**LIBRARY MANAGEMENT SYSTEM**

***77Library***

**Revision History**

| **Version** | **Date** | **Author** | **Description** |
| --- | --- | --- | --- |
| 1.00.0 | 17/11/23 | Sky | Brainstorm layout of webpage |
| 1.00.1 | 18/11/23 | Sky | Design and drafting of webpage |
| 1.00.2 | 19/11/23 | Sky | Finalise on webpage layout |
| 1.01.0 | 20/11/23 | Sky | Brainstorm entities/attributes required for database |
| 1.01.1 | 21/11/23 | Sky | Creation of Database via MySQL |
| 1.01.2 | 22/11/23 | Sky | Complete drafting entities and attributes in tables on MySQL, Update ERD on BRD. |
| 1.02.0 | 25/11/23 | Sky | Start working on ReactJS for frontend development |
| 1.02.1 | 26/11/23 | Sky | Creation of js/css files required for webpage content |
| 1.02.2 | 27/11/23 | Sky | Continued creation of js/css files. |
| 1.02.3 | 28/11/23 | Sky | Continued creation of js/css files. |
| 1.02.4 | 30/11/23 | Sky | Amendments on MySQL. Use case diagram completed. Update progress on BRD |
| 1.03.1 | 02/12/23 | Sky | Drafting UI/UX design, wireframes |
| 1.03.2 | 03/12/23 | Sky | Wireframes completed. |
| 1.04.1 | 05/12/23 | Sky | Initialisation of Springboot application |
| 1.04.2 | 18/12/23 | Sky | Springboot application integration with mysql completed |
| 1.05.1 | 18/12/23 | Sky | Integrating ReactJS to Springboot |
| 1.05.2 | 21/12/23 | Sky | Integration complete, application functions operating. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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

Contents

[BUSINESS REQUIREMENTS DOCUMENT 3](#_Toc151923916)

[PURPOSE 3](#_Toc151923917)

[OBJECTIVES 3](#_Toc151923918)

[RULES 3](#_Toc151923919)

[SCOPE 3](#_Toc151923920)

[ASSUMPTIONS 3](#_Toc151923921)

[CONSTRAINTS 3](#_Toc151923922)

[FUNCTIONAL REQUIREMENTS 4](#_Toc151923923)

[NON-FUNCTIONAL REQUIREMENTS 4](#_Toc151923924)

[PROJECT DELIVERABLES 4](#_Toc151923925)

[SYSTEM REQUIREMENTS SPECIFICATION 4](#_Toc151923926)

[DESIGN DOCUMENTS 5](#_Toc151923927)

[DATABASE DESIGN 5](#_Toc151923928)

[UML DIAGRAMS 5](#_Toc151923929)

[WIREFRAMES 5](#_Toc151923930)

# BUSINESS REQUIREMENTS DOCUMENT

*[Modify as you deem necessary]*

## PURPOSE

To develop a comprehensive library management system that utilizes Spring Boot for the backend, ReactJS for the frontend, and MySQL as the database. The system will facilitate the efficient management of library resources, including books, users, and borrowing activities.

## OBJECTIVES

1. The system meets all functional and non-functional requirements.
2. The system is successfully deployed and used by librarians, students, and faculty members.
3. The system demonstrates improved efficiency in library management operations.

RULES

1. **Book Management:**
   * **Rule 1:** A book can be added to the system only if it has a unique ISBN.
   * **Rule 2:** The system should not allow the deletion of a book that is currently checked out.
2. **Member Management:**
   * **Rule 3:** Members must have a unique member ID.
   * **Rule 4:** Only librarians and administrators can add or remove members from the system.
3. **Transaction Rules:**
   * **Rule 5:** A member can check out a maximum of three books at a time.
   * **Rule 6:** Each book must be checked out for a maximum of 14 days.
   * **Rule 7:** Overdue books will incur a fine of $0.50 per day.
4. **Reporting:**
   * **Rule 8:** Monthly reports should be generated on the 1st day of each month.
   * **Rule 9:** Reports should include information on the most borrowed books and outstanding fines.
5. **Security Rules:**
   * **Rule 10:** Only administrators can modify user roles and permissions.
   * **Rule 11:** User passwords must be encrypted and meet security standards.
6. **General System Rules:**
   * **Rule 12:** The system should be accessible 24/7, with scheduled maintenance communicated in advance.
   * **Rule 13:** Backups should be performed daily at midnight.
7. **Notification Rules:**
   * **Rule 14:** Librarians should receive a daily report of overdue books.

## SCOPE

The primary target users of the system are librarians, who will utilize the system to manage the library's collection, handle user accounts, and oversee borrowing and return processes. Additionally, students and faculty members will utilize the system to browse the library's catalogue, search for books, and place borrowing requests.

### ASSUMPTIONS

1. The library only operates from 10am to 9pm Mondays to Saturdays.

# FUNCTIONAL REQUIREMENTS

* FR#001 – **User Management:**
  + Create, edit, and delete user accounts
  + Manage user roles and permissions
  + Track user borrowing history
* FR#002 – **Book Management:**
  + Add, edit, and delete book records
  + Categorize books using genres and subjects
  + Maintain book availability status
* FR#003 – **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
* FR#004 – **Reporting and Analytics:**
  + Generate reports on book usage and borrowing trends
  + Analyze user borrowing patterns
  + Track overdue books and generate notifications

# NON-FUNCTIONAL REQUIREMENTS

* NFR#001 – **Performance:**
  + The system should be able to handle many concurrent users without performance degradation
  + Book search and filtering operations should be efficient and responsive
* NFR#002 – **Security:**
  + Implement secure user authentication and authorization mechanisms
  + Protect sensitive data, such as user information and borrowing records
  + Prevent unauthorized access to library resources
* NFR#003 – **Usability:**
  + The system should have a user-friendly interface that is easy to navigate and understand
  + Provide clear instructions and guidance for users
  + Implement consistent design patterns across the frontend and backend

# 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
* **Documentation:**
* **Business Requirements Document (BRD):**
  + **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.**
* **System Requirements Specification (SRS):**
  + **The SRS document provides detailed technical specifications, including functional and non-functional requirements. It describes the system's behaviour, interfaces, data structures, and more.**
* **Design Documents:**
  + **System Architecture Design: Describes the high-level architecture of the system, including how the frontend and backend components interact, and the database structure.**
  + **UML: Use case, Class/Object diagram;**
  + **Database Design: Contains details about the database schema, including tables, relationships, and data constraints and ER Diagram.**
  + **UI/UX Design: Includes wireframes and mockups of the user interfaces in the React.js frontend.**
* **User Manuals:**
  + **User Guide: A document for end-users (librarians and administrators) explaining how to use the system. It should cover tasks like user registration, book management, and reporting.**
  + **Librarian Manual: Provides detailed instructions for librarians on managing book borrowings and returns.**

# SYSTEM REQUIREMENTS SPECIFICATION

**Functional Requirements:**

1. **User Management:**
   * Create, update, and delete user accounts (librarians, admin, and members).
   * Define user roles and permissions.
2. **Book Management:**
   * Add, update, and delete books.
   * Record book details (title, author, ISBN, genre, etc.).
   * Track book availability and status (checked out, available, etc.).
3. **Borrowing and Returning:**
   * Check in/out books for library members.
   * Set borrowing limits and due dates.
   * Send overdue reminders.
4. **Search and Browse:**
   * Allow users to search for books using different criteria.
   * Browse books by genre, author, etc.
5. **Reporting:**
   * Generate reports on book availability, user activity, etc.

**Non-Functional Requirements:**

1. **Performance:**
   * System should handle concurrent users.
   * Quick response times for search and transactions.
2. **Security:**
   * Secure user data and login information.
   * Implement role-based access control.
3. **Scalability:**
   * System should be scalable to accommodate a growing number of books and users.
4. **Reliability:**
   * Minimal downtime and data integrity assurance.
5. **Usability:**
   * User-friendly interface.
   * Intuitive navigation for users of all levels.
6. **Compatibility:**
   * Compatible with different devices and browsers.
7. **Backup and Recovery:**
   * Regularly backup data and implement a recovery plan.
8. **Integration:**
   * Integration with other library systems or databases.
9. **Compliance:**
   * Comply with data protection regulations (GDPR, etc.).
10. **Documentation:**
    * Well-documented code and user manuals.

**Hardware Requirements:**

1. **Server:**
   * Multi-core processor
   * Ample RAM for concurrent user handling.
   * Sufficient storage for the database.
2. **Client Machines:**
   * Standard computers for librarians and admin users.
   * Compatibility with Google Chrome browser for member access.

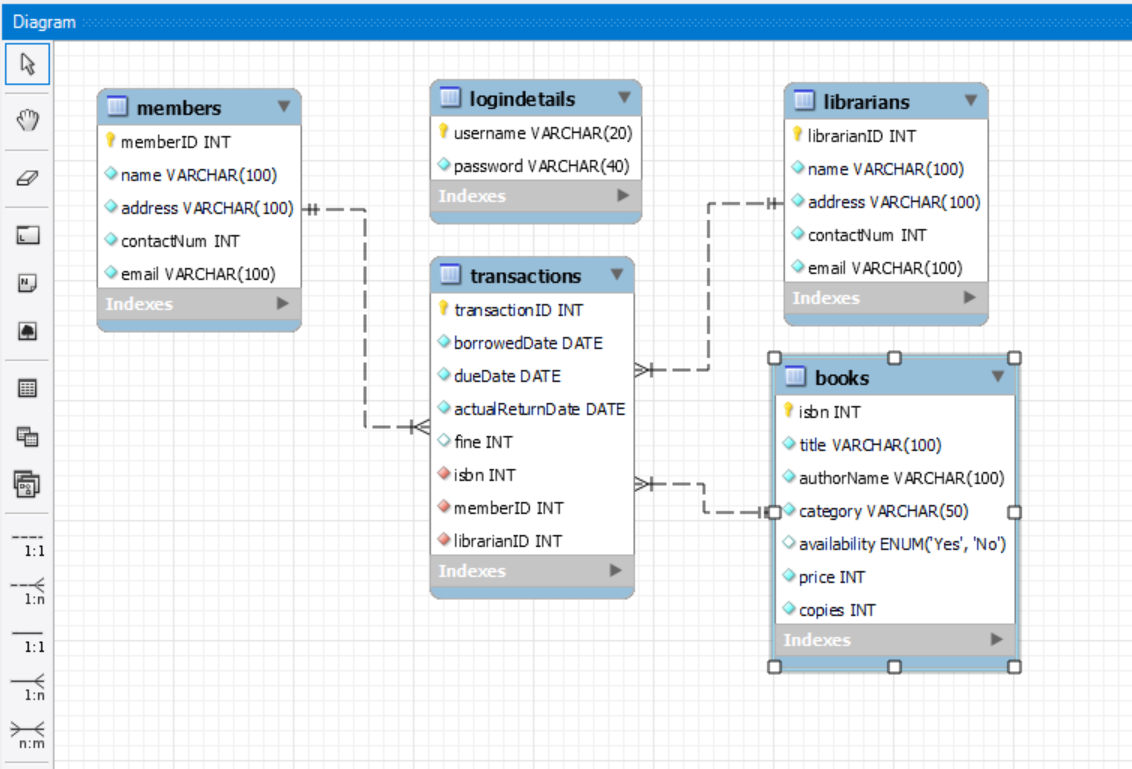
**Software Requirements:**

1. **Operating System:**
   * Server: Linux or Windows Server.
   * Clients: Windows, macOS, or Linux.

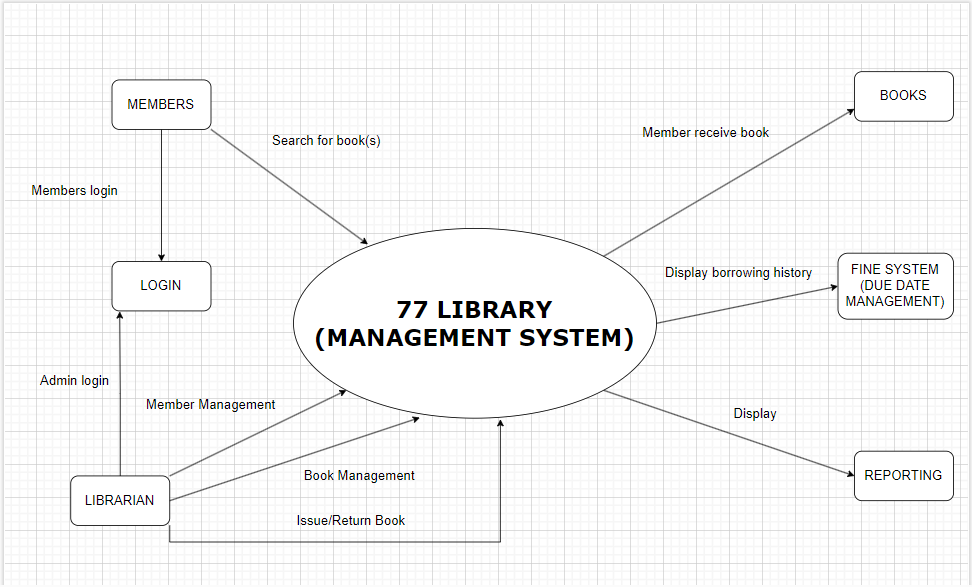
# DESIGN DOCUMENTS

1. **Presentation Tier (Client Side):**
   * **Web-based Interface:**
     + Developed using HTML, CSS, and JavaScript.
     + Responsive design for various devices.
   * **User Interface (UI) Components:**
     + Front-end framework (ReactJs).
     + Implements user interactions and displays information.
2. **Application Tier (Server Side):**
   * **Application Server:**
     + Hosts the business logic and application processes.
     + Developed using a suitable backend framework (Spring).
   * **Authentication and Authorization:**
     + Implement a secure authentication system.
     + Role-based access control for different user types.
   * **Business Logic Layer:**
     + Handles book management, user interactions, and transactions.
     + Validates and processes user requests.
   * **APIs:**
     + RESTful APIs for communication between the client and server.
     + Endpoints for user actions, book queries, and system updates.
   * **Integration Components:**
     + Connects with external databases for book information.
     + Utilizes APIs for potential integrations.
3. **Data Tier (Database):**
   * **Database Management System (DBMS):**
     + MySQL
   * **Data Storage:**
     + Store information about books, users, transactions, etc.
     + Ensure data integrity and consistency.
   * **Backup System:**
     + Regular automated backups to prevent data loss.
4. **External Services:**
   * **Security Services:**
     + SSL for secure communication.
     + Regular security audits and updates.

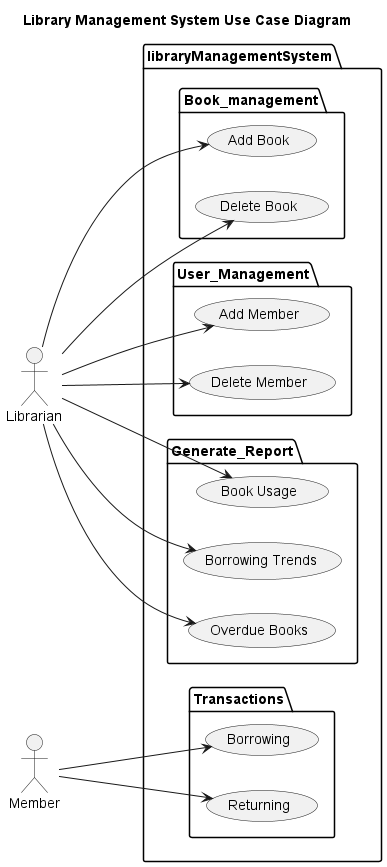
## **DATABASE DESIGN**

******

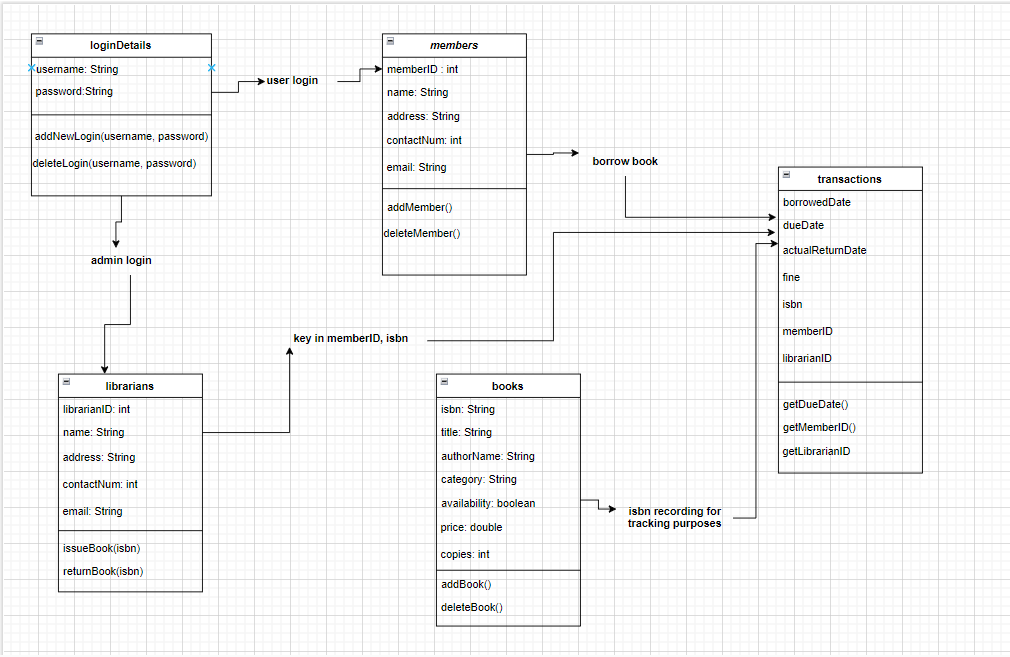
## **SYSTEM DIAGRAM**



## UML DIAGRAMS



## CLASS DIAGRAM



## WIREFRAMES

***Refer to Wireframes.pptx***