Running the Application

To run the Java application, follow these steps:

1. Open the project in IntelliJ IDEA.
2. Make sure you have the Java SDK installed and configured in IntelliJ.
3. Click on the "Run" button or press `` `Shift + F10` `` to run the application.
4. The application will start and you can access it through the API endpoints.

Running the Application with H2 Database

1. To see H2 database, browse http://localhost:8080/h2-console

2. JDBC URL is `jdbc:h2:mem:library`, username is `sa` and password is empty , then “Connect” button

3. for a reference run below query:

SELECT * FROM BOOK ;

SELECT * FROM USERLIST;

SELECT * FROM ORDERLIST;

Access the API endpoints using a tool like Postman or cURL.

1. Order a Book

Method: `POST`

Endpoint: `/orders/{user_id}/{book_id}`

Parameters:

- `user_id`: The ID of the user placing the order
- `book_id`: The ID of the book being ordered

Response:

- `` `201 Created` `` if the order is successful
- `` `422 Unprocessable Entity` `` if the request is invalid (e.g., user or book not found, book not available)

Order a Book SuccessFully

Method: `POST`

Endpoint: `/orders/1/1`

Request Body: None

Response:

{

"body": "success",

"statusCode": 201

}

b . Order a Book

Method: `POST`

Endpoint: `/orders/4/5`

Request Body: None

Response:

{

"body": "success",

"statusCode": 201

}

Order a Book Failure:

a. Invalid user with age below 18

Method: `POST`

Endpoint: `/orders/4/3`

Request Body: None

Response:

{

"body": "Invalid Request: User is below 18",

"statusCode": 422

}

b. User Borrow MAGAZINE with SILVER MEMBERSHIP

Method: `POST`

Endpoint: `/orders/4/3`

Request Body: None

Response:


```
{
  "body": "Invalid Request: User cannot borrow book",
  "statusCode": 422
}
```


c. User Borrow 4th BOOK with GOLD MEMBERSHIP

Method: `POST`

Endpoint: `/orders/2/4`

Request Body: None

Response:


```
{
  "body": "Invalid Request: User cannot borrow book",
  "statusCode": 422
}
```


c. When Book is not available

Method: `POST`

Endpoint: `/orders/4/5`

Request Body: None

Response:


```
{
  "body": "Invalid Request: Book is not available",
  "statusCode": 422
```

}

d. When Transaction are 10 in a month

Method: `POST`

Endpoint: `/orders/3/3`

Request Body: None

Response:

{

"body": "Invalid Request: Transaction limit is ended",

"statusCode": 422

}

2. Return a Book (Single Order)

Method: `PUT`

Endpoint: `/return/{orderId}`

Parameters:

- `orderId`: The ID of the order being returned

Response:

- `` `200 OK` `` if the book is returned successfully
- `` `404 Not Found` `` if the order is not found

a. Return a Book SuccessFully(Single Order)

Method: `PUT`

Endpoint: `/return/1`

Request Body: None

Response:

{

"body": "Book Returned Successfully",

"statusCode": 200

}

b. Return a Book Failure(Single Order)

Method: `PUT`

Endpoint: `/return/5`

Request Body: None

Response:

{

"body": "Invalid Request: User Not Found",

"statusCode": 404

}

3. Return Multiple Books

Method: `PUT`

Endpoint: `/return`

Request Body:

- A list of order IDs to be returned

Response:

- `` `200 OK` `` if all books are returned successfully
- `` `404 Not Found` `` if any of the orders are not found

a. Return Multiple Books SuccessFully

Request:`PUT /return`

Request Body:

```
[
  1,
  2,
  3
]
```

Response:

```
{
  "body": "Book Returned Successfully",
  "statusCode": 200
}
```

b. Return Multiple Books Failure

Request:`PUT /return`

Request Body:

```
[
  1,
  2,
 4,
  3
]
```

Response:

```
{
  "body": "Invalid Request: User Not Found",
  "statusCode": 404
}
```

Error Handling

The controller catches and logs any exceptions that occur during the execution of the endpoints. It returns a `ResponseEntity` with an error message and a corresponding HTTP status code (422 or 404) in case of an exception.