
 id | member_id | book_id | borrowed_date | due_date
----+-----------+---------+---------------+----------
(0 rows)
```

📥 Borrow a Book

Decreases available copy by 1.

```
library_db=# select * from borrow;
 id | member_id | book_id | borrowed_date |  due_date
----+-----------+---------+---------------+------------
  1 |         4 |       1 | 2025-05-21    | 2025-06-04
(1 row)


library_db=# select * from book;
 id |                          title                           |      author      |     isbn      | published_date | available_copies
----+----------------------------------------------------------+------------------+---------------+----------------+------------------
  3 | Design Patterns: Elements of Reusable Object-Oriented Software | Erich Gamma      | 9780201633610 | 1994-10-31     |                4
  4 | JUnit Pocket Guide                                       | Erich Gamma      | 9780596007430 | 2004-06-28     |                2
  1 | Clean Code                                               | Robert C. Martin | 9780132350884 | 2008-08-01     |                7
(3 rows)
```

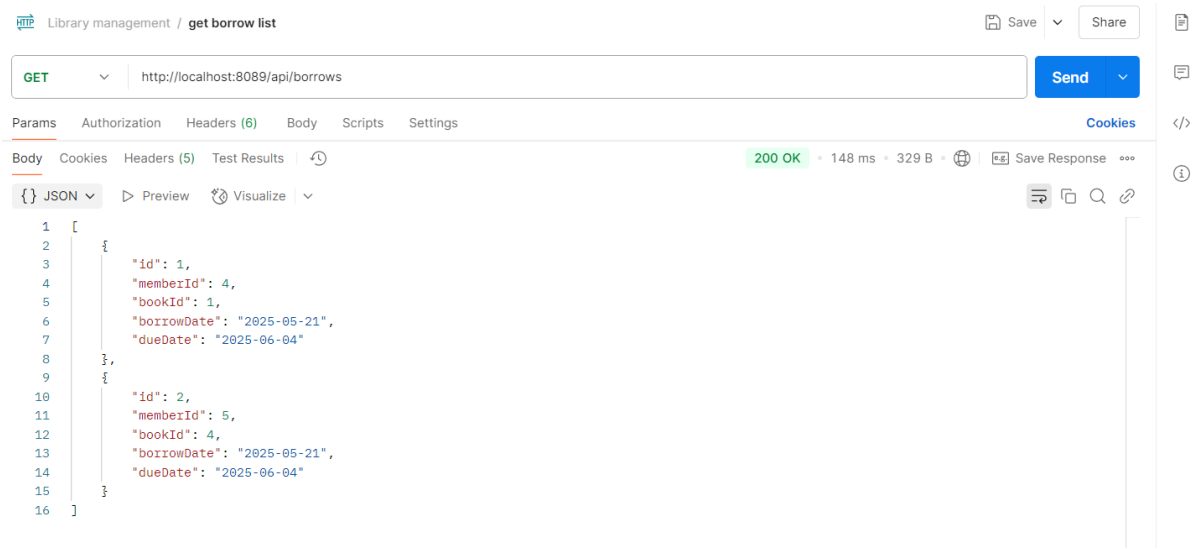
Available copy decreased by 1

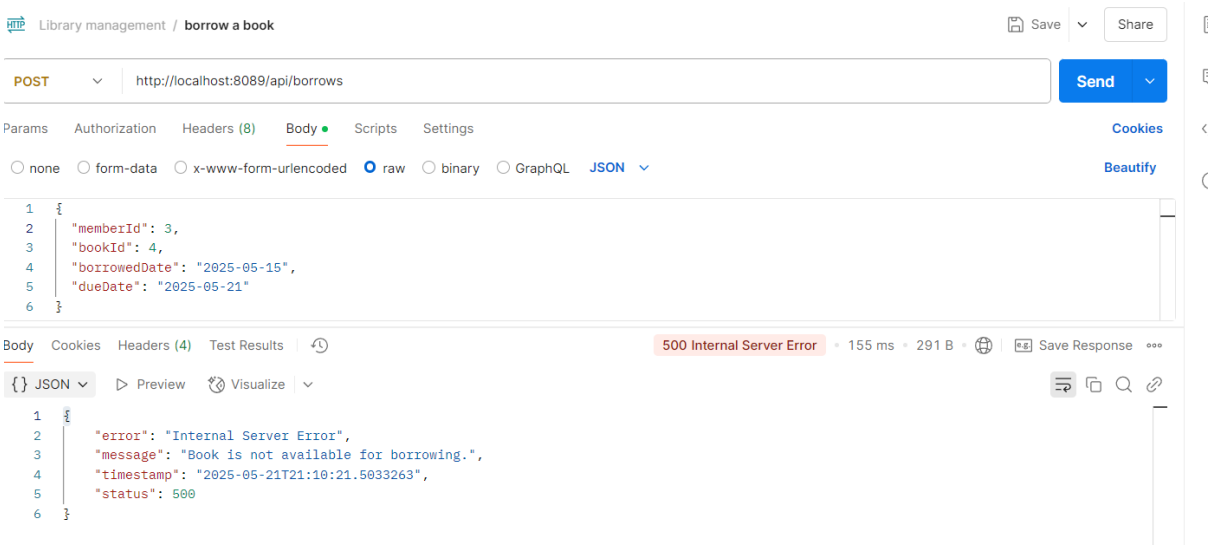
```
library_db=# select * from book;
 id |                          title                           |      author      |     isbn      | published_date | available_copies
----+----------------------------------------------------------+------------------+---------------+----------------+------------------
  3 | Design Patterns: Elements of Reusable Object-Oriented Software | Erich Gamma      | 9780201633610 | 1994-10-31     |                4
  1 | Clean Code                                               | Robert C. Martin | 9780132350884 | 2008-08-01     |                7
  4 | JUnit Pocket Guide                                       | Erich Gamma      | 9780596007430 | 2004-06-28     |                1
(3 rows)


library_db=# select * from borrow;
 id | member_id | book_id | borrowed_date |  due_date
----+-----------+---------+---------------+------------
  1 |         4 |       1 | 2025-05-21    | 2025-06-04
  2 |         5 |       4 | 2025-05-21    | 2025-06-04
(2 rows)
```

📄 Get Borrow List

Shows borrow transactions with book & member info.



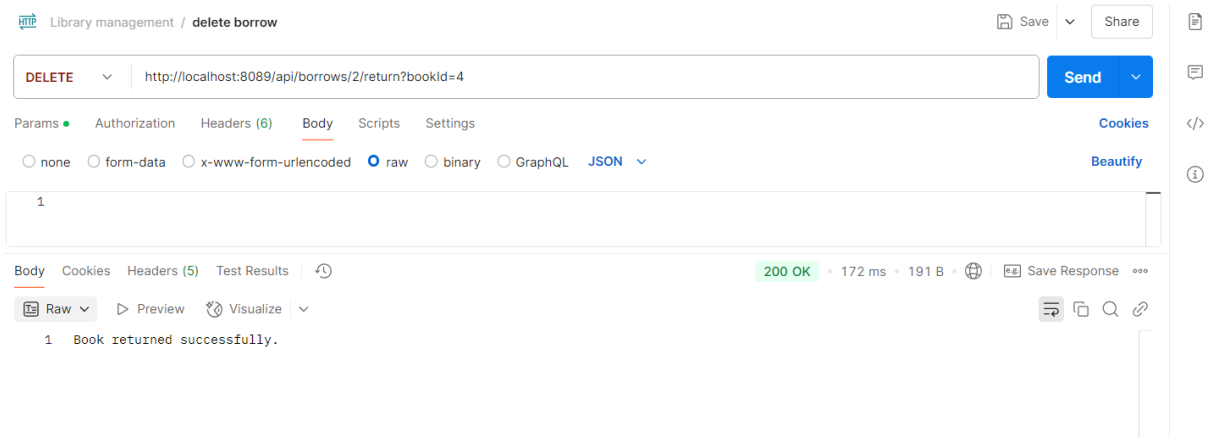



```
library_db=# select * from book;
 id |                            title                             |      author      |     isbn      | published_date | available_copies
----+--------------------------------------------------------------+------------------+---------------+----------------+-----------------
  3 | Design Patterns: Elements of Reusable Object-Oriented Software | Erich Gamma      | 9780201633610 | 1994-10-31     |                4
  1 | Clean Code                                                   | Robert C. Martin | 9780132350884 | 2008-08-01     |                7
  4 | JUnit Pocket Guide                                           | Erich Gamma      | 9780596007430 | 2004-06-28     |                0
(3 rows)


library_db=# select * from borrow;
 id | member_id | book_id | borrowed_date |  due_date
----+-----------+---------+---------------+------------
  1 |         4 |       1 | 2025-05-21    | 2025-06-04
  2 |         5 |       4 | 2025-05-21    | 2025-06-04
  3 |         1 |       4 | 2025-05-21    | 2025-06-04
(3 rows)
```

❌ Delete Borrow Transaction

Deletes borrow using book ID and member ID.
Increments available copies.



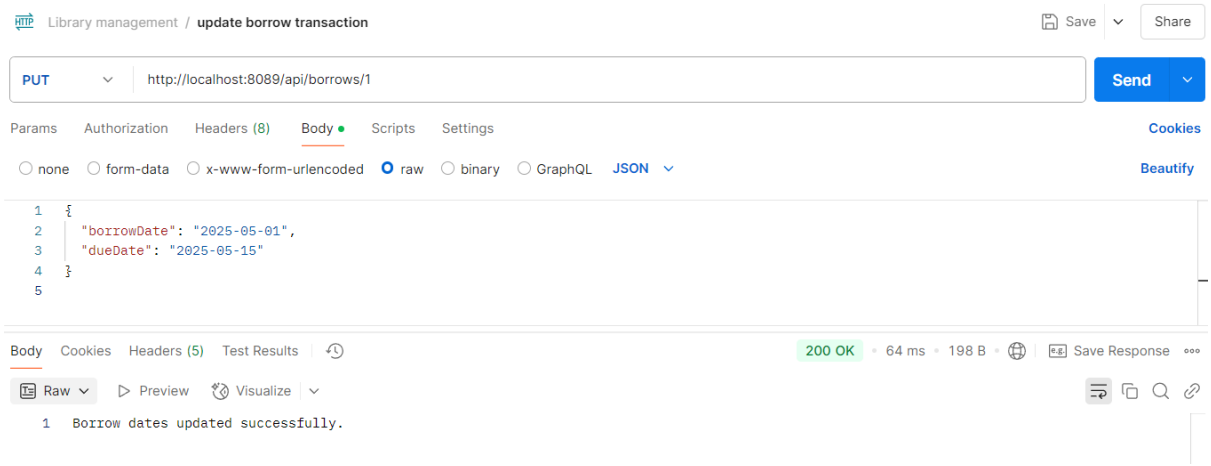
The available copies is incremented by 1 after deleting a borrow transaction of book id=4

```
library_db=# select * from borrow;
 id | member_id | book_id | borrowed_date |  due_date
----+-----------+---------+---------------+------------
  1 |         4 |       1 | 2025-05-21    | 2025-06-04
  3 |         1 |       4 | 2025-05-21    | 2025-06-04
(2 rows)


library_db=# select * from book;
 id |                                  title                                  |      author      |     isbn      | published_date | available_copies
----+-------------------------------------------------------------------------+------------------+---------------+----------------+------------------
  3 | Design Patterns: Elements of Reusable Object-Oriented Software          | Erich Gamma      | 9780201633610 | 1994-10-31     |                4
  1 | Clean Code                                                              | Robert C. Martin | 9780132350884 | 2008-08-01     |                7
  4 | JUnit Pocket Guide                                                      | Erich Gamma      | 9780596007430 | 2004-06-28     |                1
(3 rows)
```

📝 Update Borrow Record

Before and after screenshots showing date updates or member changes.

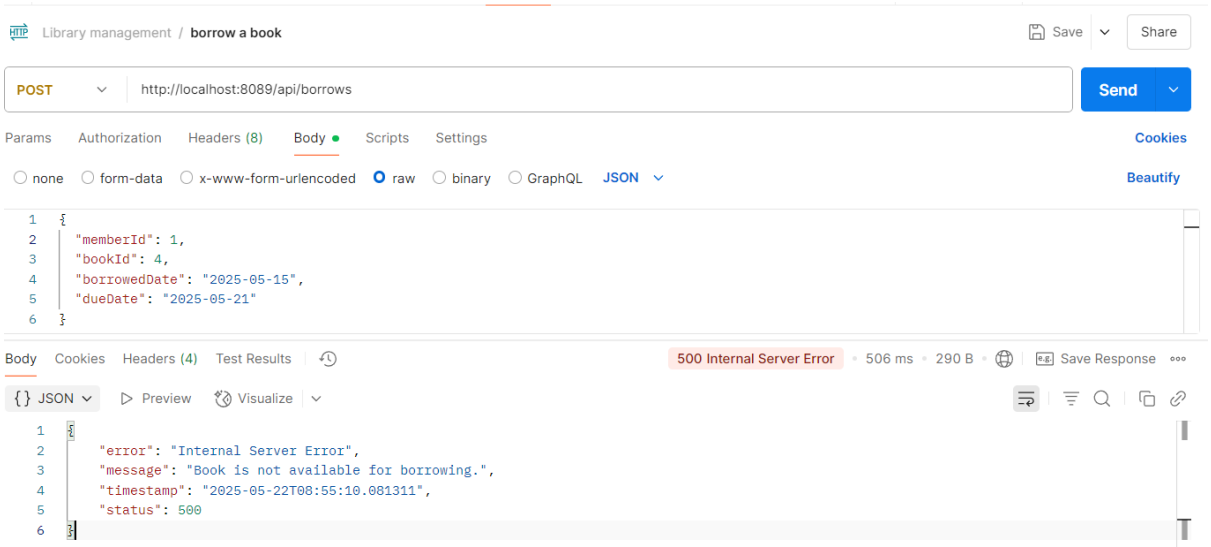


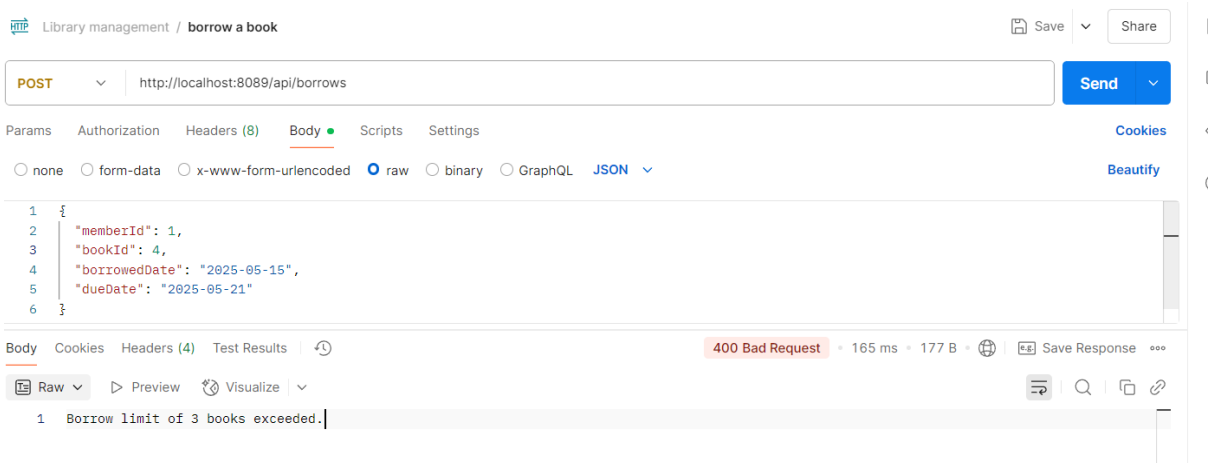
Before update:

```
library_db=# select * from borrow;
 id | member_id | book_id | borrowed_date |  due_date
----+-----------+---------+---------------+------------
  1 |         4 |       1 | 2025-05-21    | 2025-06-04
  3 |         1 |       4 | 2025-05-21    | 2025-06-04
(2 rows)
```

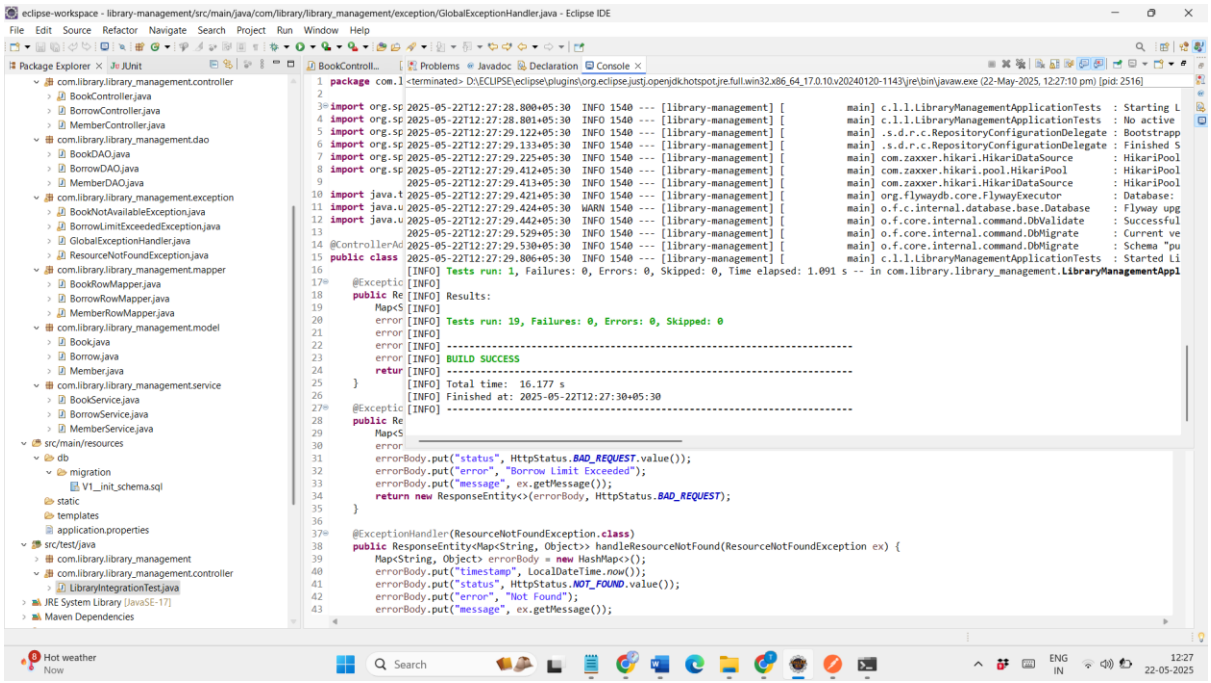
After update :

```
library_db=# select * from borrow;
 id | member_id | book_id | borrowed_date |  due_date
----+-----------+---------+---------------+------------
  3 |         1 |       4 | 2025-05-21    | 2025-06-04
  1 |         4 |       1 | 2025-05-01    | 2025-05-15
(2 rows)
```


Library management / borrow a book
Save
Share
POST
http://localhost:8089/api/borrows
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"memberId": 1,
"bookId": 4,
"borrowedDate": "2025-05-15",
"dueDate": "2025-05-21"
}
Body
Cookies
Headers (4)
Test Results
500 Internal Server Error
506 ms
290 B
Save Response
JSON
Preview
Visualize
{
"error": "Internal Server Error",
"message": "Book is not available for borrowing.",
"timestamp": "2025-05-22T08:55:10.081311",
"status": 500
}


Library management / borrow a book
Save
Share
POST
http://localhost:8089/api/borrows
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"memberId": 1,
"bookId": 4,
"borrowedDate": "2025-05-15",
"dueDate": "2025-05-21"
}
Body
Cookies
Headers (4)
Test Results
400 Bad Request
165 ms
177 B
Save Response
Raw
Preview
Visualize
Borrow limit of 3 books exceeded.

Integration Testing :



All API endpoints were tested using Postman and the backend behaved as expected, including validation and exception responses.