LIBRARY MANAGEMENT SYSTEM

**Revision History**

| **Version** | **Date** | **Author** | **Description** |
| --- | --- | --- | --- |
| 0.10 | 17/11/23 | Josh L. | Document Structure added |
| 0.20 | 18/11/23 | Josh L. | Rewrite of Business Requirement Document |
| 0.30 | 19/11/23 | Josh L. | UML Diagrams: EER added |
| 0.40 | 20/11/23 | Josh L. | Normalization on Database |
| 0.50 | 21/11/23 | Josh L. | Business Rules Formulated |
| 0.60 | 21/11/23 | Josh L. | Data Correction for Database |
| 0.70 | 22/11/23 | Josh L. | Added DB Tables to document |
| 0.80 | 22/11/23 | Josh L. | Database Design Completion |
| 0.90 | 23/11/23 | Josh L. | JDBC Connector Established |
| 1.00 | 24/11/23 | Josh L. | Removed User Manual |
| 1.05 | 25/11/23 | Josh L. | Added Class Diagram |
| 1.10 | 15/12/23 | Josh L. | Added Sequence Diagram |
| 1.15 | 16/12/23 | Josh L. | System Requirement Specs |
| 1.20 | 17/12/23 | Josh L. | Project Structure Writeup |
| 1.25 | 18/12/23 | Josh L. | Added Book Entering UI |
| 1.30 | 19/12/23 | Josh L. | Added UAT Module |
| 1.35 | 20/12/23 | Josh L. |  |

This is a structured outline for my Library Management System Documentation.

Contents

[BUSINESS REQUIREMENTS DOCUMENT 3](#_Toc153975111)

[PURPOSE 3](#_Toc153975112)

[OBJECTIVES 3](#_Toc153975113)

[SCOPE 3](#_Toc153975114)

[RULES 4](#_Toc153975115)

[ASSUMPTIONS 4](#_Toc153975116)

[CONSTRAINTS 4](#_Toc153975117)

[PROJECT DELIVERABLES 4](#_Toc153975118)

[BACKEND 4](#_Toc153975119)

[FRONTEND 5](#_Toc153975120)

[SYSTEM REQUIREMENTS SPECIFICATION 5](#_Toc153975121)

[SYSTEM ARCHITECTURE 5](#_Toc153975122)

[SYSTEMS DIAGAMS 5](#_Toc153975123)

[System Overview 7](#_Toc153975124)

[Technology 7](#_Toc153975125)

[Project Structure 8](#_Toc153975126)

[FUNCTIONAL REQUIREMENTS 9](#_Toc153975127)

[User Management 9](#_Toc153975128)

[Book Management 9](#_Toc153975129)

[Reporting and Analytics 9](#_Toc153975130)

[Borrowing and Returning 10](#_Toc153975131)

[NON-FUNCTIONAL REQUIREMENTS 11](#_Toc153975132)

[PERFORMANCE 11](#_Toc153975133)

[SECURITY 11](#_Toc153975134)

[USABILITY 11](#_Toc153975135)

[RELIABILITY 11](#_Toc153975136)

[SCALABILITY 11](#_Toc153975137)

[DESIGN DOCUMENTS 12](#_Toc153975138)

[DATABASE DESIGN 12](#_Toc153975139)

[INDEXES 12](#_Toc153975140)

[SCHEMA 12](#_Toc153975141)

[TABLES 12](#_Toc153975142)

[COLUMNS 13](#_Toc153975143)

[Entities and Attributes 15](#_Toc153975144)

[Relationships between Entities 16](#_Toc153975145)

[UML DIAGRAMS 17](#_Toc153975146)

[enhanced entity-relationship diagram 17](#_Toc153975147)

[USE CASE DIAGRAM 18](#_Toc153975148)

[CLASS DIAGRAM 20](#_Toc153975149)

[SEQUENCE DIAGRAM 24](#_Toc153975150)

[PROTOTYPING 24](#_Toc153975151)

[WIREFRAMES 25](#_Toc153975152)

[USER ACCEPTANCE TESTS 28](#_Toc153975153)

[TEST SUMMARY 29](#_Toc153975154)

[TEST CASES 29](#_Toc153975155)

[USER MANUAL 38](#_Toc153975156)

[FOR STAFF 38](#_Toc153975157)

[TO LOG IN 38](#_Toc153975158)

[LIST ALL BOOKS 40](#_Toc153975159)

[ENTER BOOK 42](#_Toc153975160)

[FOR MEMBERS 42](#_Toc153975161)

# BUSINESS REQUIREMENTS DOCUMENT

**This document outlines the business requirements, objectives, scope, and constraints of the project. It provides a high-level view of what the system is intended to accomplish.**

## PURPOSE

A new library housed inside a private residence is started. It currently has 2 staff is in process of stocking books. The library has WIFI facilities and is open to a resident size of 250. It is not open to public. As the facilities is still very new, less then 10% of residence have registered as library member. Members, books volume is expected to increase as more complete their renovation and start to move in to premise. More librarians will also be hired along the way.

The purpose of this Library Management System (LMS) is to streamline the processes involved in managing a library's resources, including books, periodicals, and user information.

## OBJECTIVES

* Improve efficiency in library operations.
* Enhance user experience for both staff and library patrons.
* Provide accurate and real-time information on library resources.

## SCOPE

The LMS will cover administrative functionalities to assist librarians in managing this library.

### RULES

1. Each member can borrow up to 8 books at once.
2. Each book borrowed is for 3 weeks.
3. Staff can only borrow book as regular member. And all library rules shall apply for staff.
4. This library uses standard ISBN13 convention for book numbering.
5. Library fines or late fees are imposed on books not returned or is returned late.
6. Library fines must be paid over the counter to a librarian before borrowing can commerce.
7. Booking ID is the session of booking. No members can share a common booking ID.
8. User ID is either the Member ID or Staff ID. If staff borrows in this library, standard borrowing rules (Including fines) shall apply.
9. The system log in is User ID and password. Currently only staff are required to log in as members can browse freely in guest mode.
10. Password used must be combination of uppercase letters, lowercase letters, numbers, and symbols. And contain at least 9 characters.
11. Each library Booking Id is 1 book. Member’s Booking Id field is updated to most recent borrowing activity of the particular member.

### ASSUMPTIONS

1. The library does not stock duplicate titles. Only 1 copy per title.
2. All staffers are library members and have library Member Id.
3. Library book availability status is maintained by Staffs upon receiving book return and keying the update into system.
4. If book status is ‘Available’, this means the book is ready to be loaned again.
5. If book status is ‘On Loan’, this means the book is currently unavailable.
6. Book index is physical location of book in library.

### CONSTRAINTS

* The system should be implemented within 1 month from November 2023 with required paperwork and source code submitted.

# PROJECT DELIVERABLES

## BACKEND

* + Spring Boot application with RESTful APIs
  + Database schema and data migration scripts
  + Unit tests for backend components

## FRONTEND

* + ReactJS application with user interface components
  + Routing and navigation between different pages
  + Integration with backend APIs for data fetching and manipulation

# SYSTEM REQUIREMENTS SPECIFICATION

**The SRS document provides detailed technical specifications, including functional and non-functional requirements. It describes the system's behavior, interfaces, data structures, and more.**

## SYSTEM ARCHITECTURE

**System Architecture Design: Describes the high-level architecture of the system, including how the frontend and backend components interact, and the database structure.**

### SYSTEMS DIAGAMS

* Spring Boot exports REST Apis using Spring Web MVC & interacts with Database using Spring JPA
* React Client sends HTTP Requests and retrieve HTTP Responses using axios, shows data on the components.
* We use React Router for navigating to pages.
* Database will either be MySQL or PostgreSQL.

#### Systems Interface

A diagram of a computer program

Description automatically generated

#### Security Layer

A diagram of security in action

Description automatically generated

`

#### Accessing Backend Database

A green arrow pointing to a square

Description automatically generated

#### Accessing Frontend Form

A diagram of a diagram

Description automatically generated

#### React with Java

A diagram of a software development process

Description automatically generated

## System Overview

The Library Management System or (LMS) in short will be a web-based application with a client-server architecture. It will include a user interface accessible through web browsers and a backend database to store and manage data.

We explore building a full stack Spring Boot + React.js + MySQL example with a CRUD App. The back-end server uses Spring Boot with Spring Web MVC for REST APIs and Spring Data JPA for interacting with MySQL/PostgreSQL database. Front-end side is made with React, React Router, Axios & Bootstrap.

### Technology

* Java 17 / 11 / 8
* Spring Boot 3 / 2 (with Spring Web MVC, Spring Data JPA)
* MySQL
* Maven

### Project Structure

A screenshot of a computer program

Description automatically generated

1. Book data model class corresponds to entity and table tutorials.
2. BookRepository is an interface that extends JpaRepository for CRUD methods and custom finder methods. It will be autowired in BookRestController.
3. BookRestController is a RestController which has request mapping methods for RESTful requests such as: getAllBooks, createBook, updateBook, deleteBook, findByBook…
4. Configuration for Spring Datasource, JPA & Hibernate in application.properties.
5. pom.xml contains dependencies for Spring Boot and MySQL.

#### Integration

JDBC CONNECTION

A screenshot of a computer

Description automatically generated

LMS will integrate with external systems for ISBN databases for book information.

# FUNCTIONAL REQUIREMENTS

Staffs making use of the system will be able to do the following:

## User Management

* Ability to add, edit, and delete library staff
* Manage user roles and permissions

## Book Management

* Add, edit, and delete books with details such as title, author, ISBN, and genre.
* Perform book search
* Categorize books using genres and subjects
* Maintain book availability status

## Reporting and Analytics

* Generate reports on book availability, user activities, and overdue books.
* Generate reports on book usage and borrowing trends
* Track book availability status
* Track user borrowing history. Analyze user borrowing patterns
* Track overdue books and generate notifications
* Export reports in different formats (e.g., PDF, CSV).

#### Book Reports

#### CSV exports

#### Analysing borrowing patterns by borrowing history

1. Member Kai is likely either Christian or a highly religious person from his borrowing activities he loaned ‘An Introduction to the Old Testament ‘and later ‘The Epic of Eden’. Both of similar book genre.
2. Member White has loaned out “Harry Potter and the Cursed Child”, “Fantastic Beasts: The Secrets of Dumbledore” Indicating a demand for children’s books. The library could do well to stock more of such genre as more residents start to move in and become a library member.
3. Member White’s library activities suggest he could be child age or have children at home who reads R K Rowlings’ books.
4. Member Dusan is quite a thriller fan. He borrowed all Dan Brown’s books and  
   is likely to return for next Dan Brown’s thriller book once done with current reading.
5. Only member Drake has outstanding uncleared library fine.
6. The missing book from library is with Ivan Toney, book titled “Man's Search for Meaning”. Borrowed since 5th Feb 2023. Book status is still ‘On Loan’ and Toney has since incurred a $11 fine.
7. Member Ivan Toney cannot borrow anymore books because of outstanding library fines amounting $11.00. Hence Toney has no borrowing activities of late.

## Borrowing and Returning

* Allow users to search for available books
* Facilitate borrowing requests for available books
* Manage borrowing periods and due dates
* Enable book returns and update book availability

Staff can also generate reports to:

* Maintain book availability status.
* Track overdue books and generate notifications
* Track user borrowing history.

3.1 User Management

3.2 Catalog Management

* Track book availability, status, and location.

3.3 Circulation

* Check-in and check-out of books.
* Reservation and hold functionalities.
* Overdue book notifications.

3.4 Reporting

* Generate reports on book availability, user activities, and overdue books.

3.5 Search and Discovery

* Integration with external databases for additional resource information.

# NON-FUNCTIONAL REQUIREMENTS

The library system will have these facilities for members:

## PERFORMANCE

* Book search operations should be efficient and responsive.
* Response time for common operations should be within acceptable limits.

## SECURITY

* Implement secure user authentication and authorization mechanisms.
* Authentication mechanisms or basic form of form verification for controlled access
* Protect sensitive data, such as user information and borrowing records.
* Prevent unauthorized access to library resources.

## USABILITY

* Consistent intuitive user interface for both library staff and patrons for easy navigation
* Provide clear instructions and guidance for users.
* Implement consistent design patterns across the frontend and backend.
* User training and support documentation

## RELIABILITY

* The system should be available 24/7 with minimal downtime for maintenance.

## SCALABILITY

* The system should be scalable to accommodate a growing number of books and users.

# DESIGN DOCUMENTS

* Java is very handy with number of data bases you can use their jars to add connections and you can perform crud operation in Java.
* Here we use Java as a backend, then spring boot for creating a microservices which act as a restful API for frontend in react JS.

## **DATABASE DESIGN**

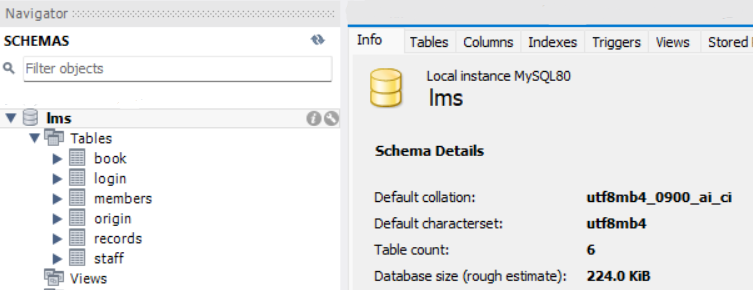
* + **Database Design: Contains details about the database schema, including tables, relationships, and data constraints.**

### INDEXES

**A screenshot of a computer

Description automatically generated**

### **SCHEMA**

****

### **TABLES**

**A screenshot of a computer

Description automatically generated**

### **COLUMNS**

**A screenshot of a computer

Description automatically generated  
**

#### Staff

**A screenshot of a computer

Description automatically generated**

#### **Members**

A screenshot of a computer

Description automatically generated

#### **Login**

A screenshot of a computer

Description automatically generated

#### Records

A screenshot of a computer

Description automatically generated

#### Book

A screenshot of a computer

Description automatically generated

#### Origin

**A screenshot of a computer

Description automatically generated**

### **Entities and Attributes**

**Staff Entity: It has Staff\_Joindate, Password, Staff\_Id, Staff\_Name, Staff\_Position. Staff\_Id is Primary Key.**

**Login Entity: It has Staff\_Id, Staff\_Email, Member\_Mobile, Member\_Email, Member\_Id, Booking\_Id  
Login\_Id is the Primary Key for Login Entity.  
Booking\_Id, Staff\_Id, Member\_Id are Foreign Keys for Login Entity.**

**Book Entity: It has Pages, Title, Category, Author.   
Title is the Primary Key for Book Entity.  
Isbn is Foreign Key for Book Entity.**

**Members Entity: It has** **Member\_Id, Email, Member\_Name, Member\_Mobile, Fine Member\_Email. Member\_Id is the Primary Key for Members Entity.  
User\_Id is Foreign Key for Members Entity.**

**Origin Entity: It has Genre, Publisher, YearOfPublication.   
ISBN is Primary Key of for Origin Entity.  
Index\_No is Foreign Key for Origin Entity.**

**Records Entity: It has Book\_Status, Booking\_Date, Index, Booking\_Id   
Index is the Primary Key for Records Entity.  
Staff\_Id is 1 of the 4 Foreign Keys for Records Entity.  
Member\_Id is 1 of the 4 Foreign Keys for Records Entity.  
Title is 1 of the 4 Foreign Keys for Records Entity.   
Isbn is 1 of the 4 Foreign Keys for Records Entity.**

### Relationships between Entities

1. **Each employee can only do 1 login session to the main library system for purpose of checking book status and indexed placing in the library.**
2. **Members are free to use library system to first check book availability through book status, and make a booking reservation to borrow it. No log in required however each member can have just 1 active browsing session with library system. Starting multiple sessions by single member is disallowed.**
3. **The library only has 1 centralised system and the system will reflect multiple book records at once.**
4. **Every book have 1 publication source.**
5. **The system also reveal multiple sources so that members can search base on a favourited publisher or book source.**

## UML DIAGRAMS

enhanced entity-relationship diagram

**The Enhanced ER Diagram include foreign keys necessary to relate to other tables.** A diagram of a company

Description automatically generated

### USE CASE DIAGRAM

A screenshot of a computer

Description automatically generated

Code:

@startuml

actor Staff as s

actor Member as m

left to right direction

package login{

usecase "Login\_Id" as UC1

usecase "Staff\_Id"

usecase "Member\_Id"

usecase "Staff\_Mobile"

usecase "Staff\_Email"

usecase "Booking\_Id"

usecase "Member\_Email"

usecase "Member\_Mobile"

}

package staff{

usecase "Staff\_Id"

usecase "Staff\_Joindate"

usecase "Password"

usecase "Staff\_Position"

usecase "Staff\_Name"

}

package origin{

usecase "ISBN"

usecase "YearOfPublication"

usecase "Genre"

usecase "Publisher"

usecase "Index\_No"

}

package records{

usecase "ISBN" as UC10

usecase "Title" as UC2

usecase "Index\_No" as UC3

usecase "Booking\_ID" as UC4

usecase "Booking\_Date" as UC5

usecase "Booking\_Status" as UC6

usecase "Fine" as UC7

usecase "Login\_Id" as UC8

usecase "Member\_Id" as UC9

}

package book{

usecase "Category"

usecase "Title"

usecase "Author"

usecase "Pages"

s --> UC1

m --> UC6

@enduml

Source URL:

<https://www.plantuml.com/plantuml/duml/VPFFJW8n48VlVOfcxoJ-KTIBGK0DHf04FNWYixkpIqEtOzhPg26-koasfCistiZtJUOtJ6T7Zi5ossW11PELAuQgak2a2s2EJOxMaqPeh5WoIQlg3SjIMInOaH7Y0uejr2WrrShiX9Ijmm8SokpPa_Mip7o7jybfxFPPtiNmBy_JVkcSSgKn8UuRKBh3nqHRPUfqzr3V5uU4dyDVSZvWbnox3vz8cH8OEse9pdsIJTKloIc_mOHQGBEVup08MNMytDbglCXYy8vWNwfbcsjLGA_n8nfxD9glT1kq7Jep9Nwj5zI9jrYGBLq_FppiIInU5MiCvYpLEhZpv9DDWxr8sIam1d-PyYi6Rbse6COL3yfWu5V_dEXryXg3l8ak8oVQTjOn0SQQx7VMNqLCxbhUa3qwaHfT9eIJWy7j_ZDfmg-X6A4fsqR_0W00>

### CLASS DIAGRAM

A computer screen shot of a computer

Description automatically generated

Source Code:

@startuml

!define DARKGREEN

class Login{

int Login\_Id

int Booking\_Id

String Staff\_Id

String Member\_Id

String Staff\_Email

String Member\_Email

int Member\_Mobile

int Staff\_Mobile

}

class Staff{

String Staff\_Id

String Staff\_Name

String Staff\_Joindate

String Staff\_Position

String Password

}

class Members{

String Member\_Id

int Booking\_Id

String Member\_Name

int Fine

String User\_Id

}

class Origin{

String Genre

String YearOfPublication

String Publisher

float Index\_No

String Book\_Title

}

class Records{

int Booking\_Id

int Fine

String Staff\_Id

String Member\_Id

String Isbn

}

Class Book{

String Title

String Author

String Categor

String Isbn

}

Book <.. Origin

Records <.. Staff

Records <.. Members

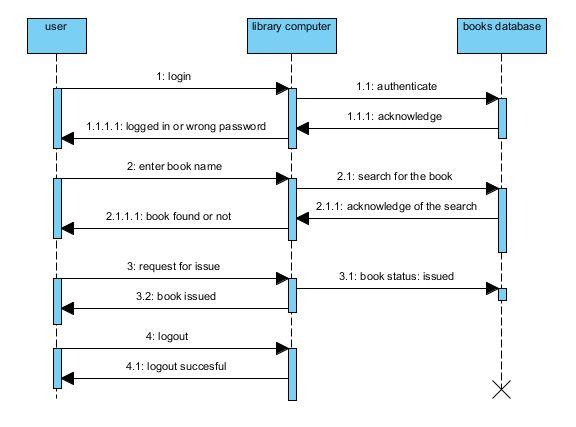
Members <.. Login

Staff <.. Login

@enduml

URL: <https://www.plantuml.com/plantuml/duml/VPFFJW8n48VlVOfcxoJ-KTIBGK0DHf04FNWYixkpIqEtOzhPg26-koasfCistiZtJUOtJ6T7Zi5ossW11PELAuQgak2a2s2EJOxMaqPeh5WoIQlg3SjIMInOaH7Y0uejr2WrrShiX9Ijmm8SokpPa_Mip7o7jybfxFPPtiNmBy_JVkcSSgKn8UuRKBh3nqHRPUfqzr3V5uU4dyDVSZvWbnox3vz8cH8OEse9pdsIJTKloIc_mOHQGBEVup08MNMytDbglCXYy8vWNwfbcsjLGA_n8nfxD9glT1kq7Jep9Nwj5zI9jrYGBLq_FppiIInU5MiCvYpLEhZpv9DDWxr8sIam1d-PyYi6Rbse6COL3yfWu5V_dEXryXg3l8ak8oVQTjOn0SQQx7VMNqLCxbhUa3qwaHfT9eIJWy7j_ZDfmg-X6A4fsqR_0W00>

### SEQUENCE DIAGRAM



## PROTOTYPING

Conceptual version of book entry and data module

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## WIREFRAMES

**UI/UX Design: Includes wireframes and mockups of the user interfaces in the React.js frontend.**

#### Login Page

A screenshot of a computer screen

Description automatically generated

#### Book Entering Form

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

<http://localhost:3000/add-employee>

A screenshot of a computer

Description automatically generated

## USER ACCEPTANCE TESTS

### TEST SUMMARY

|  |  |  |  |
| --- | --- | --- | --- |
| **TEST NO.** | **TEST MODULE** | **TEST SUMMARY** | **TEST OUTCOME** |
| TC1 | Authentication module | Can user authenticate and log into programme? | PASSED |
| TC2 | Security testing | Can user log in with a wrong Id and Password? | PASSED |
| TC3 | POST method | Can staff create new book entry? Does it appear at database too? | PASSED |
| TC4 | GET module | Can staff view all books? | PASSED |
| TC5 | PUT module | Can staff change or update book details? | PASSED |
| TC6 | Search module | Can staff look for a particular book? | PASSED |
| TC7 | Staff management module | Can manager access staff management module? | PASSED |
| TC8 | Staff management module | Can manager add new staff? |  |

### TEST CASES

Test Case #1

**Test Scenario:** Test authentication module. Can user authenticate and log into programme?

**Test Data:** UID:User | Password: 7209edea-95f4-4b5e-9425-42a139151647

Screenshots:  
A screenshot of a computer login page

Description automatically generated

**Outcome:**  
Success. User is able to log in with given ID and password.

Test Case #2

**Test Scenario:** Security testing. Can user log in with a wrong Id and Password?

**Test Data:** UID:abc | Password: 123

Screenshots:  
A screenshot of a computer screen

Description automatically generated

**Outcome:**  
Success. User is denied access with incorrect ID and password.

Test Case #3

**Test Scenario:** Test POST method. Can staff create new book entry? Does it appear at database too?

**Test Data:** Id: 15 , Title: “Breaking Dawn”, Author: “Stephenie Meyer”, Category: ”Friction” , Pages:756

Screenshots:

A screenshot of a computer

Description automatically generated

Screenshots:  
A screenshot of a computer

Description automatically generated

**Outcome:**  
Success. User is able to insert or create book data from form into database

Test Case #4

**Test Scenario:** Test GET module. Can staff view all books?

**Test Data:** http://localhost:8080/api/getallbooks

Screenshots:  
A screenshot of a computer

Description automatically generated

Screenshots:A screenshot of a computer

Description automatically generated

**Outcome:**  
Success. Staff able to view all library books

Test Case #5

**Test Scenario:** Test PUT module. Can staff change book detail in database?

**Test Data:** Author Name from “Jim Butcher” to “Kerrie L. Hughes”

Screenshots:  
Step 1 – Locate book entry  
A screenshot of a computer

Description automatically generated

Screenshots:  
Step 2 – Confirm book data in database  
A screenshot of a computer

Description automatically generated

Screenshots:  
Step 3 – Fill in required changes to book  
A screenshot of a computer

Description automatically generated

Screenshots:  
 Step 4 – Hit Refresh icon. Observe change at MySQL database  
A screenshot of a computer

Description automatically generated

**Outcome:**  
Success. Staff able to change book detail.

Test Case #6

**Test Scenario:** Book Module Test. Can staff search for a particular book?

**Test Data:** localhost:8080/api/getbook?Id=15

Screenshots:  
A screenshot of a computer

Description automatically generated

**Outcome:**  
Success. Staff able to find a book in LMS

Test Case #7

**Test Scenario:** Test Staff Management Module. Can manager access the staff management module?

**Test Data:** localhost:3000

Screenshots:  
A screenshot of a computer

Description automatically generated

**Outcome:**  
Success. Manager able to access the LMS Staff Management App

## USER MANUAL

## FOR STAFF

### TO LOG IN

Step 1 - With browser of choice, go to localhost:8080/login

A screenshot of a computer screen

Description automatically generated

Step 2 – Provide User ID as User

Step 3 – Obtain 1 time system generated secure password from console

A screenshot of a computer program

Description automatically generated

Step 3 – Provide Id, Password and click Sign in to complete signing in.

LIST ALL BOOKS  
  
Step 1 – Open preinstalled Postman App

Step 2 – Authenticate

Step 3 – Select GET method, type localhost:8080/api/getbooks

A screenshot of a computer

Description automatically generated

Step 4 – Click Visualize to view all library books

A screenshot of a computer

Description automatically generated

Source code:  
var template = `

<style type="text/css">

    .tftable {font-size:14px;color:#333333;width:100%;border-width: 1px;border-color: #87ceeb;border-collapse: collapse;}

    .tftable th {font-size:18px;background-color:#87ceeb;border-width: 1px;padding: 8px;border-style: solid;border-color: #87ceeb;text-align:left;}

    .tftable tr {background-color:#ffffff;}

    .tftable td {font-size:14px;border-width: 1px;padding: 8px;border-style: solid;border-color: #87ceeb;}

    .tftable tr:hover {background-color:#e0ffff;}

</style>

<table class="tftable" border="1">

    <tr>

        <th>ID</th>

        <th>Title</th>

        <th>Author</th>

        <th>Pages</th>

        <th>ISBN</th>

        <th>Category</th>

    </tr>

    {{#each response}}

        <tr>

            <td>{{id}}</td>

            <td>{{title}}</td>

            <td>{{author}}</td>

            <td>{{pages}}</td>

            <td>{{isbn}}</td>

            <td>{{category}}</td>

        </tr>

    {{/each}}

</table>

`;

function constructVisualizerPayload() {

    return { response: pm.response.json() }

}

pm.visualizer.set(template, constructVisualizerPayload());

ENTER BOOK  
  
Step 1 – Using web browser of choice, go to localhost:8080/mnlform

Step 2 – If prompted for authentication authenticate with user ID and password

Step 3 – Provide needed information on screen like shown

A screenshot of a computer

Description automatically generated

Step 4 – Click Submit to complete book entering.

## FOR MEMBERS

**Project Success Criteria:**

* The system meets all functional and non-functional requirements.
* The system is successfully deployed and used by librarians, students, and faculty members.
* The system receives positive feedback from users regarding its ease of use and functionality.
* The system demonstrates improved efficiency in library management operations.