

## **DATABASE**

### **Group 7**

- **Akhtar Gadang Abimanyu**
- **Faiq Athaya Urumsah**
- **Farrel Tsaqif**

## **LIBRARY BORROWING TRACKER**

The code is in the [server.js](#). In this file, we want to focus more on showing the working operations.

<b>CRUD operations</b>	<b>3</b>
1. CREATE	3
- Register Borrower	3
- Register Librarian and login to get token	4
- Add Book but we need authorization token from librarian	5
2. READ	6
- Get Books	6
3. UPDATE	7
- Process of borrowing a book	7
4. DELETE	9
- Deleting a book and its history	9

# CRUD operations

## 1. CREATE

### - Register Borrower

HTTP <http://localhost:5000/borrower/register>

POST <http://localhost:5000/borrower/register>

Docs Params Authorization Headers (8) Body Scripts Settings

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL [JSON](#)

```
1 {  
2   "borrowerName": "Test User",  
3   "borrowerEmail": "test@example.com",  
4   "borrowerPass": "password123"  
5 }
```

Body Cookies Headers (8) Test Results

{ } JSON ▾ ▶ Preview Visualize

```
1 {  
2   "message": "Borrower registered"  
3 }
```

	userID	borrowerName	borrowerEmail	borrowerPass
▶	1	Test User	test@example.com	\$2b\$10\$d/kIBgwInEY6KHRHZQ4BbetB2hiqQISp...
*	NULL	NULL	NULL	NULL

- Register Librarian and login to get token

The screenshot shows the Postman interface with two requests:

- POST http://localhost:5000/librarian/register**
- POST http://localhost:5000/librarian/login**

**Request 1: Register Librarian**

**Body (JSON):**

```

1 {
2   "librarianName": "Boss",
3   "librarianEmail": "boss@lib.com",
4   "librarianPass": "123"
5 }
```

**Response 1: 200 OK**

**Body (JSON):**

```

1 {
2   "message": "Librarian registered"
3 }
```

**Table Data:**

	librarianID	librarianName	librarianEmail	librarianPass
▶	1	Boss	boss@lib.com	\$2b\$10\$Pq0BjdhFJ3dd5u/bG9p.6uKlrZCo0LwsT...
*	HULL	HULL	HULL	HULL

**Request 2: Login**

**Body (JSON):**

```

1 {
2   "librarianName": "Boss",
3   "librarianEmail": "boss@lib.com",
4   "librarianPass": "123"
5 }
```

**Response 2: 200 OK**

**Body (JSON):**

```

1 {
2   "message": "Login success",
3   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwicm9sZSI6ImxpYnJhcmlhbiiIsImlhCI6MTc2NDI1NTExNSwiZXhwIjoxNzY0MjgzOTcifQ.8q0078UMyeysXwotsZ290s03Tj6XRMygtuhBY-HwMY"
4 }
```

- Add Book but we need authorization token from librarian

HTTP <http://localhost:5000/books>

POST <http://localhost:5000/books>

Docs Params Authorization **Headers (9)** Body Scripts Settings

Auth Type  
Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.

HTTP <http://localhost:5000/books>

POST <http://localhost:5000/books>

Docs Params Auth **Headers (9)** Body **Body** Scripts Settings

raw JSON

```

1   {
2     "bookISBN": "000-001",
3     "bookTitle": "Database Introduction",
4     "bookGenre": "Education"
5   }

```

Body 200 OK

{ } JSON Preview Visualize

```

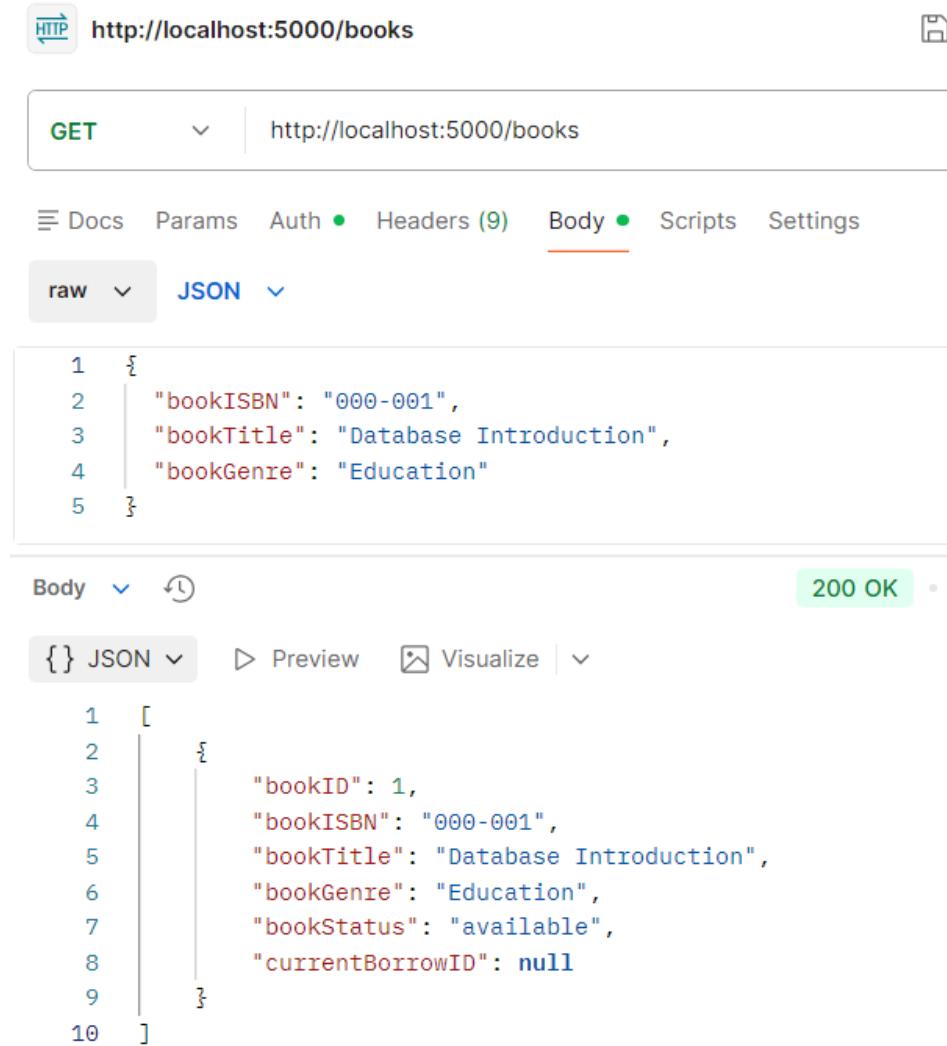
1   {
2     "message": "Book added"
3   }

```

	bookID	bookISBN	bookTitle	bookGenre	bookStatus
▶	1	000-001	Database Introduction	Education	available
*	NULL	NULL	NULL	NULL	NULL

## 2. READ

### - Get Books



The screenshot shows a POSTMAN interface with the following details:

- HTTP Method:** GET
- URL:** http://localhost:5000/books
- Headers:** (9 items)
- Body:** (Empty)
- Response Status:** 200 OK
- Response Body (JSON):**

```
1 {  
2   "bookISBN": "000-001",  
3   "bookTitle": "Database Introduction",  
4   "bookGenre": "Education"  
5 }
```

```
1 [  
2   {  
3     "bookID": 1,  
4     "bookISBN": "000-001",  
5     "bookTitle": "Database Introduction",  
6     "bookGenre": "Education",  
7     "bookStatus": "available",  
8     "currentBorrowID": null  
9   }  
10 ]
```

### 3. UPDATE

- Process of borrowing a book

HTTP <http://localhost:5000/borrow>

POST <http://localhost:5000/borrow>

Docs Params Auth Headers (9) Body Scripts Settings

raw JSON

```
1 {  
2   "bookID": 1  
3 }
```

Body [↻](#) 200 OK

{ } JSON ▾ ▶ Preview [Visualize](#) ▾

```
1 {  
2   "message": "Request sent"  
3 }
```

POST <http://localhost:5000/borrower/login> Send

Docs Params Authorization Headers (8) Body Scripts Settings Cookies

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL [JSON](#) [Beautify](#)

```
1 {  
2   "borrowerName": "Test User",  
3   "borrowerEmail": "test@example.com",  
4   "borrowerPass": "password123"  
5 }
```

Body Cookies Headers (8) Test Results | 10:10 PM 200 OK × 1.09 s × 467 B ⏺ ⏹ ⏷ ⏸ ⏵ ⏴

{ } JSON ▾ ▶ Preview [Visualize](#) ▾

```
1 {  
2   "message": "Login success",  
3   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwicm9sZSI6ImJvcnJvd2ViIiwiaWF0IjoxNzY0MjU2MjU0LCJleHAiOjE3NjQyODUwNTR9.cBI-I8oSQQEwc5xEHmSPxD20jd_1b_aJQ_VEPfS8"  
4 }
```

POST <http://localhost:5000/borrow/approve>

☰ Docs Params Auth • Headers (9) Body • Scripts Settings

raw ▾ JSON ▾

```
1  {
2  |  "borrowID": 1,
3  |  "days": 7
4 }
```

Body ▾ ⏪

200 OK

{ } JSON ▾ ▶ Preview ⚒ Visualize ▾

```
1  {
2  |  "message": "Approved"
3 }
```

	bookID	bookISBN	bookTitle	bookGenre	bookStatus
▶	1	000-001	Database Introduction	Education	borrowed
*	NULL	NULL	NULL	NULL	NULL

## 4. DELETE

- Deleting a book and its history

The screenshot shows the Postman interface with the following details:

- HTTP Method:** DELETE
- URL:** <http://localhost:5000/books/1>
- Auth Type:** Bearer Token
- Token:** A redacted token value.
- Body:** JSON response (200 OK):

```
1 {  
2   "message": "Book and history deleted"  
3 }
```
- Table:** A table showing the state of the database after deletion:

	bookID	bookISBN	bookTitle	bookGenre	bookStatus
*	NULL	NULL	NULL	NULL	NULL