# **Library Management API Documentation**

This API allows for managing books, borrowers, borrowing, and returning books.

**Base URL**

http://localhost:8080/api

**1. Add a Borrower**

**Endpoint**

POST /borrowers

**Request Headers**

Content-Type: application/json

**Request Body**

{

"name": "John Doe",

"email": "john@example.com"

}

**Responses**

* **200 OK** - Returns the saved borrower.
* **400 Bad Request** - If a borrower with the same name or email already exists.

**Example cURL**

Under Windows:  
curl -X POST http://localhost:8080/api/borrowers -H "Content-Type: application/json" -d "{\"name\": \"John Doe\", \"email\": \"john@example.com\"}"  
  
Under Linux:

curl -X POST http://localhost:8080/api/borrowers -H "Content-Type: application/json" -d '{"name": "John Doe", "email": "john@example.com"}'  
  
return:  
{"id":1,"name":"John Doe","email":"john@example.com"}

**2. Add a Book**

**Endpoint**

POST /books

**Request Headers**

Content-Type: application/json

**Request Body**

{

"isbn": "1234567890",

"title": "Spring Boot Guide",

"author": "Jane Smith",

"borrow": 0

}

**Responses**

* **200 OK** - Returns the saved book.
* **400 Bad Request** - If a book with the same ISBN but a different title or author already exists.

**Example cURL**

under Windows:

curl -X POST http://localhost:8080/api/books -H "Content-Type: application/json" -d "{\"isbn\": \"1234567890\", \"title\": \"Spring Boot Guide\", \"author\": \"Jane Smith\", \"borrow\": 0}"

under Linux:

curl -X POST http://localhost:8080/api/books -H "Content-Type: application/json" -d '{"isbn": "1234567890", "title": "Spring Boot Guide", "author": "Jane Smith", "borrow": 0}'

Return:  
{"id":2,"isbn":"1234567890","title":"Spring Boot Guide","author":"Jane Smith","borrow":0}

**3. List All Books**

**Endpoint**

GET /books

**Responses**

* **200 OK** - Returns a list of all books.

**Example cURL**under Windows / Linux:

curl -X GET http://localhost:8080/api/books

return:  
"Spring Boot Guide","author":"Jane Smith","borrow":0}]

**4. Borrow a Book**

**Endpoint**

POST /borrow/{bookId}?borrowerId={borrowerId}

**Path Parameter**

* bookId (required): ID of the book to borrow.

**Query Parameter**

* borrowerId (required): ID of the borrower.

**Responses**

* **200 OK** - If the book was borrowed successfully.
* **400 Bad Request** - If the book or borrower doesn't exist, or if the book is already borrowed.

**Example cURL**

under windows:  
curl -X POST "http://localhost:8080/api/borrow/1?bookId=1&borrowerId=1"

under Linux:  
curl -X POST http://localhost:8080/api/borrow/1?bookId=1&borrowerId=1

return:  
Book borrowed successfully.or  
Book is already borrowed.

**5. Return a Book**

**Endpoint**

POST /return/{bookId}

**Path Parameter**

* bookId (required): ID of the book to return.

**Responses**

* **200 OK** - If the book was returned successfully.
* **400 Bad Request** - If the book doesn't exist, isn't borrowed, or if the borrowing record is missing.

**Example cURL**

under windows:

curl -X POST "http://localhost:8080/api/return/1?bookId=1"

under Linux:

curl -X POST http://localhost:8080/api/return/1?bookId=1

**Error Responses**

* **400 Bad Request** - Returns error messages for validation failures or unknown exceptions.

**Transactional Behavior**

* The **/borrow/{bookId}** and **/return/{bookId}** endpoints use transactions to ensure data consistency.
* The book record is **locked for updates** during borrowing and returning operations to avoid concurrent conflicts.

**Notes**

1. The borrow field in the book entity:

* 0 means **available**.
* 1 means **borrowed**.

1. The Borrow entity keeps track of borrowTime and returnTime for historical records.
2. Validation ensures:

* No duplicate borrower names or emails.
* Book ISBNs can repeat, but the combination of title and author must match for duplicate ISBNs.

This documentation should provide a clear guide for interacting with the API.

the test result





