

LIBRARY MANAGEMENT SYSTEM

Create a database in postgres 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.

The screenshot shows the Postman application interface. The top bar indicates the URL is `http://localhost:8089/api/books`. The method dropdown shows `POST`. The body tab contains a JSON payload for a new book:

```
1 {
2   "title": "Clean Code",
3   "author": "Robert C. Martin",
4   "isbn": "9780132350884",
5   "publishedDate": "2008-08-01",
6   "availableCopies": 10
7 }
```

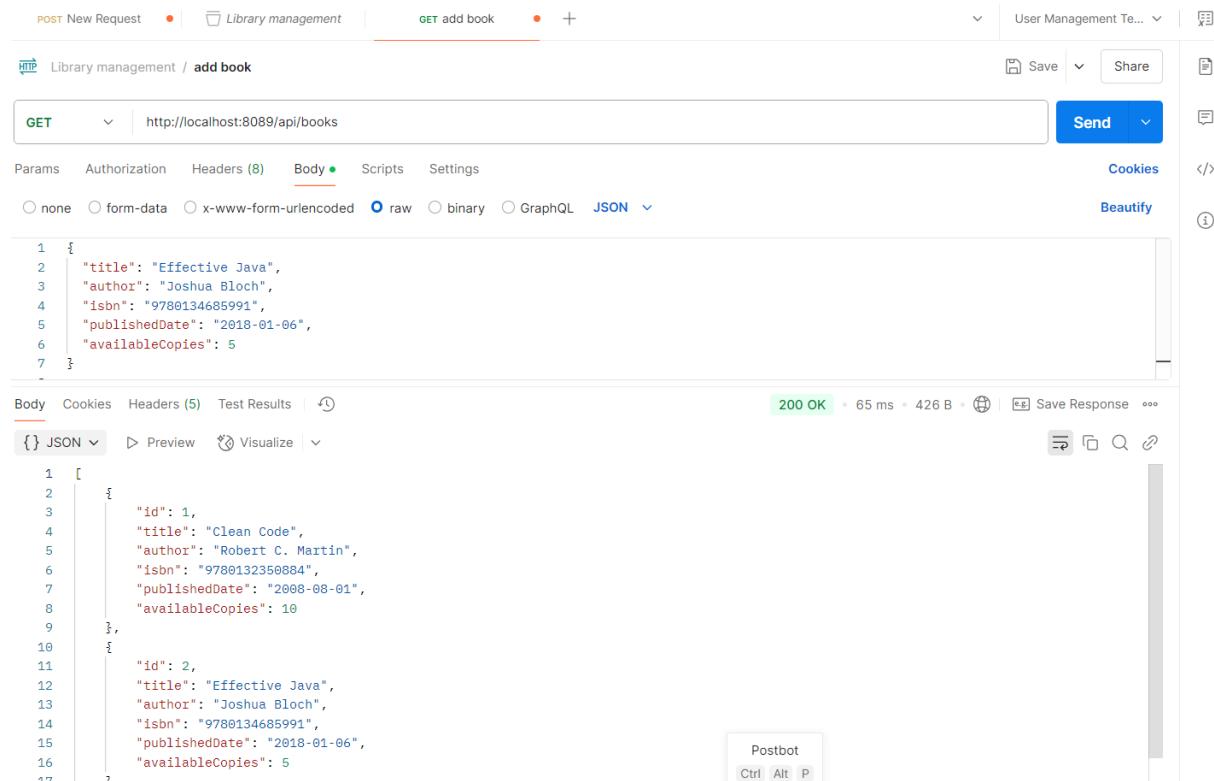
The bottom status bar shows a successful response: `200 OK`, `198 ms`, `123 B`.

GET ALL BOOKS:

Description:

Returns a list of all books present in the catalog.

Expected Result: JSON array of books.



The screenshot shows the Postman interface with a GET request to `http://localhost:8089/api/books`. The request body is a JSON object:

```
1 {
2   "title": "Effective Java",
3   "author": "Joshua Bloch",
4   "isbn": "9780134685991",
5   "publishedDate": "2018-01-06",
6   "availableCopies": 5
7 }
```

The response is a 200 OK status with a JSON array of two book objects:

```
1 [
2   {
3     "id": 1,
4     "title": "Clean Code",
5     "author": "Robert C. Martin",
6     "isbn": "9780132350884",
7     "publishedDate": "2008-08-01",
8     "availableCopies": 10
9   },
10   {
11     "id": 2,
12     "title": "Effective Java",
13     "author": "Joshua Bloch",
14     "isbn": "9780134685991",
15     "publishedDate": "2018-01-06",
16     "availableCopies": 5
17   }
]
```

Get book details by ID:

Description:

Fetches a single book based on ID.

Expected Result: Returns book details if exists, else triggers 404.

The screenshot shows a Postman request for a GET operation on the URL `http://localhost:8089/api/books/1`. The request includes a parameter named "Key" with the value "Description". The response is a 200 OK status with a JSON payload:

```
1 {  
2   "id": 1,  
3   "title": "Clean Code",  
4   "author": "Robert C. Martin",  
5   "isbn": "9780132350884",  
6   "publishedDate": "2008-08-01",  
7   "availableCopies": 10  
8 }
```

Update book by ID :

Description:

Updates a book's title or available count.

Note: Here available copies changed from 10 to 8.

The screenshot shows a Postman request for a PUT operation on the URL `http://localhost:8089/api/books/1`. The request includes a parameter named "Key" with the value "availableCopies" and a value of 8. The response is a 200 OK status with a JSON payload:

```
1 {  
2   "id": 1,  
3   "title": "Clean Code",  
4   "author": "Robert C. Martin",  
5   "isbn": "9780132350884",  
6   "publishedDate": "2008-08-01",  
7   "availableCopies": 8  
8 }
```

Available_count updated from 10 to 8

The screenshot shows a POST request to `http://localhost:8089/api/books/1`. The request body is a JSON object with the following content:

```
1 {
2   "title": "Clean Code",
3   "author": "Robert C. Martin",
4   "isbn": "9780132350884",
5   "publishedDate": "2008-08-01",
6   "availableCopies": 8
7 }
```

The response status is 200 OK, with a response time of 19 ms and a response size of 293 B. The response body is identical to the request body.

Delete book by id :

Description:

Deletes the book with the specified ID.

Note: After deletion, the number of books decreased.

The screenshot shows a DELETE request to `http://localhost:8089/api/books/2`. The request body is empty. The response status is 200 OK, with a response time of 46 ms and a response size of 123 B. The response body contains the number '1'.

Deleted the book with id=2:

Get all books has only one book now

HTTP Library management / get all books

GET http://localhost:8089/api/books

Params Authorization Headers (8) Body Scripts Settings

Body Cookies Headers (5) Test Results

200 OK 11 ms 295 B Save Response

```
[{"id": 1, "title": "Clean Code", "author": "Robert C. Martin", "isbn": "9780132350884", "publishedDate": "2008-08-01", "availableCopies": 8}]
```

```
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.

HTTP Library management / add book

POST http://localhost:8089/api/books

Params Authorization Headers (8) Body Scripts Settings

Body none form-data x-www-form-urlencoded raw binary GraphQL JSON

1 {"title": "Design Patterns: Elements of Reusable Object-Oriented Software", "author": "Erich Gamma", "isbn": "9780201633610", "publishedDate": "1994-10-31", "availableCopies": 4}

200 OK 306 ms 123 B Save Response

Body Cookies Headers (4) Test Results

Raw Preview Visualize

```
1 |
```

```
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)
```

HTTP Library management / add book

POST http://localhost:8089/api/books

Params Authorization Headers (8) Body Scripts Settings Cookies </> Beautify

Body none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "title": "JUnit Pocket Guide",
3   "author": "Erich Gamma",
4   "isbn": "9780596007430",
5   "publishedDate": "2004-06-28",
6   "availableCopies": 2
7 }
8

```

Reactions Body Cookies Headers (4) Test Results

Raw Preview Visualize

200 OK 51 ms 123 B Save Response

```

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.

HTTP Library management / search books

GET http://localhost:8089/api/books/search?keyword=Gamma

Params Authorization Headers (8) Body Scripts Settings Cookies </>

Query Params

Key	Value	Description	Bulk Edit
<input checked="" type="checkbox"/> keyword	Gamma		

Body Cookies Headers (5) Test Results

{} JSON Preview Visualize

200 OK 133 ms 475 B Save Response

```

1 [
2   {
3     "id": 3,
4     "title": "Design Patterns: Elements of Reusable Object-Oriented Software",
5     "author": "Erich Gamma",
6     "isbn": "9780201633610",
7     "publishedDate": "1994-10-31",
8     "availableCopies": 4
9   },
10  {
11    "id": 4,
12    "title": "JUnit Pocket Guide",
13    "author": "Erich Gamma",
14    "isbn": "9780596007430",
15    "publishedDate": "2004-06-28",
16    "availableCopies": 2
17  }
18 ]

```

⚠ Exception Handling: Book Not Found

Trigger:

GET request to non-existing book ID.

Flow:

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

The screenshot shows a Postman interface. At the top, it says "HTTP Library management / get book by id". Below that, a "GET" method is selected with the URL "http://localhost:8089/api/books/10". The "Body" tab is active, showing the message "This request does not have a body". In the results section, the status is "404 Not Found" with a timestamp of "2025-05-21T16:16:49.5222758", a status code of "404", and an error message of "Book Not Available". The message also states "Book with id 10 is not available.".

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

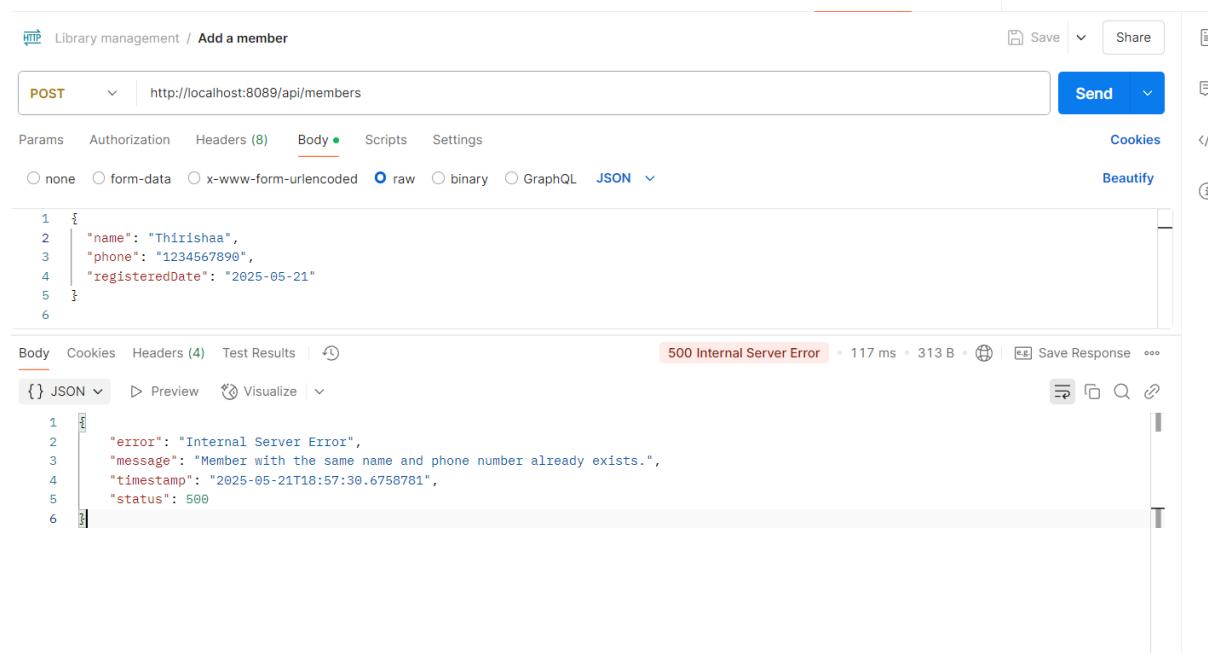
✓ Add a Member

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)
```



The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** http://localhost:8089/api/members
- Body:** Raw JSON (selected) containing:

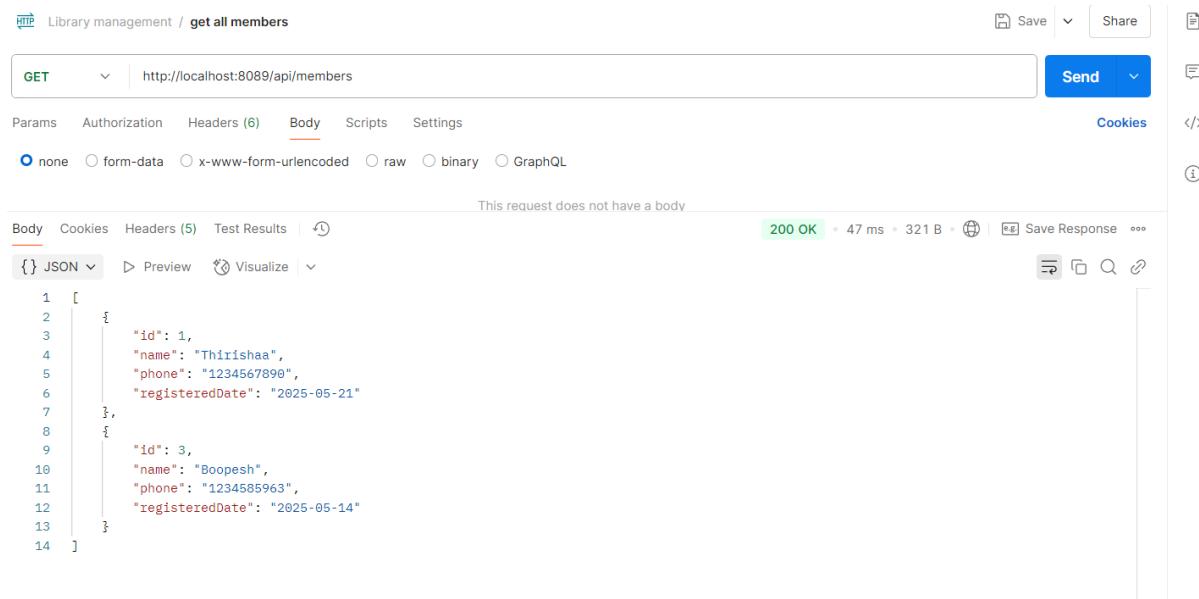
```
1 {
2   "name": "Thirishaa",
3   "phone": "1234567890",
4   "registeredDate": "2025-05-21"
5 }
```

- Headers:** (8) including Content-Type: application/json
- Response:** Status: 500 Internal Server Error, Time: 117 ms, Size: 313 B. The response body is:

```
1 {
2   "error": "Internal Server Error",
3   "message": "Member with the same name and phone number already exists.",
4   "timestamp": "2025-05-21T18:57:30.6758781",
5   "status": 500
6 }
```

Get All Members

Returns all registered members.



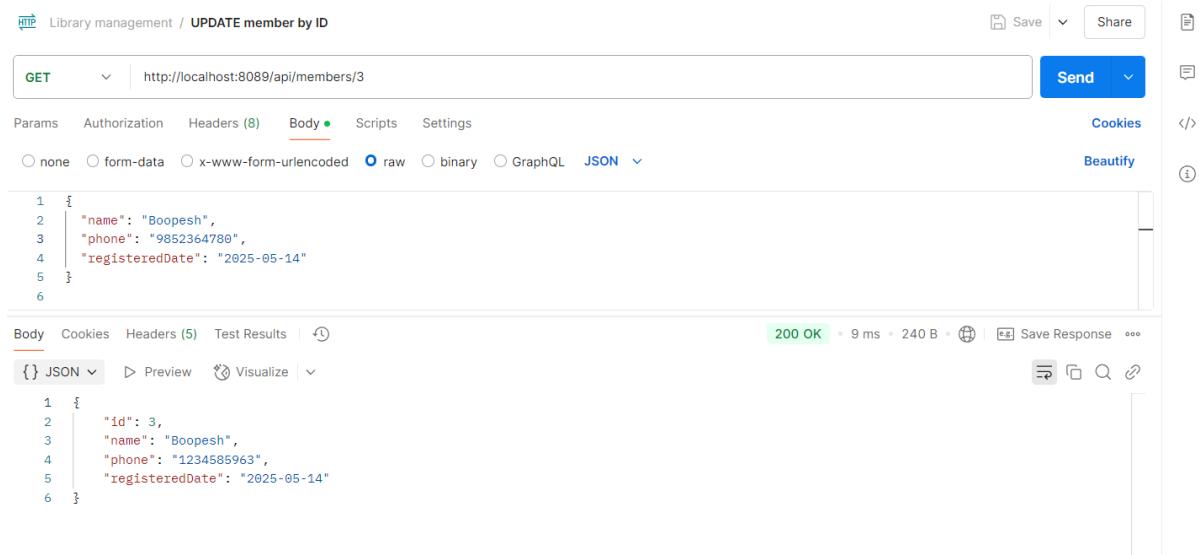
The screenshot shows the Postman interface with the following details:

- Method:** GET
- URL:** http://localhost:8089/api/members
- Body:** This request does not have a body.
- Response:** 200 OK (47 ms, 321 B)
 - Save Response
- Code Snippet (JSON):**

```
1 [  
2   {  
3     "id": 1,  
4     "name": "Thirishaa",  
5     "phone": "1234567890",  
6     "registeredDate": "2025-05-21"  
7   },  
8   {  
9     "id": 3,  
10    "name": "Boopesh",  
11    "phone": "1234585963",  
12    "registeredDate": "2025-05-14"  
13  }  
]
```

Update Member by ID

Updates phone or name.



The screenshot shows the Postman interface with the following details:

- Method:** GET
- URL:** http://localhost:8089/api/members/3
- Body:** (raw JSON)

```
1 {  
2   "name": "Boopesh",  
3   "phone": "9852364780",  
4   "registeredDate": "2025-05-14"  
5 }  
6
```
- Response:** 200 OK (9 ms, 240 B)
 - Save Response
- Code Snippet (JSON):**

```
1 {  
2   "id": 3,  
3   "name": "Boopesh",  
4   "phone": "1234585963",  
5   "registeredDate": "2025-05-14"  
6 }
```

```
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.

The screenshot shows a Postman interface. The URL is `http://localhost:8089/api/members/3`. The response status is `200 OK` with a time of `157 ms` and a size of `192 B`. The response body contains the message `Member deleted successfully.`.

```
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)
```

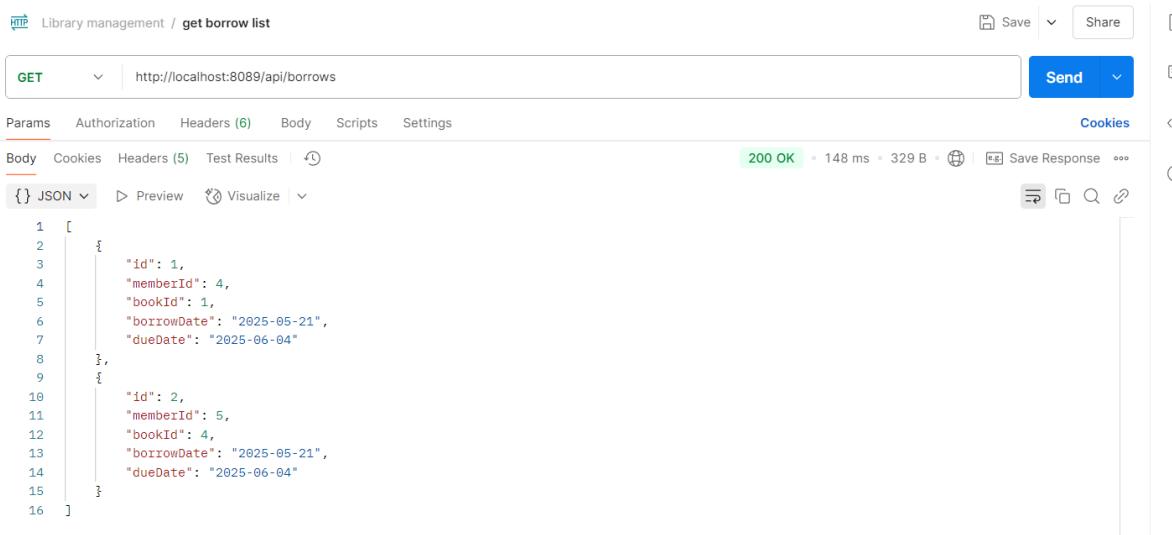
Available copy decreased by 1

```
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.



The screenshot shows a Postman API client interface. The URL is `http://localhost:8089/api/borrows`. The response status is `200 OK` with a response time of `148 ms` and a size of `329 B`. The response body is displayed as JSON:

```
1 [ 
2   { 
3     "id": 1,
4     "memberId": 4,
5     "bookId": 1,
6     "borrowDate": "2025-05-21",
7     "dueDate": "2025-06-04"
8   },
9   { 
10    "id": 2,
11    "memberId": 5,
12    "bookId": 4,
13    "borrowDate": "2025-05-21",
14    "dueDate": "2025-06-04"
15  }
]
```

HTTP Library management / borrow a book

POST http://localhost:8089/api/borrows

Params Authorization Headers (8) Body Scripts Settings Cookies Beautify

Body none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "memberId": 3,
3   "bookId": 4,
4   "borrowedDate": "2025-05-15",
5   "dueDate": "2025-05-21"
6 }
```

Body Cookies Headers (4) Test Results | 500 Internal Server Error 155 ms 291 B Save Response

{ } JSON Preview Visualize

```

1 {
2   "error": "Internal Server Error",
3   "message": "Book is not available for borrowing.",
4   "timestamp": "2025-05-21T21:10:21.5033263",
5   "status": 500
6 }
```

```

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.

HTTP Library management / delete borrow

DELETE http://localhost:8089/api/borrows/2/return?bookId=4

Params Authorization Headers (6) Body Scripts Settings Cookies Beautify

Body none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1
```

Body Cookies Headers (5) Test Results | 200 OK 172 ms 191 B Save Response

Raw Preview Visualize

1 Book returned successfully.

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.

The screenshot shows a POSTman interface with the following details:

- Method: PUT
- URL: <http://localhost:8089/api/borrows/1>
- Headers: (8) - includes Authorization, Headers, Body, Scripts, Settings, Cookies, and Beautify.
- Body: Raw JSON payload:

```
1 {
2   "borrowDate": "2025-05-01",
3   "dueDate": "2025-05-15"
4 }
```
- Response: 200 OK, 64 ms, 198 B, Save Response, etc.
- Content: Borrow dates updated successfully.

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)
```

HTTP Library management / borrow a book

POST http://localhost:8089/api/borrows

Send

Params Authorization Headers (8) Body Scripts Settings Cookies Beautify

None form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {  
2   "memberId": 1,  
3   "bookId": 4,  
4   "borrowedDate": "2025-05-15",  
5   "dueDate": "2025-05-21"  
6 }
```

Body Cookies Headers (4) Test Results

500 Internal Server Error 506 ms 290 B Save Response

{ } JSON Preview Visualize

```
1 {  
2   "error": "Internal Server Error",  
3   "message": "Book is not available for borrowing.",  
4   "timestamp": "2025-05-22T08:55:10.081311",  
5   "status": 500  
6 }
```

HTTP Library management / borrow a book

POST http://localhost:8089/api/borrows

Send

Params Authorization Headers (8) Body Scripts Settings Cookies Beautify

None form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {  
2   "memberId": 1,  
3   "bookId": 4,  
4   "borrowedDate": "2025-05-15",  
5   "dueDate": "2025-05-21"  
6 }
```

Body Cookies Headers (4) Test Results

400 Bad Request 165 ms 177 B Save Response

Raw Preview Visualize

```
1 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.