Contents

[Library management system 1](#_Toc172882603)

[The base project 2](#_Toc172882604)

[Project tools 2](#_Toc172882605)

[Project Structure 2](#_Toc172882606)

[Building the project 3](#_Toc172882607)

[Running the application. 4](#_Toc172882608)

[Testing the first endpoint 5](#_Toc172882609)

[Application requirements 6](#_Toc172882610)

[ERD Diagram 6](#_Toc172882611)

[User stories 6](#_Toc172882612)

[Register customer 6](#_Toc172882613)

[Edit customer details 7](#_Toc172882614)

[Search customer by ID number 7](#_Toc172882615)

[Search customer by email address 8](#_Toc172882616)

[Register a book publisher 8](#_Toc172882617)

[Search a book publisher by ISNI 9](#_Toc172882618)

[Add book to library 9](#_Toc172882619)

[Search books by ISBN 10](#_Toc172882620)

[Lend a book to customer 10](#_Toc172882621)

[View all books issued to customer. 11](#_Toc172882622)

[Send threating email notifications to customers with books overdue. 11](#_Toc172882623)

[Capture book return by customer. 12](#_Toc172882624)

[What to expect on the next interactions 12](#_Toc172882625)

# Library management system

This project is all about building a library management system using **spring boot**. This project will touch almost all-important features of spring boot. There is going a lot of code to write and lots of learning. After completing this project, you be well on your way mastering spring boot.

# The base project

We are going of a base project that has already been created with plenty of missing features. We walk through base project by building it from scratch.

## Project tools

* JDK 11
* Maven
* **NB The base project uses H2 database. H2 is java in memory database which is already include in this project works like any other SQL database. In the later stage going replace it with MySQL or PostgreSQL.**
* Any IDE
* Internet time for research.
* Postman preferred or culr.

## Project Structure

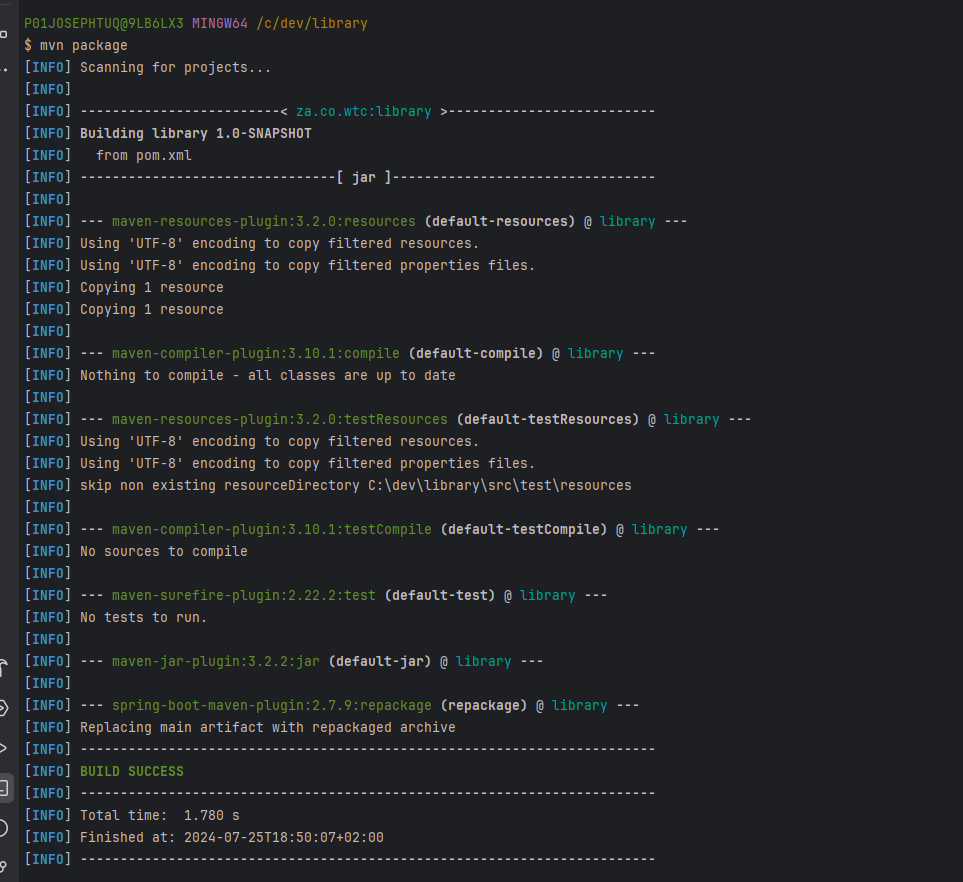
A screenshot of a computer

Description automatically generated

## Building the project

**NOTE you will need to be connected to internet to build the project for the first time.**

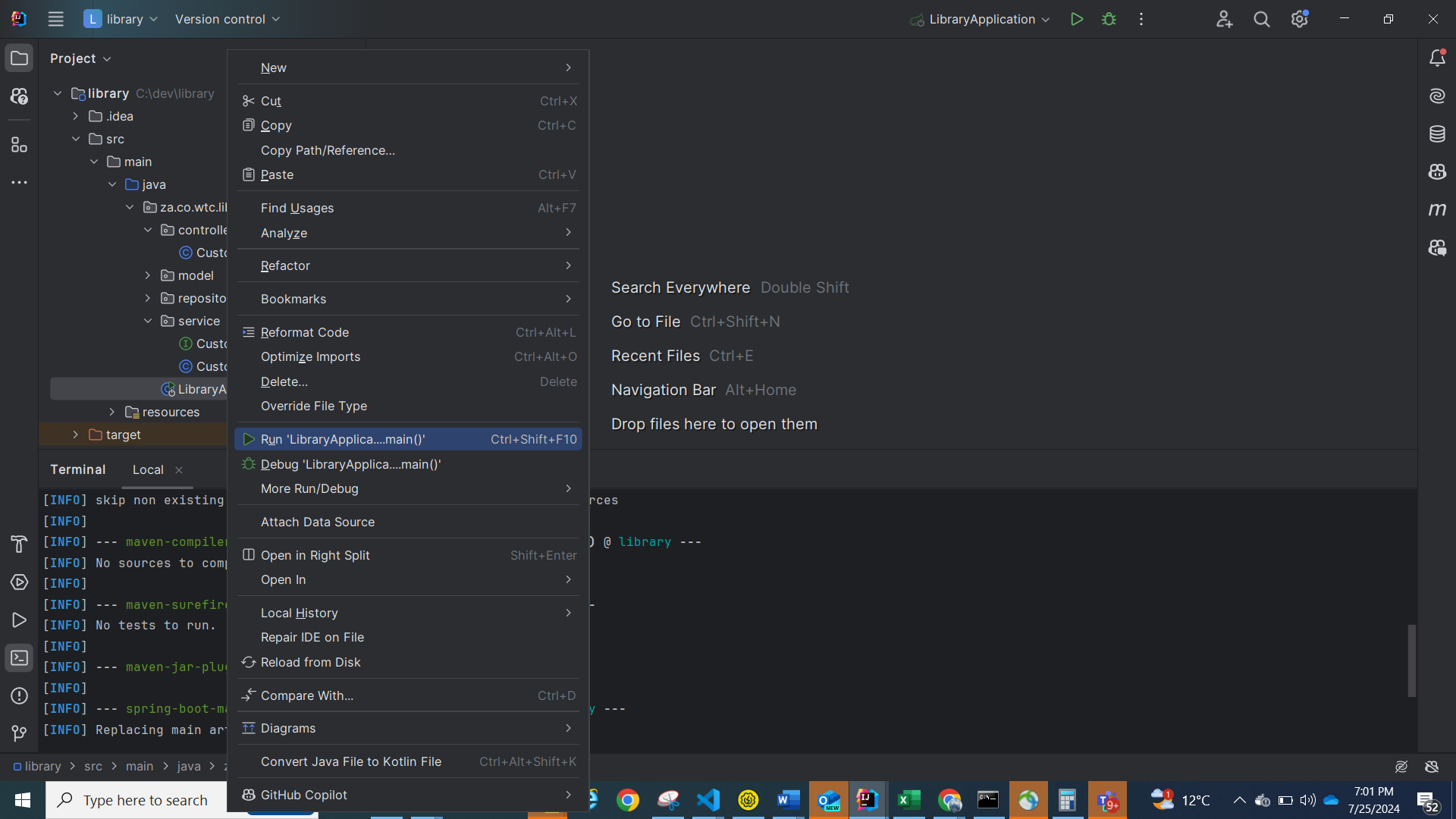
1. Unzip the project and place in your preferred directory
2. Open the project directory from IntelliJ.
3. Open new terminal in IntelliJ.
4. In the terminal type mvn package



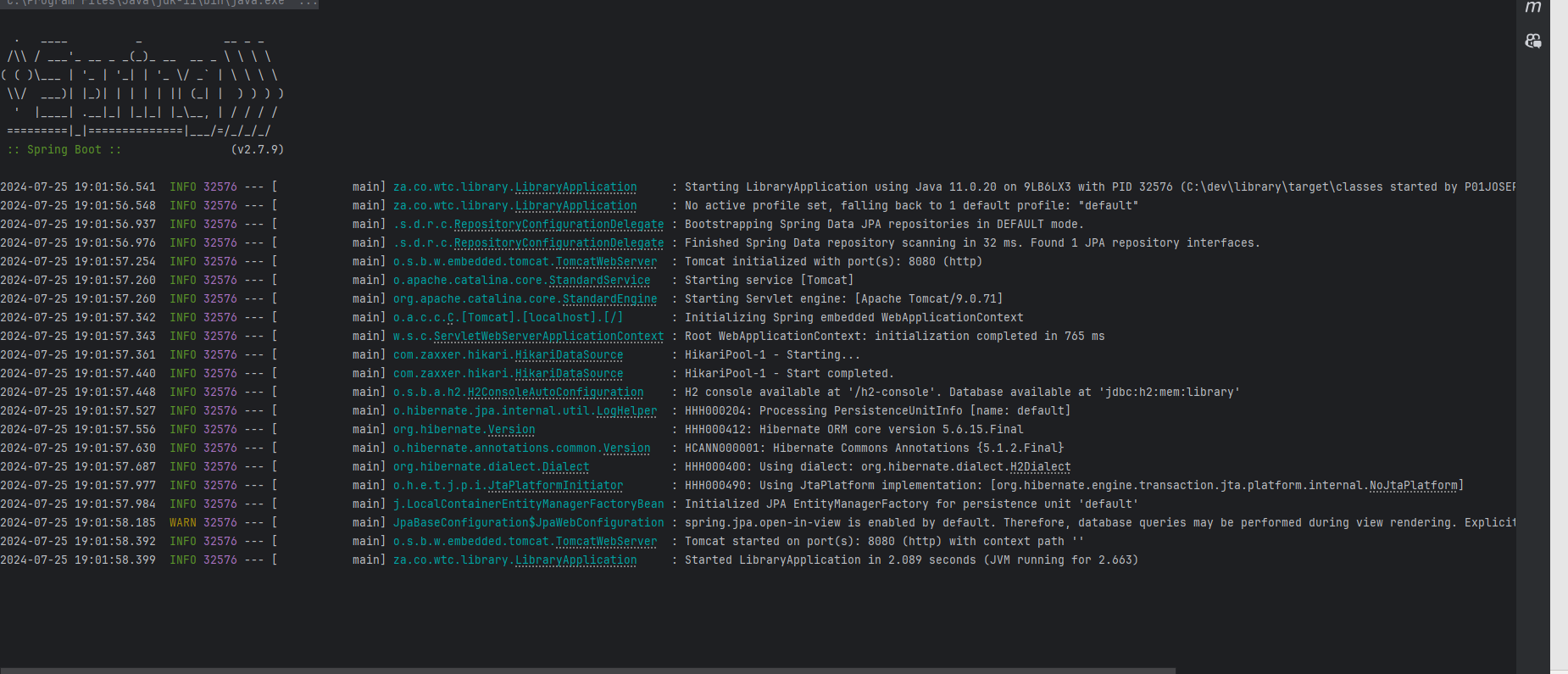
**If the project does not build do not worry, we will fix it during the session.**

## Running the application.

Right click on LibraryApplication class and choose to run.

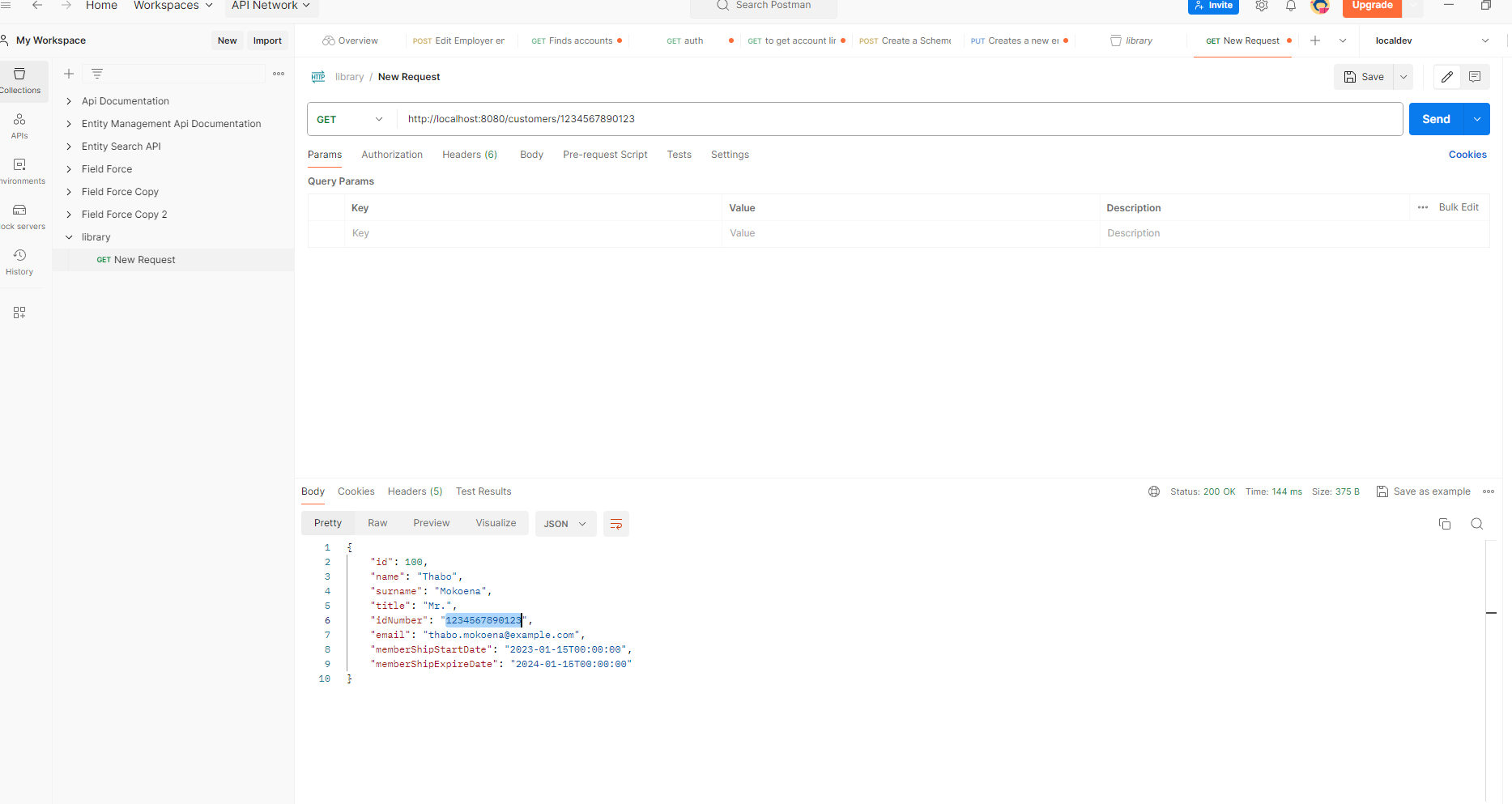


**If everything goes well, you should see.**



## Testing the first endpoint

At this you are ready to test the first HTTP endpoint.



**At this point you feel excited the easy part is done, now we are getting coding business.**

# Application requirements

The WTC library has been experiencing theft of books by students and staff. This issue arises from the lack of a library management system, allowing anyone to take a book without returning it. To address the issue of book theft at the WTC library, we recommend implementing a comprehensive library management system. This system will track the borrowing and returning of books, ensuring accountability and reducing the likelihood of theft by both students and staff.

## ERD Diagram

A diagram of a computer

Description automatically generated with medium confidence

## User stories

### Register customer

|  |  |  |
| --- | --- | --- |
| No | User story | Acceptance Criteria |
| Register new customer | As a stuff member, I want to be able to register a new customer | Given that the staff member captured all customer details, then the customer should be registered |

#### Conditions:

* A customer with same id number should not exist.
* A customer should provide a valid email address.
* A customer with same email address should not exist.
* It’s the responsibility of the application to ensure that expiry date of membership is 1 year from the date of registration.

#### Endpoint:

POST http://localhost:8080/customers

### Edit customer details

|  |  |  |
| --- | --- | --- |
| No | User story | Acceptance Criteria |
| Edit customer details | As a stuff member, I want to be able edit customer details. | Given that the customer details have changed, as a stuff member I want to edit the customer details. |

#### Conditions:

* A customer must already be registered.

#### Endpoint:

PUT http://localhost:8080/customers

### Search customer by ID number

|  |  |  |
| --- | --- | --- |
| No | User story | Acceptance Criteria |
| Search customer by their ID number | As a stuff member, I want to be able search customer by their ID number. | Given that a valid customer ID number was entered, I expect to see the customer details. |

#### Conditions:

* A registered customer ID number must have been entered.

#### Endpoint:

GET http://localhost:8080/customers/id-number/{customer-id-number-goes-here}

### Search customer by email address

|  |  |  |
| --- | --- | --- |
| No | User story | Acceptance Criteria |
| Search customer by their ID number | As a stuff member, I want to be able search customer by their ID number. | Given that a valid customer email address was entered, I expect to see the customer details. |

#### Conditions:

* The email address must be in a valid format.

#### Endpoint:

GET http://localhost:8080/customers/email/{customer-email-goes-here}

### Register a book publisher

|  |  |  |
| --- | --- | --- |
| No | User story | Acceptance Criteria |
| Register a book publisher | As a stuff member, I want to be able create a publisher | Given that I have captured the details of the book publisher. I expect to see the details of the publisher. |

#### Conditions:

* The publisher should not already exist by ISNI.
* The ISNI should be 16 digits long.

#### Endpoint:

POST http://localhost:8080/publishers

### Search a book publisher by ISNI

|  |  |  |
| --- | --- | --- |
| No | User story | Acceptance Criteria |
| Search a publisher by their ISNI. | As a stuff member, I want to be able search publisher by their ISNI. | Given that I have entered a valid ISNI that exist. I expect to see the details of the publisher. |

#### Conditions:

* A 16 Digit ISNI must be entered.

#### Endpoint:

GET http://localhost:8080/publishers/isni/{isni-goes-here}

### Add book to library

|  |  |  |
| --- | --- | --- |
| No | User story | Acceptance Criteria |
| Add new book to library | As a stuff member, I want to be able add a new book to the library. | Given that I have captured the book details, I want the book to added to the library. |

#### Conditions:

* The book title must be captured can not be null or empty
* A 13-digit ISBN number is captured.
* We should be able to add multiple books with same title and ISBN.
* A book category should be captured.
* The publisher of the book must exist in the library.

#### Endpoint:

POST http://localhost:8080/books

### Search books by ISBN

|  |  |  |
| --- | --- | --- |
| No | User story | Acceptance Criteria |
| Search books by their ISBN. | As a stuff member, I want to be able search a book by the ISBN. | Given that I have entered a valid ISBN that exist. I expect to see the details of the book. |

#### Conditions:

* A 16 Digit ISBN must be entered.

#### Results expected

* If a book with a given ISBN exists in the library the book details should returned.
* If a book with a given ISBN does not exists nothing should be returned.

#### Endpoint:

GET http://localhost:8080/books/isbn/{isbn-goes-here}

### Lend a book to customer

|  |  |  |
| --- | --- | --- |
| No | User story | Acceptance Criteria |
| Lend a book to customer. | As a stuff member, I want to be  lend a book to a customer | Given that I have entered a valid ISBN that exist. I want to be a to lend the book to the customer. |

#### Conditions:

* A 16 Digit ISBN must be entered.
* The book must exit in the library.
* The customer must be registered and have an active membership.
* A customer cannot have more than 2 books issued to them.
* There must be at least one copy of the book in the library that is currently not with other customer. Example If there are 2 copies of the book and one is currently issued to other customer, then the second should be issued the second copy of the book.

#### Results expected

* If a book was successfully issued to the customer, then this should reflect in the database.
* A book issued to the customer, should be overdue a month after the date of issue.
* If a book with a given ISBN does not exists nothing should be returned.

#### Endpoint:

POST http://localhost:8080/issuesing

### View all books issued to customer.

|  |  |  |
| --- | --- | --- |
| No | User story | Acceptance Criteria |
| View all books issued to customer. | As a stuff member, I want to be  view books currently issued to a customer. | Given that I have entered a valid customer ID number that exist. I want to be able view all books currently issued to the customer |

#### Conditions:

* A customer must exit in the library.

#### Results expected

* If there are books currently issued to the customer, The list of books must be listed.

#### Endpoint:

GET [http://localhost:8080/issuesing/customer/customer-id/{customer-id-goes-here}](http://localhost:8080/issuesing/customer/customer-id/%7bcustomer-id-goes-here%7d)

### Send threating email notifications to customers with books overdue.

|  |  |  |
| --- | --- | --- |
| No | User story | Acceptance Criteria |
| Customer notification. | As a stuff member, I want to be  Send email to all customers that have books that already overdue. | Given that I have entered a valid customer ID number that exist. I want to be able view all books currently issued to the customer |

#### Conditions:

* Email should be sent to customers with books overdue.

#### Results expected

* If a customer a has book overdue they should a threating email.

#### Endpoint:

POST http://localhost:8080/issuesing/notifications/overdue

### Capture book return by customer.

|  |  |  |
| --- | --- | --- |
| No | User story | Acceptance Criteria |
| Capture book return | As a stuff member, I want to be  able captured a book retuned by a customer. | Given that I have entered a valid customer ID number that exist and ISBN of book. I want to capture a book return and have that reflect in the database. |

#### Conditions:

* The customer must be registered
* The book with a given ISBN must exit in the library.
* The book should currently be issued the given customer.

#### Results expected

* If all conditions are met, then book must reflect as returned to the library.

#### Endpoint:

POST http://localhost:8080/issuesing/returns/{isbn-goes-here}/{customer-id-goes-here}

This was just the first interactions.

## What to expect on the next interactions

* Currently our library does not have any build in security in terms of staff log in.
* Currently our library does not verify customer when they register. Example upload customer documents such as copy of ID and proof of residence.
* Rating of books.
* The CEO does have any visibility about the running of the library. The CEO wants a report dashboard about the library stats.
* The library system is only back-end where the UI is.
* Spring boot has tone of features and the aim is cover all most important features of spring boot.