Design Document

Table of Contents

[1. System Architecture Design 2](#_mkugwyrxmoy9)

[Backend Architecture 2](#_lksuormaawec)

[Frontend Architecture 2](#_cejbpmd1amcn)

[Interaction between Backend and Frontend 2](#_rf5ldwx1v73d)

[2. Database Design 2](#_7llfmhm5euc7)

[Entity-Relationship Diagram (ERD) 2](#_b4mhm8g0hkis)

[Table Definitions 2](#_9uwv9vdsbum6)

[Data Constraints and Relationships 3](#_75tpgqng3ohk)

[3. UI/UX Design 3](#_xdgzbn2k6a4i)

[Wireframes 3](#_toihw0lfmzmq)

[Mockups 3](#_b871pzifxhc6)

[Design Consistency 3](#_rla6a8g46pse)

[4. Deployment Strategy 3](#_3spxc02e28y6)

[Backend Deployment 3](#_e69pzkh5us45)

[Frontend Deployment 3](#_kvpu6yiike1s)

[5. Integration Plan 4](#_esv0b18hv7zq)

[Backend-Frontend Integration 4](#_fjmz7l6guua)

[Frontend-Database Integration 4](#_v6bhoopdr6jy)

[6. Technology Stack 4](#_nftigeyhlyke)

[Frameworks and Technologies Used 4](#_yfx4vc3v3h05)

[Justification 4](#_h5x863j82g7w)

# 

# **Library Management System**

This Design Document provides an overview of the architectural and design aspects of the Library Management System (LMS) - Liby. It focuses on detailing the system's high-level architecture, database structure, and user interface design.

### **1. System Architecture Design**

The system architecture of Liby outlines the interaction between the backend, frontend, and database components.

#### Backend Architecture

The backend of Liby follows a modular structure with Spring Boot. It includes controllers, services, repositories, and security configurations. RESTful APIs manage CRUD operations for entities like Users, Books, and Borrows. Security configurations ensure endpoint protection based on user roles.

#### Frontend Architecture

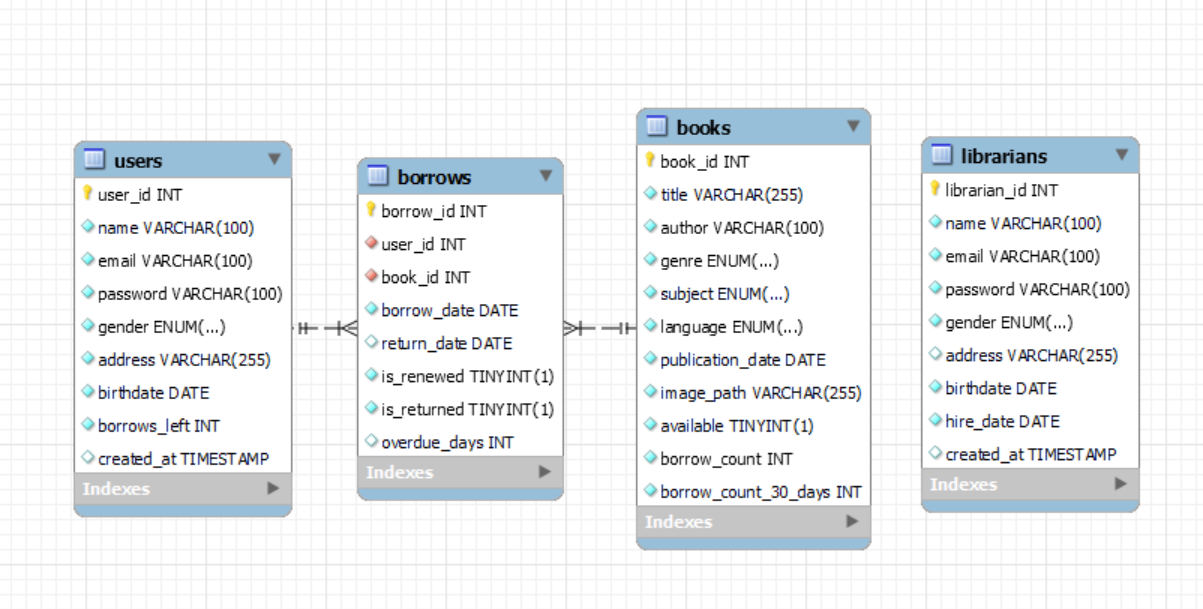
ReactJS powers the frontend with components such as NavBar, LogIn, SignUp, BrowseBooks, BorrowBooks, and ManageUsers. These components interact with backend APIs through HTTP calls, managing data flow and UI updates.

#### Interaction between Backend and Frontend

Communication between backend APIs and frontend components is achieved through RESTful architecture. APIs handle user authentication, book borrowing, and other functionalities. For instance, when a user logs in through the frontend LogIn component, it triggers an API call to the backend's authentication endpoint, passing credentials. Upon successful authentication, the backend responds with an authentication token or user data.

### **2. Database Design**

#### Entity-Relationship Diagram (ERD)



#### Table Definitions

##### Users Table

##### *Stores information about users who interact with the library system*

user\_id: Primary key for each user.

name: User's name.

email: User's email address (unique).

password: User's password.

gender: User's gender (Male, Female, Other).

address: User's address.

birthdate: User's date of birth.

borrows\_left: Number of books a user can currently borrow (default 8).

created\_at: Timestamp of user account creation.

##### Librarians Table

*Manages details of librarians working within the library system.*

librarian\_id: Primary key for each librarian.

name: Librarian's name.

email: Librarian's email address (unique).

password: Librarian's password.

gender: Librarian's gender (Male, Female, Other).

address: Librarian's address.

birthdate: Librarian's date of birth.

hire\_date: Date when the librarian was hired.

created\_at: Timestamp of librarian account creation.

##### Books Table

*Holds comprehensive data about the books available in the library.*

book\_id: Primary key for each book.

title: Title of the book.

author: Author of the book.

genre: Genre of the book (Fiction, Non-fiction).

subject: Subject of the book (e.g., Literature, Science, etc.).

language: Language of the book.

publication\_date: Date when the book was published.

image\_path: Path to the book's image.

available: Availability status of the book (default TRUE).

borrow\_count: Total number of times the book has been borrowed (default 0).

borrow\_count\_30\_days: Number of times the book was borrowed in the last 30 days (default 0).

##### Borrows Table

*Tracks borrowing transactions made by users.*

borrow\_id: Primary key for each borrow record.

user\_id: Foreign key referencing the Users table.

book\_id: Foreign key referencing the Books table.

borrow\_date: Date when the book was borrowed.

return\_date: Date when the book was returned.

is\_renewed: Boolean indicating if the book borrow was renewed (default FALSE).

is\_returned: Boolean indicating if the book has been returned (default FALSE).

overdue\_days: Number of days the book was overdue (default 0).

These tables establish relationships between users, books, and borrow transactions, allowing for efficient management of book borrowing and returns.

#### Data Constraints and Relationships

##### Users Table

**Primary Key:** user\_id (Unique identifier for each user)

**Unique Constraint:** email (Ensures each email is unique)

**Data Validation Rules:**

* name, email, password, address: Must not be empty.
* birthdate: Must be a valid date.
* borrows\_left: Specifies the maximum number of books a user can borrow at a given time (default: 8).

##### Librarians Table

**Primary Key:** librarian\_id (Unique identifier for each librarian)

**Unique Constraint:** email (Ensures each email is unique)

**Data Validation Rules:**

* name, email, password: Must not be empty.
* birthdate, hire\_date: Must be valid dates.

##### Books Table

**Primary Key:** book\_id (Unique identifier for each book)

**Unique Constraint:** title (Ensures each title is unique)

**Data Validation Rules:**

* title, author, image\_path: Must not be empty.
* publication\_date: Must be a valid date.
* borrow\_count, borrow\_count\_30\_days: Counters for book borrowing.

##### Borrows Table

**Primary Key:** borrow\_id (Unique identifier for each borrowing transaction)

**Foreign Key:** user\_id (References user\_id in Users Table)

**Foreign Key:** book\_id (References book\_id in Books Table)

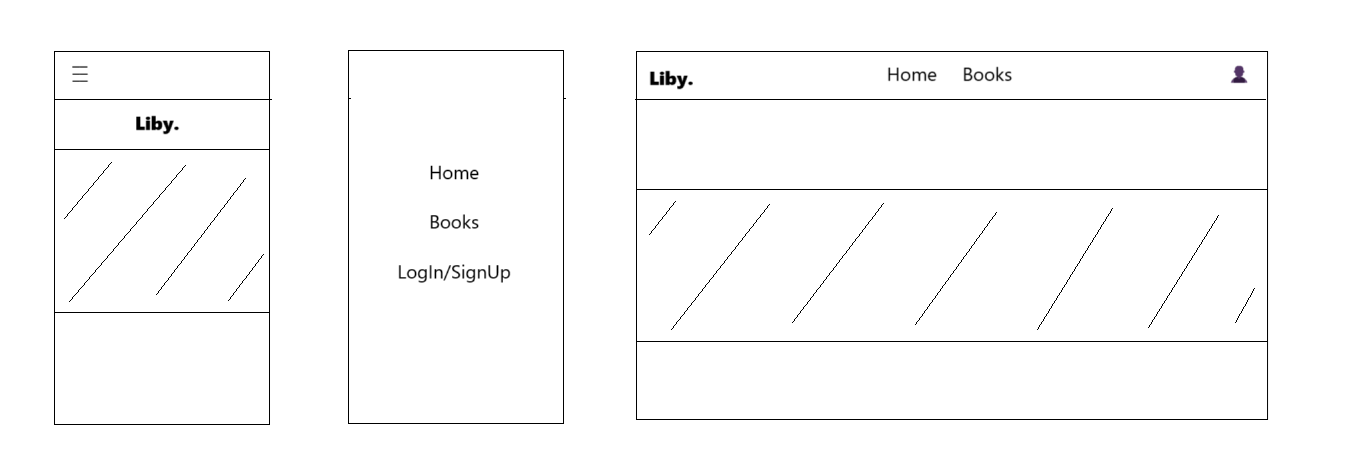
**Data Validation Rules:**

* borrow\_date: Must be a valid date.
* Due\_date: Must be a valid date. An auto generated value.
* return\_date: Must be a valid date and can be NULL until the book is returned.
* is\_renewed: Boolean flag indicating if a book is renewed.
* is\_returned: Boolean flag indicating if a book is returned.
* overdue\_days: Tracks the number of days a book is overdue.

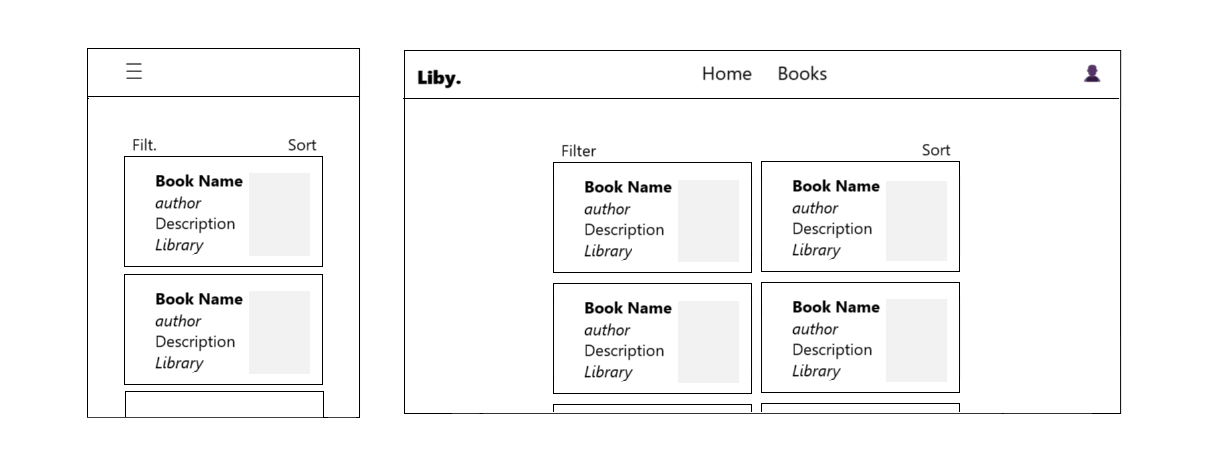
### **3. UI/UX Design**

#### Wireframes

Landing Page/ Navigation



Book Browsing



#### Design Consistency

Ensures that consistent design patterns and elements are across the frontend for a cohesive user experience.

### **4. Deployment Strategy**

#### Backend Deployment

TBA:

[ Explanation of deployment strategies for the Spring Boot application, considering scalability and reliability.]

#### Frontend Deployment

TBA:

[ Details on deploying the ReactJS application, including routing and navigation considerations.

Database Deployment

Guidelines for deploying and managing the MySQL database, ensuring data integrity and security. ]

### **5. Integration Plan**

#### Backend-Frontend Integration

Steps and protocols for integrating the frontend with the backend, focusing on data exchange and API usage.

#### Frontend-Database Integration

Details on integrating the frontend with the MySQL database, ensuring smooth data retrieval and manipulation.

### **6. Technology Stack**

#### Frameworks and Technologies Used

Spring Boot, a Java-based framework, simplifies and accelerates the development of robust, enterprise-grade applications. It stands out for its convention-over-configuration approach and extensive ecosystem, offering a range of tools like Spring Data and Spring Security. Its scalability and strong community support make it an ideal choice for complex projects. JDK allows it to run across all devices seamlessly.

ReactJS, developed by Facebook, is a JavaScript library for building dynamic user interfaces. Its component-based architecture promotes reusable and maintainable UI components. Leveraging a Virtual DOM and a declarative syntax, React ensures efficient rendering and easier comprehension of code. Its rich ecosystem supports state management and routing, enhancing the development experience. ReactJS supports the use of libraries such as sass, font-awesome and Bootstrap for the easy integration of design components into the application in order to enhance the UI/UX experience.

MySQL is an open-source relational database, renowned for its reliability, scalability, and user-friendly interface. It ensures robust ACID-compliant transactions, making it suitable for diverse application scales. Its ease of use, strong performance, and extensive community support position it as a popular choice for managing relational data in web applications.

#### Justification

TBA:

[Reasons for selecting specific frameworks and technologies for different system components.]