**Capstone Project Documentation**

**Business Requirements**

Project Goals:

The goal of this project is to create a new Library Management System (LMS), which is a web application designed to streamline library operations for staff and enhance user experience.

Target audience:

1. Librarians: To efficiently manage books, members, and borrowing/returning processes.
2. Members: To easily search for books and view their current loans as well as any outstanding fines.

Objectives:

1. Develop a fully functional web application using Spring Boot, React and MySQL.
2. Enable librarians to efficiently and easily manage books and members.
3. Provide a user-friendly interface for members to search for books and view their current loans and outstanding fines.
4. Create a robust and scalable system that can handle a moderate number of books and members. In this project, a moderate number is defined as up to 15,000 books and 1,000 members.

**Functional Requirements**

1. There will be 3 user roles: appointed System Administrator for the setup and maintenance of the full stack web application, Staff Admin (Librarian) and Member.
2. Features vs role access: Table 1 shows the functions or features that can be performed by the respective user roles.

**Table 1**

| **Features** | | | | | | | | | **System Admin** | | | | | **Staff Admin** | | | **Member** | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. **System management** | | | | | | | | | | | | | | | | | | | |
| 1. Manage system settings and configurations | | | | | | | | | ✔️ | | | | |  | | |  | | |
| 1. Librarian account registration  * Create/modify/delete librarian accounts * Note: librarian is also a member by default (i.e., a member with staff permissions) | | | | | | | | | ✔️ | | | | |  | | |  | | |
| 1. Maintain database integrity and backups | | | | | | | | | ✔️ | | | | |  | | |  | | |
| 1. **Member management** | | | | | | | | | | | | | | | | | | | |
| 1. User registration  * Register new members with personal details (name, address, contact information, etc.) | | | | | | | | |  | | | | | ✔️ | | |  | | |
| 1. User profile management  * Edit member information, including password reset | | | | | | | | |  | | | | | ✔️ | | |  | | |
| 1. Update/delete members from the system | | | | | | | | |  | | | | | ✔️ | | |  | | |
| 1. Search for members by name or ID | | | | | | | | |  | | | | | ✔️ | | |  | | |
| 1. Generate/View full member list | | | | | | | | |  | | | | | ✔️ | | |  | | |
| 6. User login/logout | | | | | | | | |  | | | | | ✔️ | | | ✔️ | | |
| **Features** | | | | | | | | | **System Admin** | | | | | **Staff Admin** | | | **Member** | | |
| 1. **Book management** | | | | | | | | |  | | | | |  | | |  | | |
| 1. Add, update, delete books  * Book details must include: ISBN, title, author, category, publication year, total copies | | | | | | | | |  | | | | | ✔️ | | |  | | |
| 1. View, search and filter books by title or author | | | | | | | | |  | | | | | ✔️ | | | ✔️ | | |
| 1. Track book status (available, borrowed, reserved, overdue) and generate list | | | | | | | | |  | | | | | ✔️ | | |  | | |
| **D. Lending management** | | | | | | | | |  | | | | |  | | |  | | |
| 1. Borrow, return and renew books for members | | | | | | | | |  | | | | | ✔️ | | |  | | |
| 1. View active loans, due dates, fines by member’s username | | | | | | | | |  | | | | | ✔️ | | |  | | |
| 1. View active loans and due date tracking  * Member is able to view due date of borrowed books in his/her account | | | | | | | | |  | | | | |  | | | ✔️ | | |
| 1. Automatic fine calculation  * Member is able to view fine in his/her account (member to pay fine at the library) | | | | | | | | |  | | | | |  | | | ✔️ | | |

1. Business rules:
   1. Membership rules:
      1. Members can borrow a maximum of 3 books at any time.

* Members cannot borrow if active loan items = 3 books
  + 1. Membership is valid for 1 year from registration
       - Start of membership = Date of registration
       - End of membership = 1 year from date of registration
    2. Members must have valid membership to borrow books
  1. Lending rules:
     1. Loan duration: 14 days
     2. Maximum of 2 renewals per book
     3. Member cannot borrow if having overdue books or accumulated fines exceeding $10
  2. Fine calculation:
     1. $0.50 per day for overdue books
     2. Fine starts accumulating from day after due date
     3. Maximum fine per book: $20

**Software Requirements Specification**

1. Technology stack used:
   1. Backend: Spring Boot (Java)
   2. Database: MySQL
   3. Frontend: React
   4. Version Control: Git
2. Backend
   1. MySQL database
   2. Spring Boot framework
   3. RESTful API architecture
   4. Spring Security for authentication
   5. JPA/Hibernate for database operations
3. Frontend (React)
   1. React router for navigation
   2. React hooks for state management
   3. Bootstrap or Material-UI for styling
   4. Axios for API communication
4. Database requirements
   1. Proper relationship mapping
   2. Referential integrity
   3. Appropriate indexing
   4. Data validation rules
5. Security requirements
   1. Password encryption
   2. Role-based access control
   3. JWT token-based authentication
   4. Input validation and sanitization

**System Architecture**

**Architecture type**: Model-View-Controller (MVC)

MVC (Model–View–Controller) helped in system implementation by providing a clear separation of concerns, which made the development, testing, and maintenance of the system more structured, modular and efficient.

* **Model** handled the data and business logic (e.g., database queries, loan rules in a library system)
* **View** managed how information was presented to users (e.g., React components)
* **Controller** acted as the bridge, processing user inputs, updating models, and returning the correct views

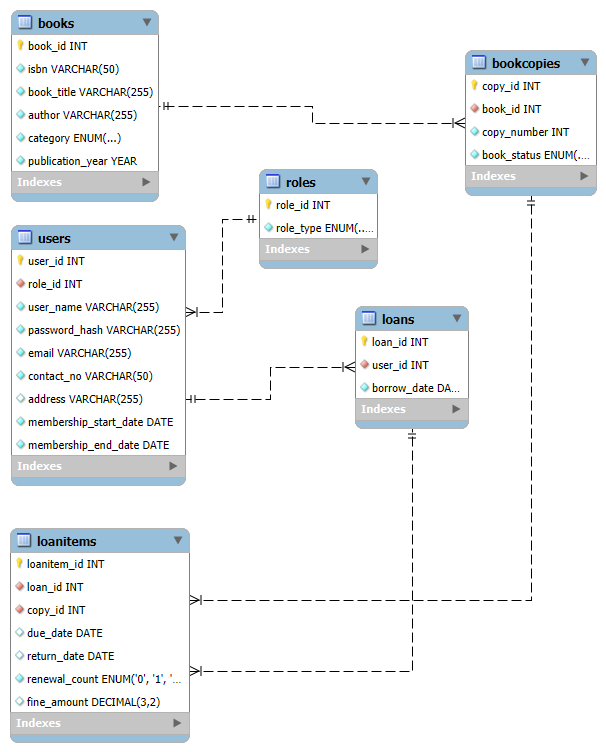
Benefits:

1. Flexibility and scalability
   1. UI (View) could be redesigned without touching business logic
   2. Business rules in the Model could be updated without changing the UI
   3. Controller ensured smooth communication, so scaling or adding new features became easier
2. Code reusability
3. Improved debugging and testing

**Deliverable Acceptance Criteria**

1. Acceptance criteria
   1. All core features implemented and functional
   2. Code follows provided coding standards (refer to coding document)
   3. Basic test coverage
   4. Proper error handling
   5. Completed documentation
   6. No critical bugs

**MySQL ERD**

****

Legend:

|  | This denotes one-to-many relationship. |
| --- | --- |