

Library Management System

Author: Elbadry Mohamed Elsayed Elbadry







Email: elbadry.mohamed2025@outlook.com

How to run the application:

1. Go to the directory where the JAR file is placed. **Download (JAR) (could not be uploaded to GitHub due to size limitations)**
2. Open the CMD then write the following command “java -jar library.jar”.
3. Instead, you can just double click on the JAR file and the backend server will start.
4. You can open the project through IntelliJ then run the project.
5. The server is running on port 8081.

Notes:

- A. H2 database is used.
- B. The ID of both books and patrons are incremental so if you tried to set an id of one of them that would not take effect.
- C. A book or a patron could not be deleted if there is a borrowing record, because this will violate the referential integrity.
- D. Post Man could be used to test the API endpoints.
- E. Test Coverage = 62% (Line Coverage)

Element ^	Class, %	Method, %	Line, %
✓  LibraryMS.MS	78% (11/14)	59% (49/82)	62% (67/108)
>  controller	66% (4/6)	47% (18/38)	45% (23/51)
>  model	100% (4/4)	53% (14/26)	53% (14/26)
>  repository	100% (0/0)	100% (0/0)	100% (0/0)
>  service	100% (3/3)	100% (17/17)	100% (30/30)
 MsApplication	0% (0/1)	0% (0/1)	0% (0/1)

How to use the API endpoints:

- 1) **GET /api/books**
 - Description: Retrieves a list of all books in the library.
 - Example Usage:

GET /api/books

Response:

```
[
  {
    "id": 1,
    "author": "maids",
    "isbn": "1234567215",
    "title": "new Book",
    "publication_Year": "2020-01-01"
  },
  {
    "id": 2,
    "author": "maids99",
    "isbn": "1234567212122511",
    "title": "new Book99",
    "publication_Year": "2020-01-01"
  }
]
```

2) **GET /api/books/{id}**

- Description: Retrieves details of a specific book by its ID.
- Example Usage:

GET /api/books/1

Response:

```
{
  "id": 1,
  "author": "maids",
  "isbn": "1234567215",
  "title": "new Book",
  "publication_Year": "2020-01-01"
}
```

3) **POST /api/books**

- Description: Adds a new book to the library.
- Example Usage:

POST /api/books

Request Body:

```
{
  "title": "Book1",
  "author": "maids",
  "publication_Year": "2020-01-01",
  "isbn": "1234567215"
}
```

4) PUT /api/books

- Description: Updates an existing book in the library.
- Example Usage:

```
PUT /api/books
```

Request Body:

```
{
  "id": 1,
  "title": "new Book",
  "author": "maids",
  "publication_Year": "2020-01-01",
  "isbn": "1234567215"
}
```

5) DELETE /api/books/{id}

- Description: Deletes a book from the library by its ID.
- Example Usage:

```
DELETE /api/books/1
```

6) GET /api/patrons

- Description: Retrieves a list of all patrons in the library system.
- Example usage

```
GET /api/patrons
```

Response:

```
[
  {
    "id": 1,
    "name": "John Doe",
    "email": "john@example.com",
    "phone": "123-456-7890"
  },
  {
    "id": 2,
    "name": "Jane Smith",
    "email": "jane@example.com",
    "phone": "987-654-3210"
  },
  ...
]
```

7) **GET /api/patrons/{id}**

- Description: Retrieves details of a specific patron identified by their ID.
- Example usage

```
GET /api/patrons/1
```

Response:

```
{
  "id": 1,
  "name": "John Doe",
  "email": "john@example.com",
  "phone": "123-456-7890"
}
```

8) **POST /api/patrons**

- Description: Adds a new patron to the library system.
- Example Usage:

```
POST /api/patrons
```

Request Body:

```
{
  "name": "Elbadry",
  "email": "elbadry.mohamed2025@outlook.com",
  "phone": "01006638953",
  "password": "1234"
}
```

9) **PUT /api/patrons**

- Description: Updates details of an existing patron in the library system.
- Example Usage:

```
PUT /api/patrons
```

Request Body:

```
{
  "id": 1,
  "name": "John Doe",
  "email": "john.doe@example.com",
  "phone": "123-456-7890"
}
```

10) **DELETE /api/patrons/{id}**

- Description: Deletes a patron from the library system based on their ID.
- Example Usage:

```
DELETE /api/patrons/1
```

11) GET /api/borrow

- Description: Retrieves a list of all borrowing records in the library system.
- Example Usage:

```
GET /api/borrow
```

Response:

```
[
  {
    "bookID": 1,
    "patronID": 1,
    "borrow_date": "2024-04-12",
    "return_date": "2024-04-12"
  },
  {
    "bookID": 2,
    "patronID": 1,
    "borrow_date": "2024-04-12",
    "return_date": null
  }
]
```

12) POST /api/borrow/{bookId}/patron/{patronId}

- Description: Allows a patron to borrow a book from the library.
- Example Usage:

```
POST /api/borrow/1/patron/1
```

13) PUT /api/return/{bookId}/patron/{patronId}

- Description: Allows a patron to return a borrowed book to the library.
- Example Usage:

```
PUT /api/return/1/patron/1
```