**Library Inventory System**

**Software Requirements Specification**

**Version 2**

**Team Number: 10**

**Project Manager: Claudia Dorin**

**Mentor: Sandeep Ginna**

**Team Members: William Chalk, James Cotton, Lauren Mcneill, Fredrick Nkrumah**

**Revisions**

| **Version** | **Primary**  **Author(s)** | **Description of Version** | **Date Completed** |
| --- | --- | --- | --- |
| 1 | Claudia Dorin, William Chalk, Frederick Nkrumah, Lauren Mcneill, James Cotton | Initial draft: Added project overview, use cases, user stories, scope, and all requirements | 3/9/25 |
| 1.1 | Claudia Dorin, William Chalk, Frederick Nkrumah, Lauren Mcneill, James Cotton | Edited for grammar/spelling, refined project description, converted use cases to diagrams | 3/16/25 |
| 2.1 | Claudia Dorin, William Chalk, Frederick Nkrumah, Lauren Mcneill, James Cotton | Initial draft: Added sections 6.1 through 6.9 for draft of version 2. |  |
| 2.2 | Claudia Dorin, William Chalk, Frederick Nkrumah, Lauren Mcneill, James Cotton | Edited for grammar/spelling, refined sections & updated diagram colors. |  |

**Review History**

| **Reviewer** | **Version Reviewed** | **Date** |
| --- | --- | --- |
| Sandeep Ginna | 1.1 | 3/16/25 |
| Sandeep Ginna | 2.1 |  |

**Table of Contents (version 1)**

1. Introduction

1.1. Project Objectives

1.2. Project Scope

1.3. Project Overview

2. Project Description

2.1. Project Features / Functions

2.2. User Stories

2.3. Use Cases

2.4. Project Assumptions and Dependencies

3. Project Collaboration and Documentation

4. Project Management

5. Requirements Specification

5.1. Business Requirements

5.2. User Requirements

5.3. Functional Requirements

5.4. Non-Functional Requirements

5.5. Implementation (Performance) Requirements (Optional)

**1.** **Introduction**

**1.1. Project Objectives**

The objective of the Library Inventory System software is to simplify book management for library staff, while enabling customers to easily search for specific books. The software will provide a user-friendly interface that will allow users to add, search, and delete books from the library inventory. The Library Inventory System software will provide real-time status of books, so users can view if a book is available or checked out. The software will be designed as a standalone system so that we can ensure a focused and efficient catalog management system; only the inventory of the library implementing it will be shown.

**1.2. Project Scope**

The Library Inventory System will provide a focused inventory management system for the library that implements it. By providing a reliable system, the library can inform customers what books they have on hand while noting what books are checked out by other customers. The software is designed to be a standalone system which will provide the inventory of the specific library that has implemented it with no external data sharing.

**In Scope:**

* Modern user-friendly interface.
* CRUD functionality for librarians to use.
* Users will have the ability to search for books by author, title, category, or ISBN.
* Track book availability and inventory status.
* Report generation

**Out Of Scope:**

* Database will not connect with external libraries
* Late fees and due dates will not be tracked

**1.3. Project Overview**

The outcome of the software is to create a reliable and manageable inventory system for libraries that is easy to use and learn. The software will achieve these goals by using industry standard database procedures to store and maintain the information of customers and books. The software will include a simple yet modern user interface which will aid librarians in adding, editing, and removing books from inventory as well as track customer information. The project is in the research phase and will require UX designers, front-end developers to implement the UI from the UX team, and back-end engineers to connect the database to the software and ensure a seamless relationship between the two.

**2.** **Project Description**

**2.1. Project Features / Functions**

### **2.1.1. Book Inventory Management**

This module is designed to streamline the management of library book records, exclusively accessible to administrators. It includes the following functionalities:

* *Add New Book Records:* Enable administrators to input detailed information about new books, administrators can enter detailed book information, including title, author, genre, ISBN, publication date, edition, publisher, and availability status.
* *Support for bulk book uploads:* Via CSV files to streamline large-scale data entry
* *Update Book Records*: Allow administrators to edit existing book details to ensure information remains accurate and up-to-date, such as correcting titles, changing genres, or updating availability. Change book status (e.g., "Checked Out," "On Hold," "Available").
* *Delete Book Records:* Facilitate the removal of obsolete or irrelevant book entries from the system. Modify book details to correct errors or update information. Change book status (e.g., "Checked Out," "On Hold," "Available").
* *Track Book Details:* Maintain a comprehensive database of book metadata, enabling seamless retrieval. Maintain a comprehensive book metadata database for easy retrieval. Store historical data on previous borrowers and lending duration.
* *Search and Filter Options:* Provide administrators with tools to quickly locate books using multiple criteria such as title, author, ISBN, or availability. Advanced search functionality allows books to be searched by title, author, ISBN, genre, publisher, and availability. Filters to refine searches by publication year, category, format (eBook, hardcover, etc.), and language.

### **2.1.2. User Management System**

This component focuses on the effective administration of library users and role-based permissions. Key features include:

* *User Registration:* Facilitate the creation of user profiles for library members and staff with relevant details such as name, contact information, and role. Allow library members, staff, and administrators to create accounts with unique credentials. Support for automated email verification during registration.
* *Profile Updates:* Allow administrators or users (depending on permissions) to modify profile information as needed. Users and administrators can modify personal details, such as contact information and password updates. Staff can update user roles as required.
* *Role-Based Access Control:* Define specific roles with tailored access rights, including:
  + Admin: Full access to all system functionalities, including book inventory management and reporting.
  + Staff**:** Limited management access (adding books, updating records, assisting members).
  + Member: Limited access to browsing, borrowing, and viewing account details.
* *Account Status Management*: Enable the activation, suspension, or deactivation of user accounts. Monitor user borrowing history.

### **2.1.3. Search and Status System**

#### **Advanced Search Functionality:**

* Enable users to search for books by title, author, genre, ISBN, or publisher.
* Implement auto-suggestion to help users find books faster.

#### **Filter Options:**

* Allow users to filter books by genre, availability status, author.

#### **Real-Time Availability Status:**

* Clearly indicate if a book is "Available" or "Unavailable" to minimize frustration.
* Display real-time book availability, including statuses such as “available,” “unavailable,” “removed,” or “reserved.”
* Notify users when a previously unavailable book becomes available for borrowing.

**2.2.**  **User Stories**

**2.2.1. Book Management**

**“As a library staff member, I want to add new books to the inventory, so that the library's collection is kept up-to-date.”**

*Criteria*: The system should allow staff to enter book details (title, author, ISBN, genre, and more).

*Expected Result*: A new book is successfully added and appears in the inventory list.

**“As a library staff member, I want to search for books by name, author, or ISBN, so that I can quickly find specific books for customers.”**

*Criteria*: The search function should return relevant results based on user input.

*Expected Result*: The staff can locate books accurately and efficiently.

**2.2.2. User Interface**

**“As a library staff member, I want a user-friendly interface for adding, searching, updating, and deleting books, so that I can manage the inventory efficiently without technical difficulties.”**

*Criteria*: The interface should be intuitive, visually accessible, and require minimal training..

*Expected Result*: Staff can perform inventory management tasks seamlessly.

**“As a library customer, I want to search for books by title, author, or ISBN, so that I can easily find books I am interested in borrowing.”**

*Criteria*: The system should provide keyword search and filtering options. Customers should be able to search for books without requiring staff assistance.

*Expected Result*: Customers can find books based on search criteria.

**2.2.3. Book Availability**

**“As a library staff member, I want to see the availability status of each book, so that I can inform customers about the current stock.”**

*Criteria*: The inventory list should clearly display book availability.

*Expected Result*: Staff can quickly identify which books are available or checked out.

**“As a library customer, I want to know if a book is available before visiting the library, so that I don't make unnecessary trips.”**

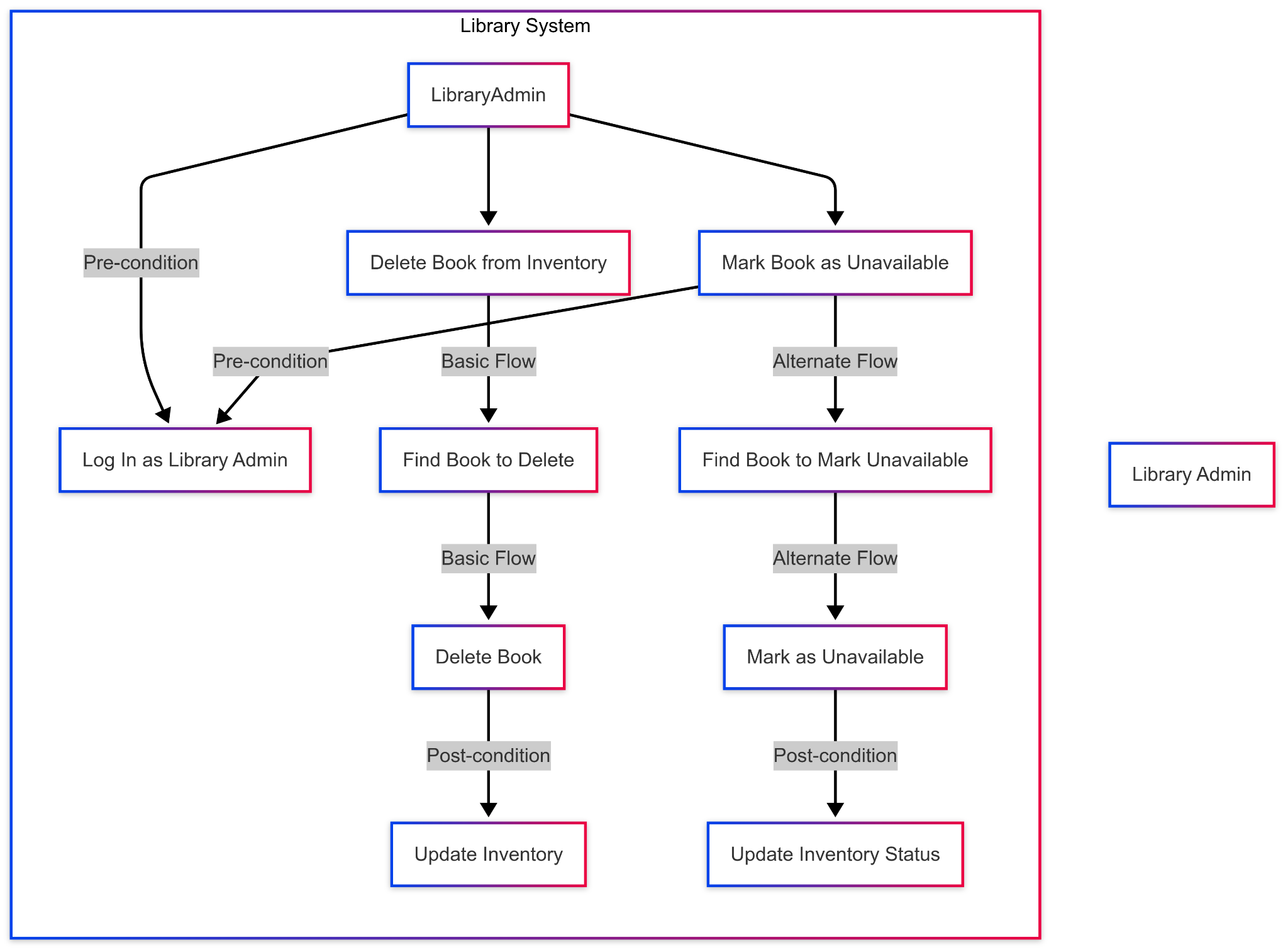
*Criteria*: The system should provide real-time status updates on book availability.

*Expected Result*: Customers can check availability online before coming to the library.

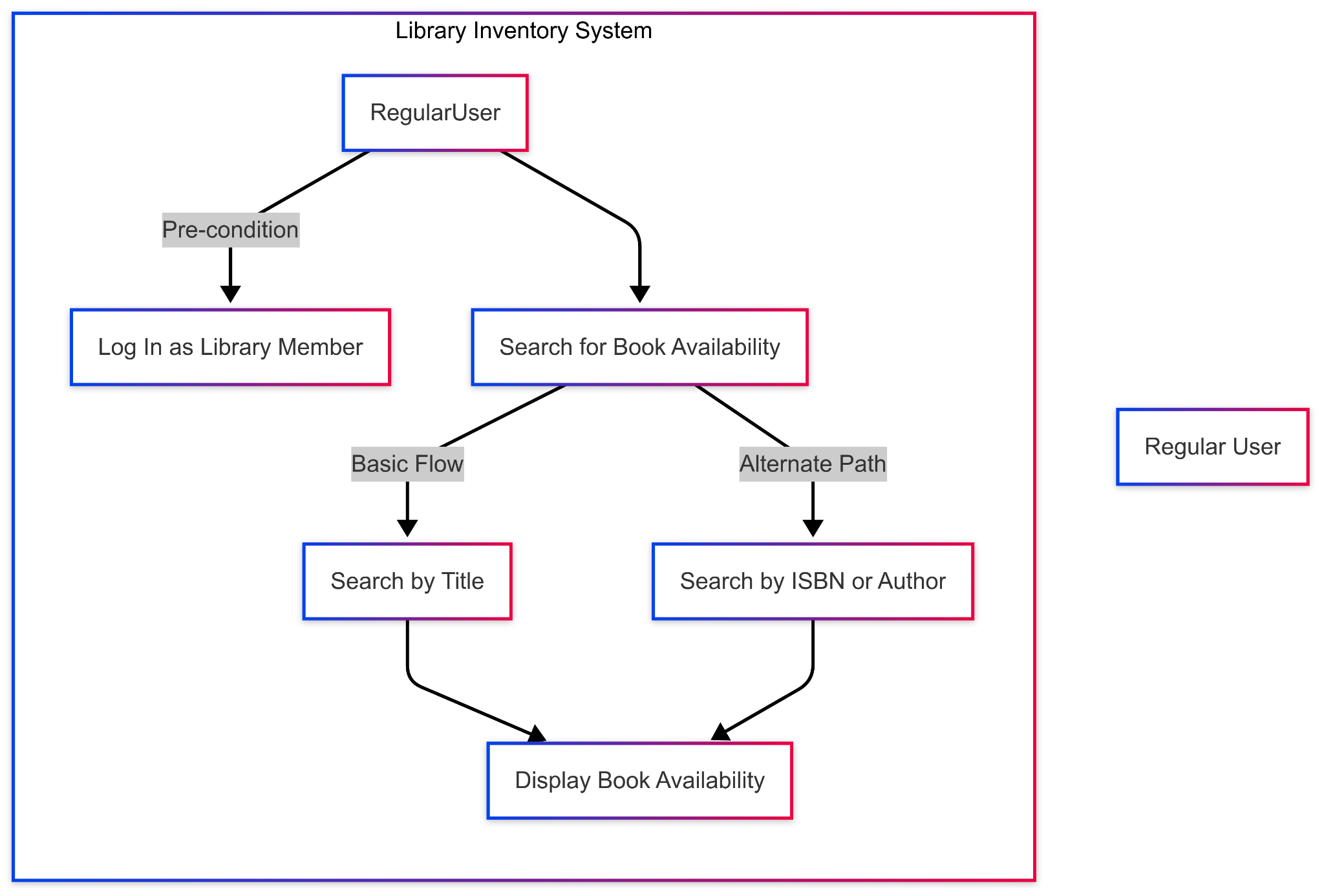
**2.3.** **Use Cases**

Use Cases describe a specific interaction between a user (or "actor") and the application to achieve a particular goal. It outlines the steps involved in this interaction and can help to clarify functional requirements. These use cases are written using mermaid syntax for readability.

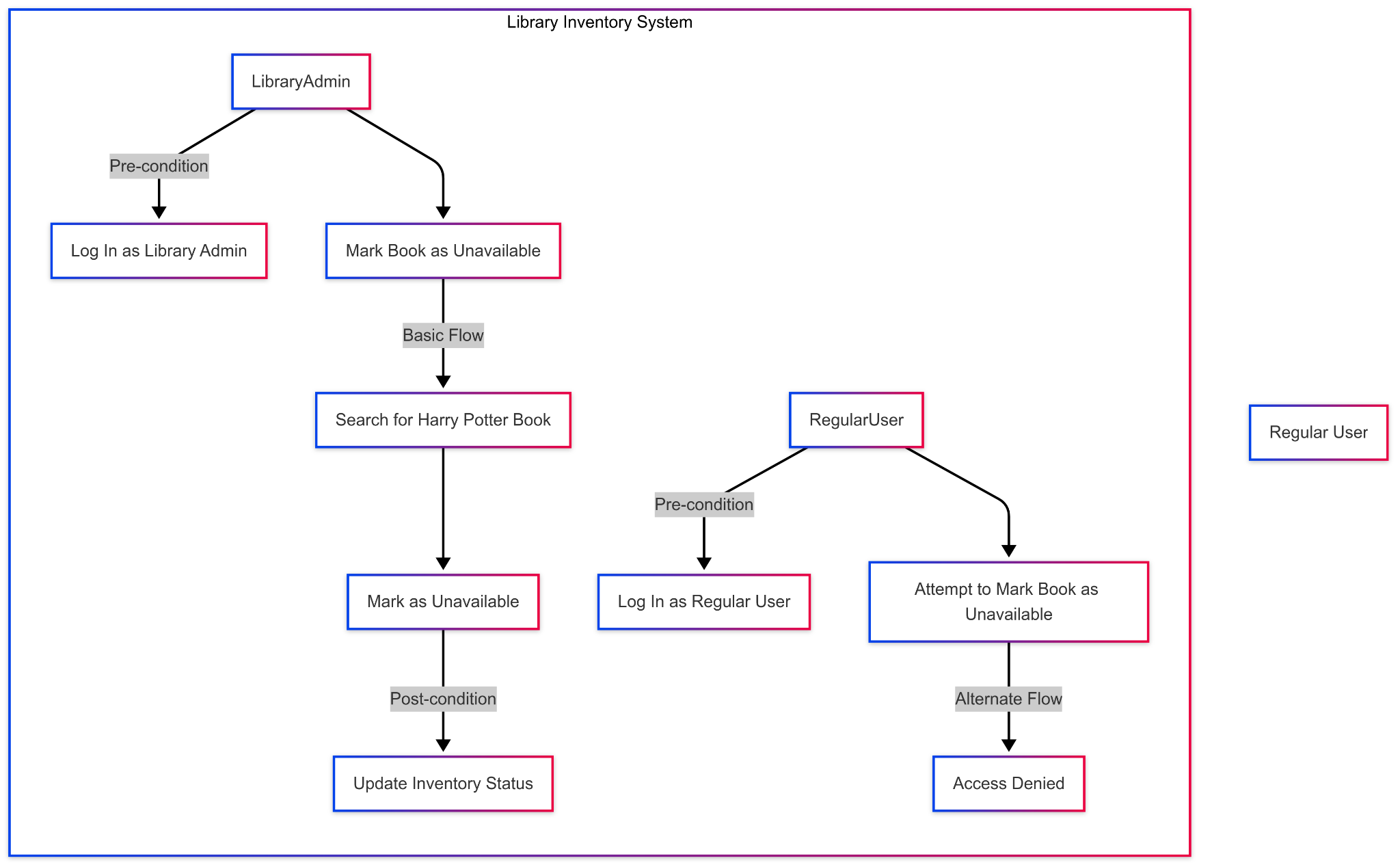
**2.3.1. Use Case Diagram 1**

****

**2.3.2. Use Case Diagram 2**

****

**2.3.3. Use Case Diagram 3**



**2.4. Project Assumptions and Dependencies**

Project Assumptions and Dependencies:

*Assumptions*:

* Stable Internet Connection: The system requires a reliable internet connection for database access, updates, and synchronization.
* Consistent Data Entry: Library staff will adhere to standardized data entry procedures to ensure accurate records.
* User Training: Staff and administrators will receive proper training on how to use the system effectively.
* Hardware Availability: Necessary hardware such as computers, barcode scanners, and printers will be available.
* Regular Maintenance: Routine software updates and maintenance will be conducted to ensure security and performance.

*Dependencies*:

* Database Management System (DBMS): The system will utilize a relational database (e.g., MySQL, PostgreSQL) for data storage and management.
* Third-Party Libraries/Frameworks: Frameworks such as Flask or Django will be used for backend development.
* Operating System Compatibility: The software will be designed to work on Windows, macOS, and Linux.
* External API Integration: The system may integrate with ISBN lookup APIs for book metadata retrieval.

**3.** **Project Collaboration and Documentation**

**Trello – Task Management and Agile Workflow**

Trello serves as our primary tool for project management, organized around a Kanban board with three core columns: "To Do," "Doing," and "Done." This layout allows the team to visually track task progression, set priorities, and maintain momentum throughout the project lifecycle.

* Each team member is responsible for updating their assigned cards to reflect the current task status.
* Labels, due dates, and checklists are used to enhance task clarity and accountability.
* The board will be reviewed and updated regularly during sprint planning and stand-up meetings.

**Lucidchart – Visual Documentation and System Design** Lucidchart will be utilized to create and maintain all technical diagrams, including:

* System architecture
* Data flow diagrams
* Class diagrams
* Entity-relationship models

These visuals help clarify complex system components and ensure shared understanding among team members. Lucidchart’s collaborative features enable multiple contributors to work on diagrams simultaneously, with version history available for change tracking.

**Microsoft Teams – Communication and Documentation Hub** MS Teams is our centralized communication platform, supporting:

* Real-time messaging for day-to-day questions and discussions.
* Video conferencing for scheduled meetings, sprint reviews, and ad-hoc problem-solving
* File sharing, ensuring that all documents are easily accessible in one shared workspace  
   All team meetings will be conducted via MS Teams, and minutes, key decisions, and task assignments will be recorded in dedicated channels for easy access and traceability.  
   Additionally, MS Teams integrates with both Trello and Lucidchart, creating a seamless digital workspace for the team.

#### **Documentation Practices**

* All project artifacts, including meeting notes, technical diagrams, testing plans, and design decisions, will be stored in a structured shared folder accessible via MS Teams.
* Version control and naming conventions will be enforced to avoid confusion and maintain traceability.
* The team will follow Agile best practices for documentation by prioritizing clarity and value over excessive detail.

4. **Project Management**

Our team will adopt the Agile Scrum methodology to manage the software development lifecycle of this project. Scrum supports iterative development, continuous feedback, and adaptive planning—making it well-suited for dynamic and collaborative environments. This approach enables us to maintain momentum, quickly address challenges, and adapt to evolving project requirements.

#### **Scrum Framework and Workflow**

* The project will be structured around a series of sprints, each spanning approximately one week.
* At the beginning of each sprint, we will conduct a Sprint Planning Meeting to define deliverables, assign tasks, and set priorities.
* We will hold bi-weekly meetings to review progress, discuss blockers, and ensure alignment on upcoming goals.
* At the end of each sprint, we will conduct a Sprint Review and Retrospective to evaluate the sprint’s outcomes and identify areas for improvement.

#### **Task Management with Trello**

Trello will function as our Kanban board, providing a visual and collaborative space for task tracking and status updates. The board will include key columns:

* *To Do:* Tasks awaiting development
* *Doing:* Tasks currently in progress
* *Done:* Completed and reviewed tasks

Each task card will include:

* A clear title and detailed description
* Due dates and priority labels
* Assigned team members
* Checklists and attachments (if needed)

Trello enables real-time updates and accountability, ensuring that all stakeholders have visibility into the current project status.

#### **Communication and Collaboration via Microsoft Teams**

Microsoft Teams will serve as our central communication hub throughout the project. We will use Teams for:

* Sprint planning and daily discussions
* Video conferencing for team meetings and check-ins
* Document sharing and co-authoring
* Real-time messaging to quickly resolve questions and coordinate tasks

All key project artifacts, decisions, and notes will be documented and archived within Teams channels for easy reference.

**5.** **Requirements Specification**

**5.1.** **Business Requirements**

| Requirement ID | Requirement Description | MOSCOW |
| --- | --- | --- |
| BR1 | Software must use a relational database to store data for books and customers. | M |
| BR2 | Software should allow for late fees and due dates on books. | S |
| BR3 | Software must allow libraries to generate and export a report of books based on status, author, ISBN, and title. | M |

**5.2.** **User Requirements**

| Requirement ID | Requirement Description | MOSCOW |
| --- | --- | --- |
| UR1 | The system must provide a searchable interface that allows users to find books by title, author, or ISBN. | M |
| UR2 | Software should allow users to view any outstanding late fees associated with their account. | S |
| UR3 | Software must store user email and name using industry-standard security practices. | M |

**5.3.** **Functional Requirements**

| Requirement ID | Requirement Description | MOSCOW |
| --- | --- | --- |
| FR1 | The application shall provide user authentication mechanisms to ensure secure access. | M |
| FR2 | The system shall allow library staff to add, edit, and delete book records in the inventory. | M |
| FR3 | The system shall enable users to search for books by title, author, or ISBN and view their availability status. | M |

**5.4.** **Non-Functional Requirements**

| Requirement ID | Requirement Description | MOSCOW |
| --- | --- | --- |
| NFR1 | The application shall provide a user-friendly interface that allows users to complete tasks without prior training. | M |
| NFR2 | The application shall store data securely, ensuring that only authorized users can modify inventory records. | M |

**5.5.** **Implementation (Performance) Requirements (Optional)**

| Requirement ID | Requirement Description | MOSCOW |
| --- | --- | --- |
| IR1 | The development environment shall be set up using Visual Studio Code for code development. | S |
| IR2 | The UI design shall be created using Figma for wireframing and prototyping. | M |
| IR3 | The application shall use Python with Tkinter for building the graphical user interface (GUI). | M |

**6.** **High-level Design**

**6.1.** **Security**

**Requirement ID:** SEC1  
**Description:** Secure Login and Role-Based Access Control  
**Requirement:**The system shall implement a secure login mechanism using username and password credentials. Passwords will be hashed using a secure algorithm before being stored in the database. Upon successful authentication, the system will generate a session token or use cookies to manage user sessions securely.

Users shall be assigned a role of either *admin* or *customer*.

* *Admins* will have full access to book management features, including the ability to add, edit, delete, and mark books as unavailable.
* *Customers* will only be able to search for books and view availability.

Access to certain routes will be restricted based on the authenticated user's role, using middleware or decorators to enforce authorization rules.

#### **6.1.2 Specification 2: User and Role Management**

**Requirement ID:** SEC2  
**Description:** Role Assignment and Permissions Enforcement  
**Requirement:**The system shall implement a role-based user management system, assigning new users the default role of *customer*. Only *admin* users will have the ability to alter user accounts.

The application will include safeguards to prevent privilege escalation attacks (e.g., a regular user attempting to access admin-only endpoints). Unauthorized access attempts will be logged and monitored for security review.

Each role will be associated with a set of permissions, and the system will enforce access control at both the UI and backend levels to ensure consistent security enforcement.

**6.2.** **Hardware**

**Requirement ID:**HW1

**Description:**Server Processor

**Requirement:**A dedicated or cloud based server with a large memory capacity to host the database and application backend efficiently.

* The server must support high availability and scalability to handle multiple user requests.
* The server should include security features like encryption and access control.
* The system should be compatible with Linux/Windows Server OS for hosting purposes.

**Requirement ID:** HW2

**Description:**Networking Infrastructure

**Requirement:** A reliable LAN/Wi-Fi network with firewall security to ensure smooth communication between client devices and the server.

* Network switches and routers must support VLANs for secure traffic segmentation.
* Should allow for remote access via VPN for administrators.
* Must have redundancy to ensure uptime.

**Requirement ID:** HW3

**Description:** Workstations or User Terminals

**Requirement:** Library staff and patrons need computers or tablets with at least a stable internet connectivity to access the system.

* Must support web browsers like Chrome, Firefox, Edge for web-based access.
* Should include barcode scanner support for book check-in/check-out.
* Devices must comply with security policies, including regular software updates.

**6.3.** **User Experience**

### **User Experience 1: Librarian Adding a Book**

**Requirement ID:** UX1  
**Description:** Adding a new book to the inventory.  
**Requirement:**

* The librarian logs in using their credentials.
* They navigate to the "Books" section from the dashboard’s navigation bar.
* They click the "Add Book" button.
* A form appears with input fields for ISBN, Title, and Author.
* The librarian fills in the details and clicks the "Save" button.
* In the settings nav bar librarian can update book status , change password and check out book requests
* Librarian can mark book available and unavailable from the update book status page
* Librarian can check out records of reserved/unavailable or returned/available books

#### **Specifications**

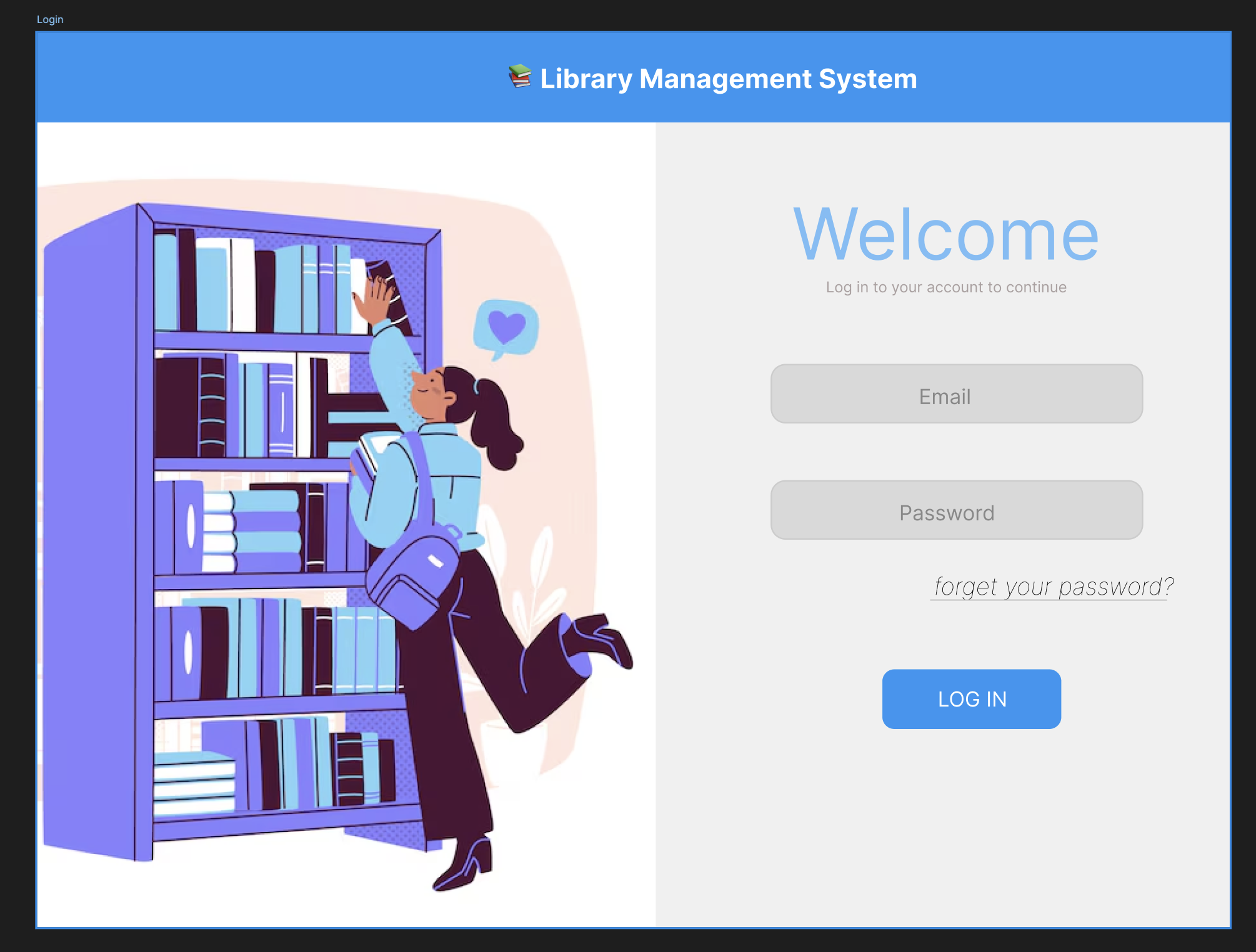
**6.3.1. Functional:**

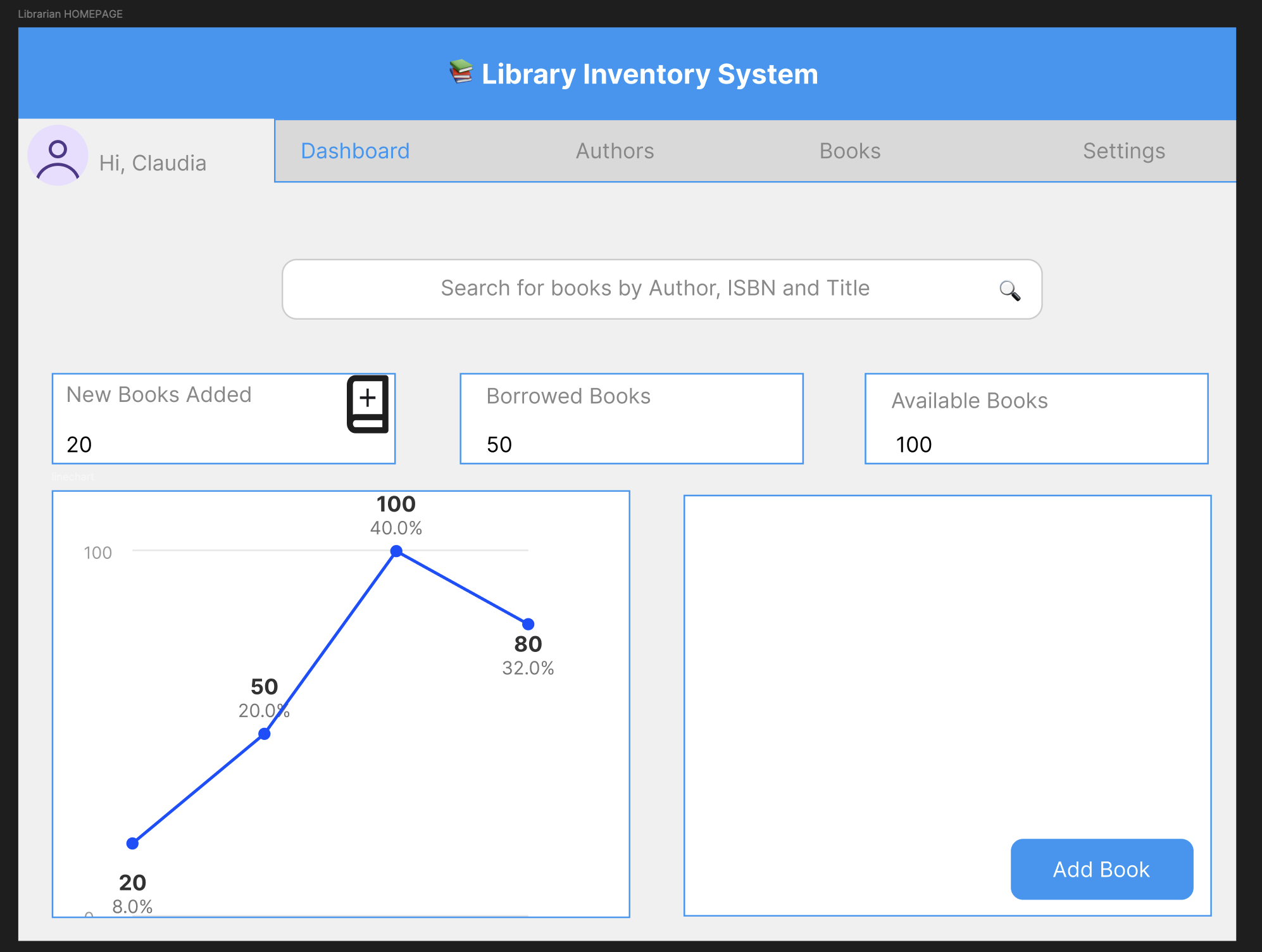
* Form validation ensures no fields are empty.
* ISBN is checked for uniqueness.

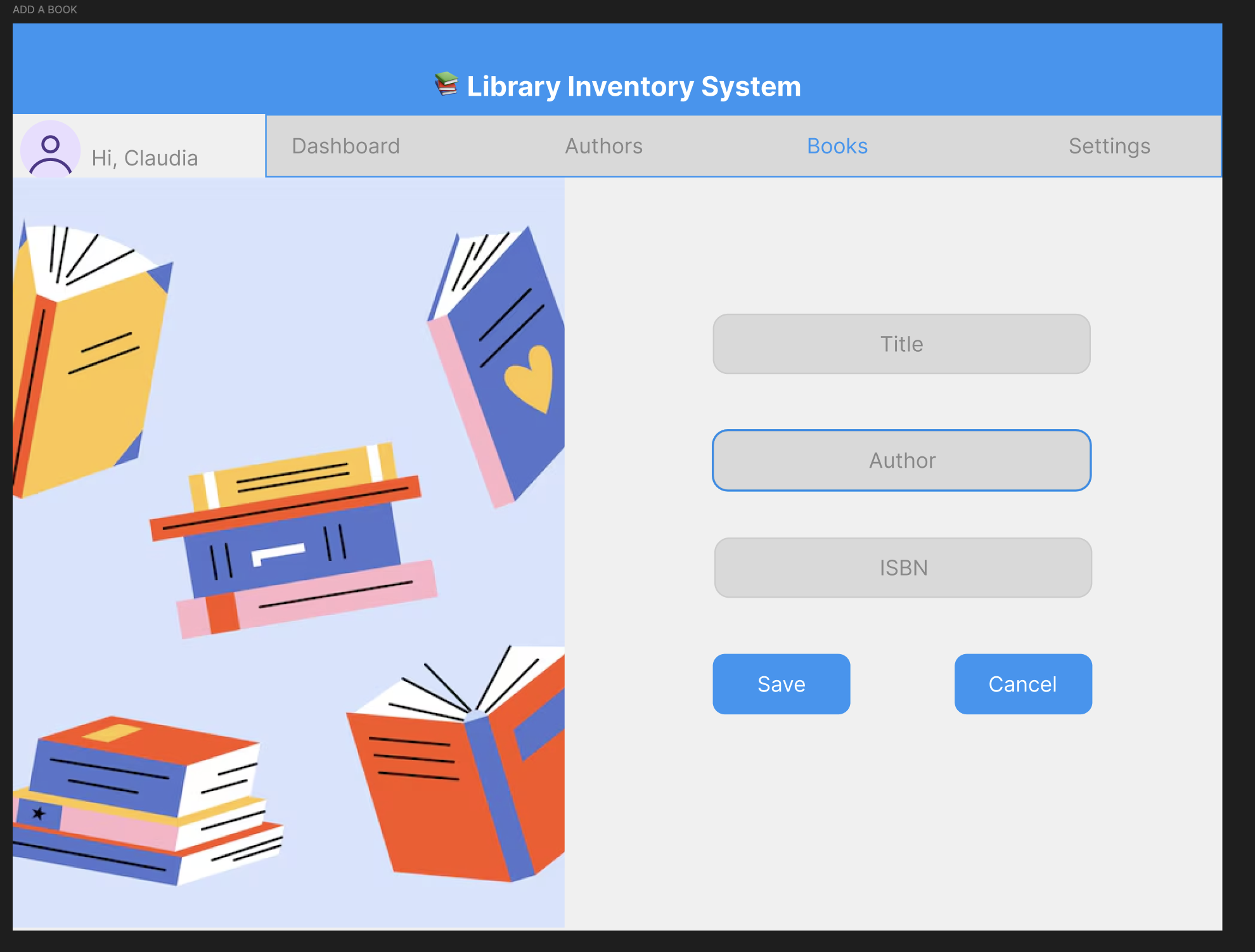
**6.3.2. Non-Functional:**

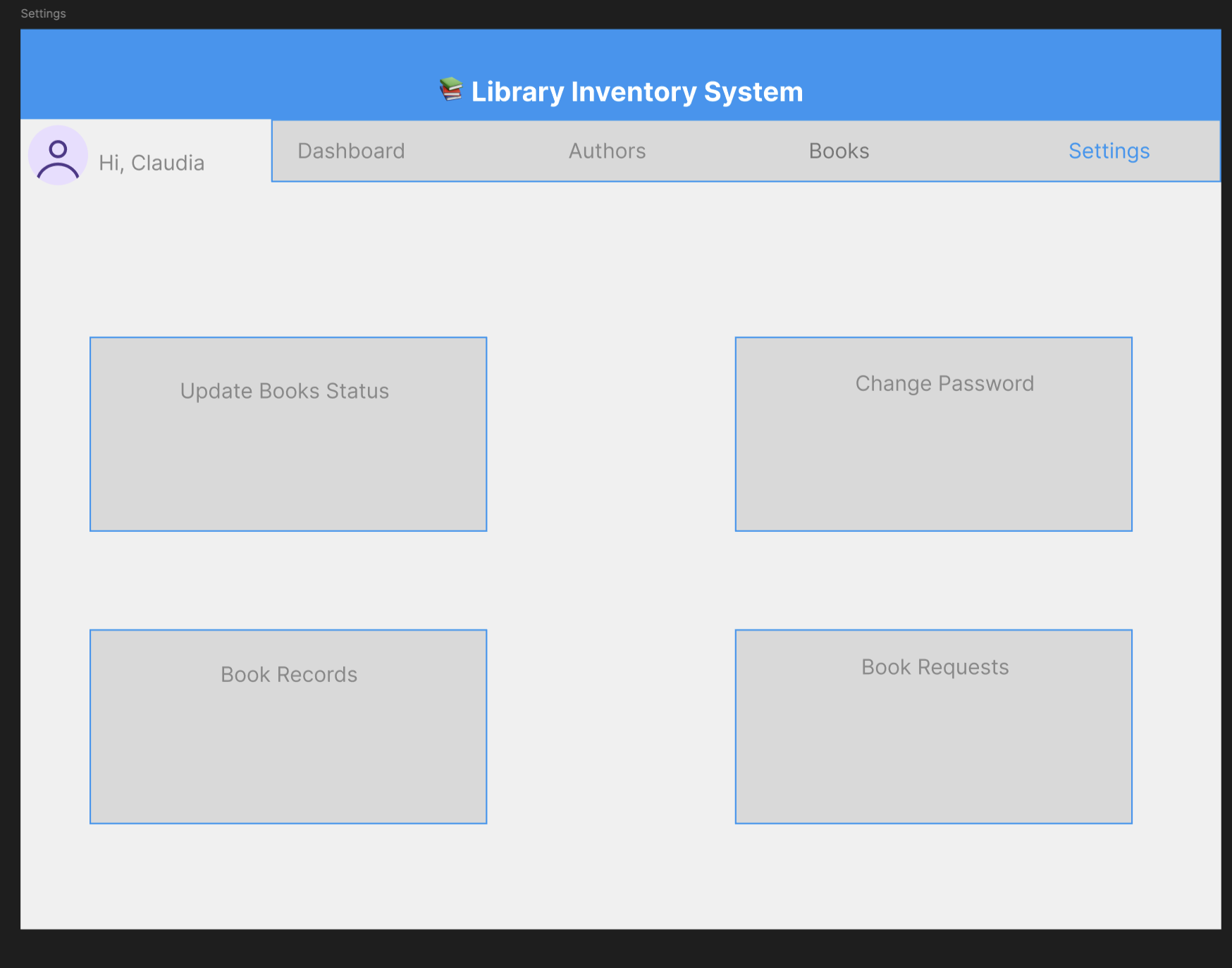
* User-friendly design with clear labels and placeholders.
* Quick response time when submitting.

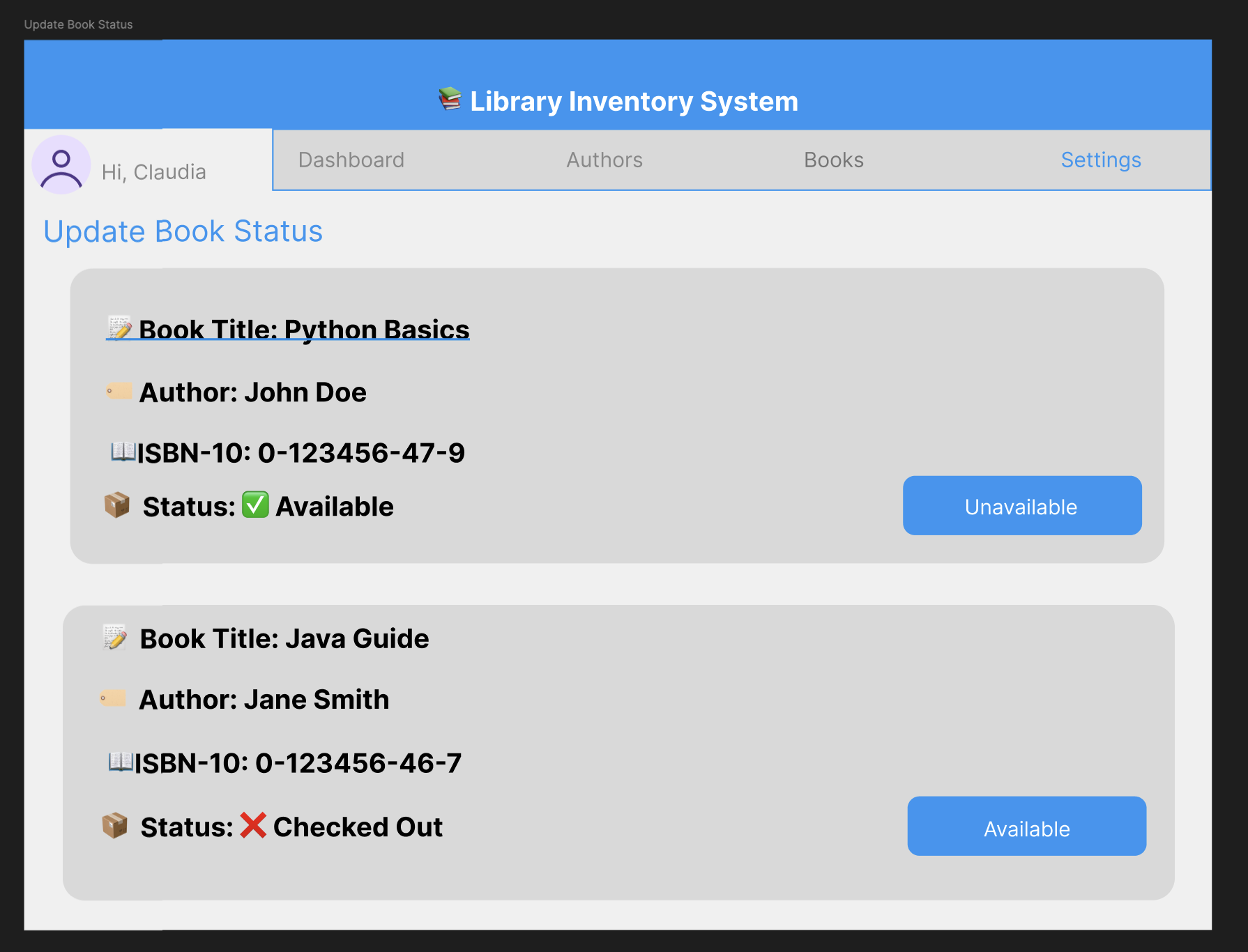
**6.3.3. User Interface Wireframe**

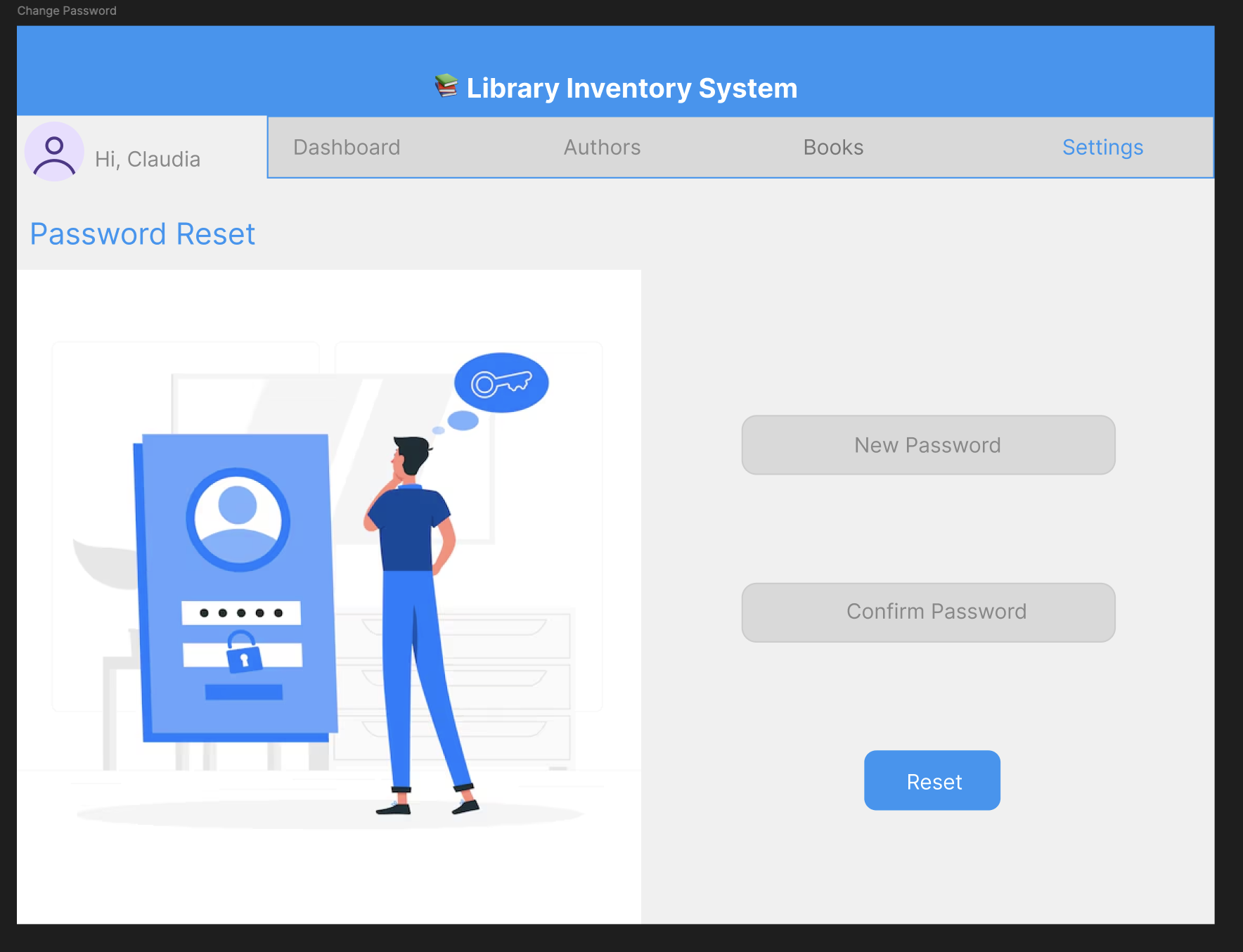
****

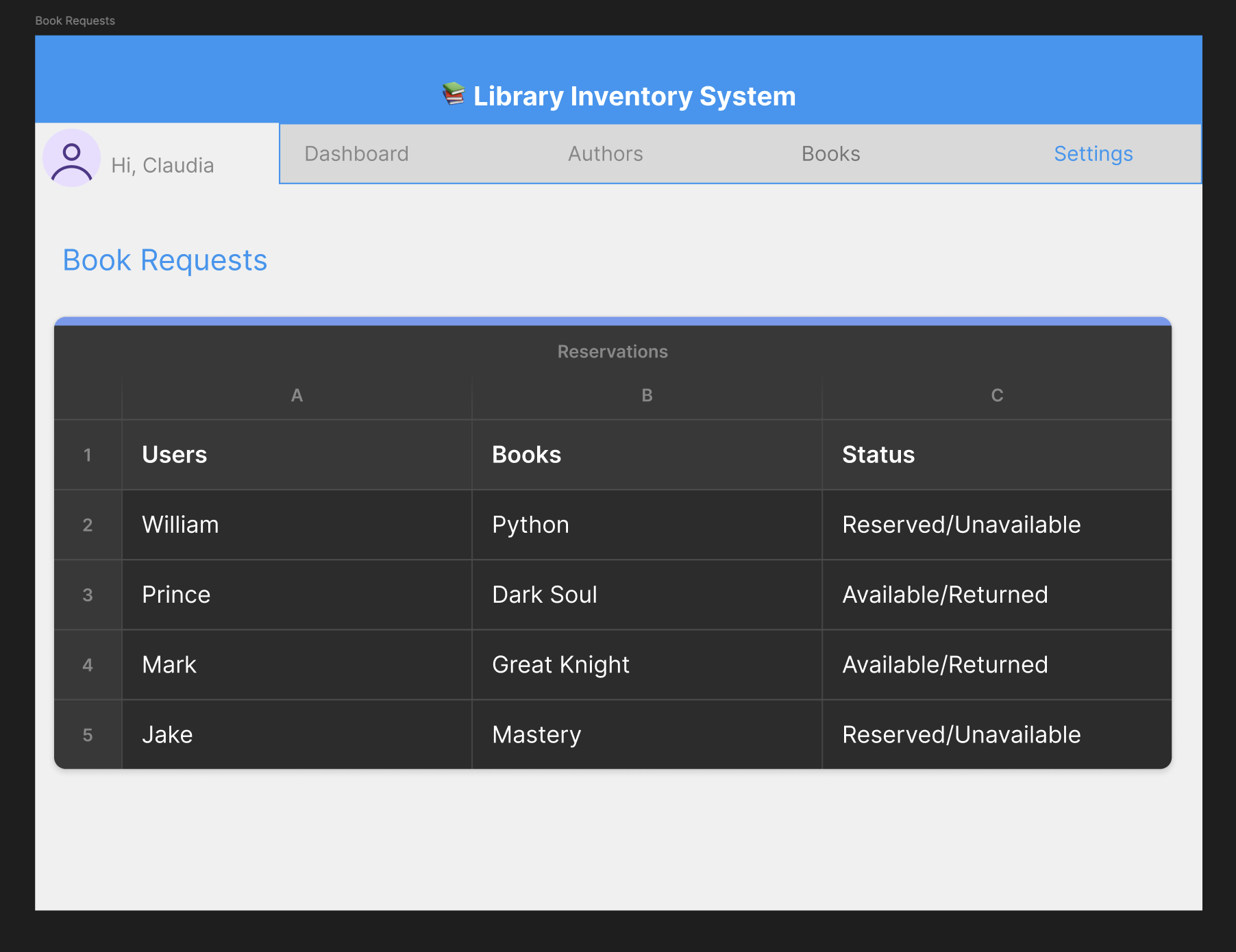
****

****

****

****

****

****

### **User Experience 2: Reader Reserving a Book**

**Requirement ID: UX2  
Description: A reader reserves a book.**

**Requirements:**

* The reader logs in using their credentials.
* They navigate to the "Books" section.
* A list of available books is displayed in a table.
* The reader clicks the "Reserve" button next to a book.
* A confirmation message appears, and the book is marked as "Reserved." on status.

#### **Specifications**

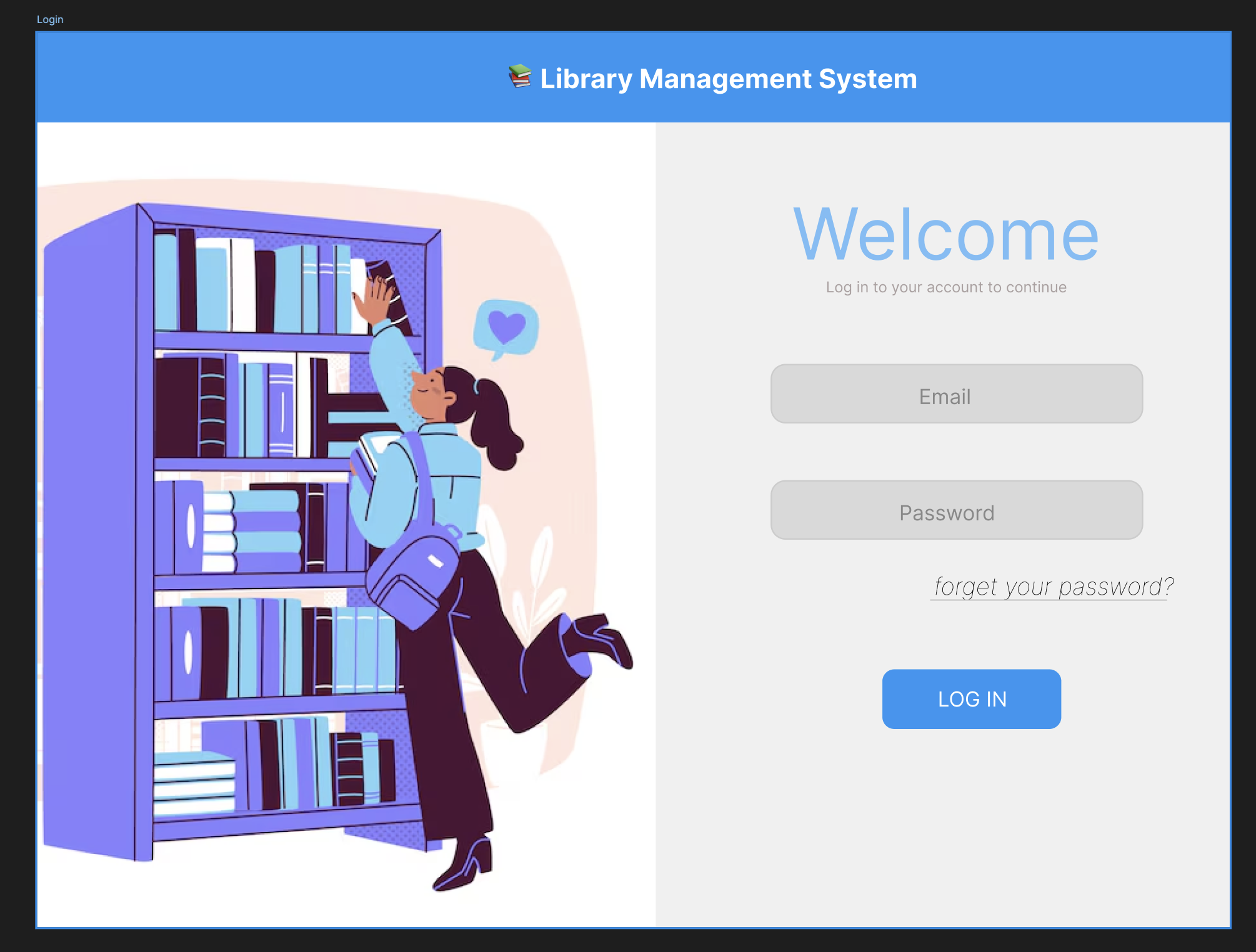
**6.3.2. Functional:**

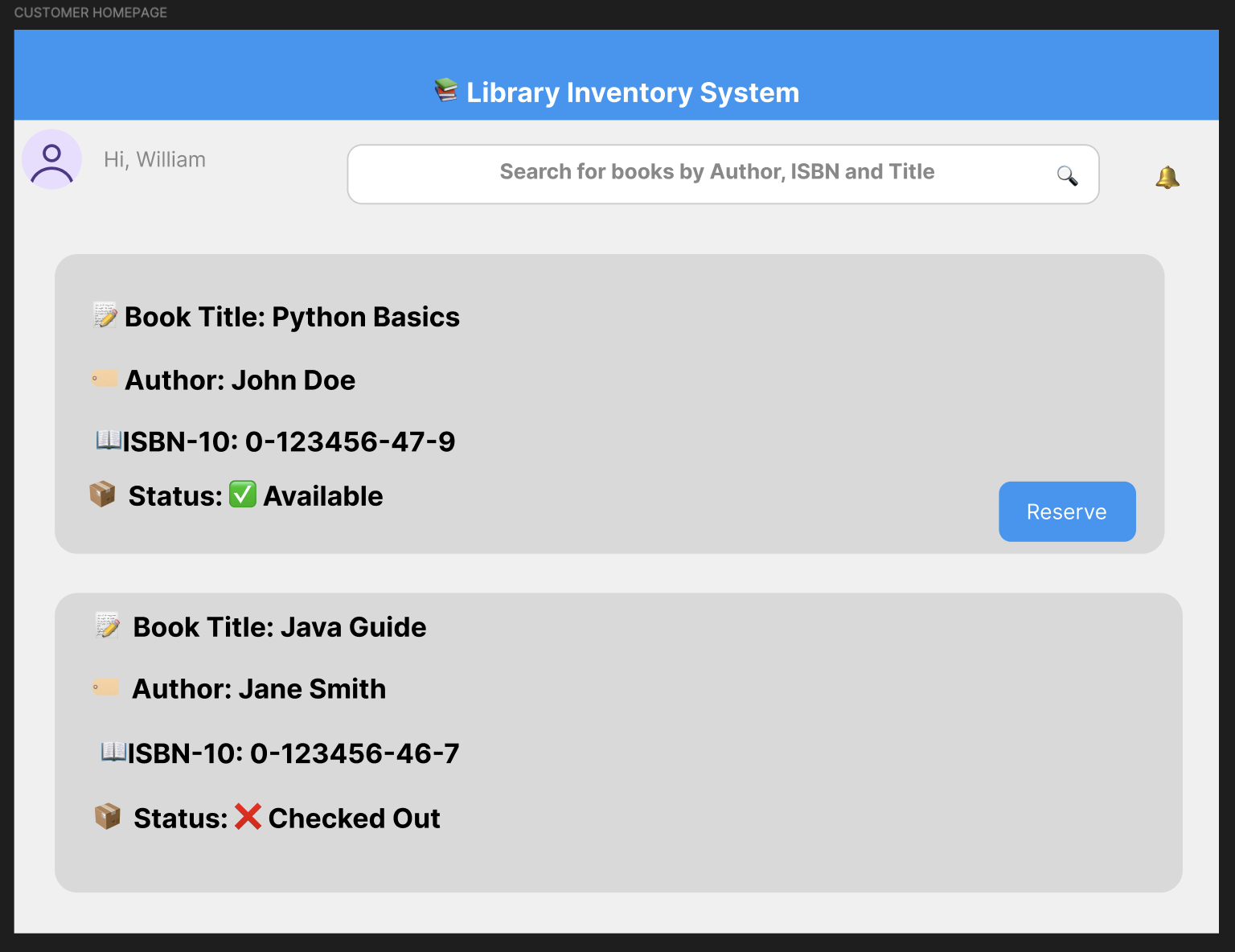
* The system verifies book availability before confirming the reservation.
* The reader can view their reserved books under "My Reservations."

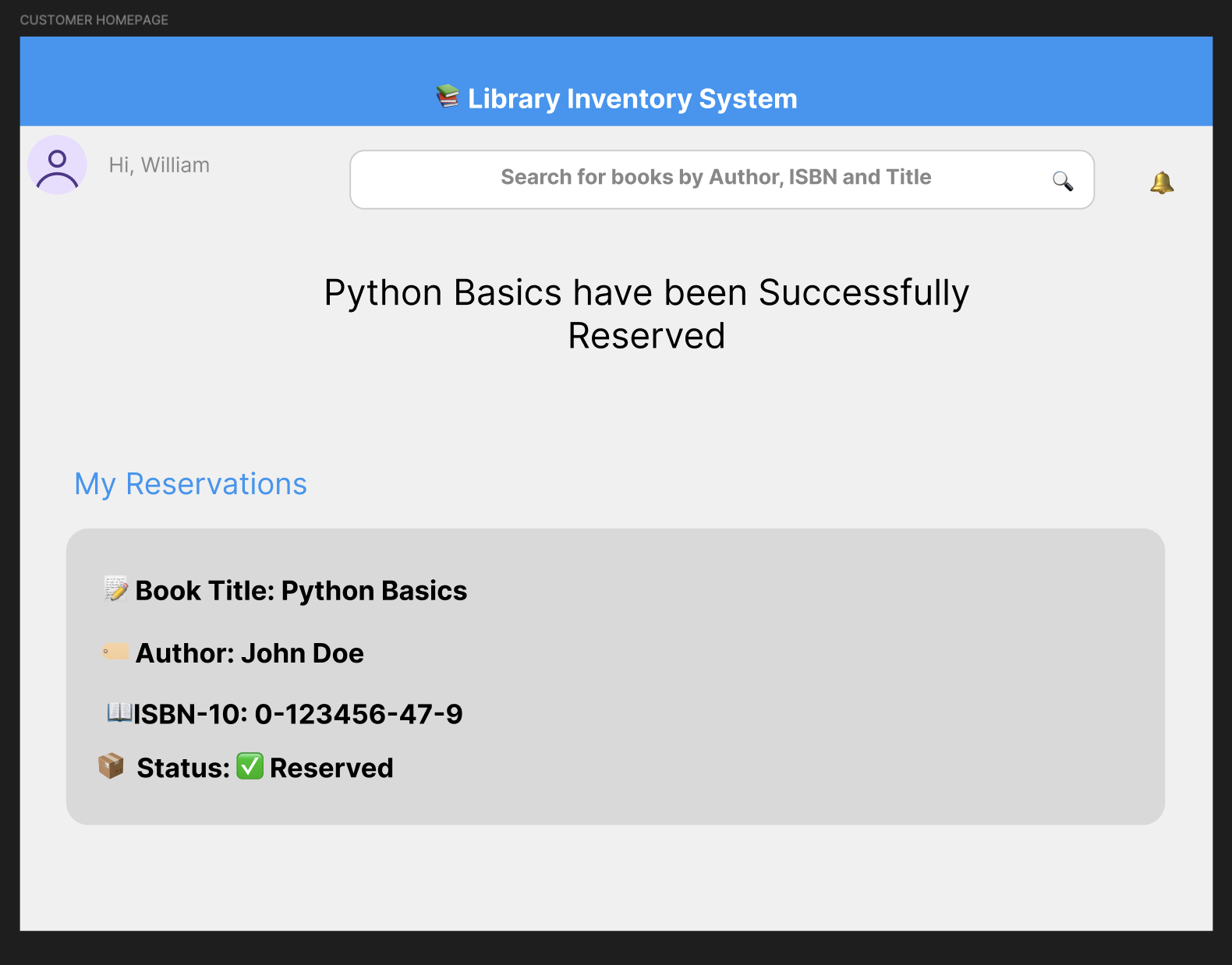
**6.3.3. Non-Functional:**

* Fast response when reserving a book.
* Clear status updates on book reservations.

**1.3.3. User Interface WireFrame**







**6.4.** **Architecture**

The software system follows a three-tier architecture to ensure modularity, scalability, and maintainability. The layers include:

1. **Presentation Layer** (Frontend)
2. **Business Logic Layer** (Backend)
3. **Data Layer** (Database)

Each layer is responsible for specific operations and interacts with the others to ensure a smooth flow of data.

*Requirement ID*: ARCH1

*Description*: Presentation Layer (Frontend)

*Requirements*: Handles user interaction, input validation, and UI rendering.

**Technology Stack**:

* **HTML, CSS, JavaScript** for web interface
* **Bootstrap** or **Tailwind CSS** for responsive design
* **React.js** or **Vue.js** for dynamic UI components

*Interaction*: Sends user requests (e.g., search queries, book management actions) to the backend via API calls.

*Requirement ID*: ARCH3

*Description*: Business Logic Layer (Backend)

*Requirements*: Processes requests, enforces business rules, and interacts with the database.

**Technology Stack**:

* Python (Flask/Django) or Java (Spring Boot) for API and business logic
* RESTful API for communication between frontend and backend

*Interaction*: Receives API requests from the frontend, processes them, and queries the database.

*Requirement ID*: ARCH4

*Description*: Data Layer (Database)

*Requirement*: Stores, retrieves, and manages application data.

**Technology Stack**:

* PostgreSQL/MySQL for relational database management
* SQLAlchemy (if using Python) or Hibernate (if using Java) for ORM

*Interaction*: Backend queries the database to fetch, update, or delete records.

*Requirement ID*: ARCH5

*Description*: Technologies Used

*Requirement*: The following technologies will be utilized for development:

**Programming Languages**:

* Python
* JavaScript (Frontend)
* SQL

**Frameworks/Libraries:**

* Frontend: React.js, Bootstrap, Tailwind CSS
* Backend: Django/Flask (Python), Spring Boot (Java)
* Database: PostgreSQL, MySQL

**APIs**:

* ISBN Lookup API for book metadata
* External Authentication API (if needed)
* Version Control: GitHub for repository management and team collaboration

*Requirement ID*: ARCH6

*Description*: Constraints & Considerations

*Requirement*: The system must adhere to the following constraints and considerations:

**Scalability**: Must handle increasing users and data volumes efficiently.

**Security**: Implement authentication (OAuth, JWT), data encryption, and role-based access.

**Performance**: Optimize database queries and use caching mechanisms.

**Cross-Platform Support**: Ensure compatibility with Windows, macOS, and Linux.

**Maintainability**: Modular design to facilitate future enhancements.

**6.5.** **Database**

The Library Inventory System will use an industry standard RDBMS(Relational Database Management System) called MySQL. MySQL is a free open source RDBMS which makes it extremely cost effective and has scalability in mind which will help build out the Library Inventory System to support more ideas in the future.

***Requirement ID:*** DB1

***Description:*** Library Inventory System Database

***Requirement:*** The Library Inventory System will use a relational database structure. A relational database will allow the system to connect books with authors and statuses. This type of structure will also allow libraries to easily pull and export reports of books and customers based on provided queries.

**6.5.1**. **Specifications**

### **Database Management System (DBMS): MySQL**

#### **Data Types:**

* **Categories:** cat\_id (Number), genre (VarChar)
* **Book\_Authors:** author\_id (Number), book\_id (Number)
* **Books:** book\_id (Number), cat\_id (Number), isbn (Number), author (VarChar), title (VarChar), status (VarChar)
* **Authors:** author\_id (Number), fname (VarChar), lname (VarChar)
* **Users:** user\_id (Number), fname (VarChar), lname (VarChar), email (VarChar), role (VarChar)
* **User\_Checkouts:** user\_id (Number), book\_id (Number)

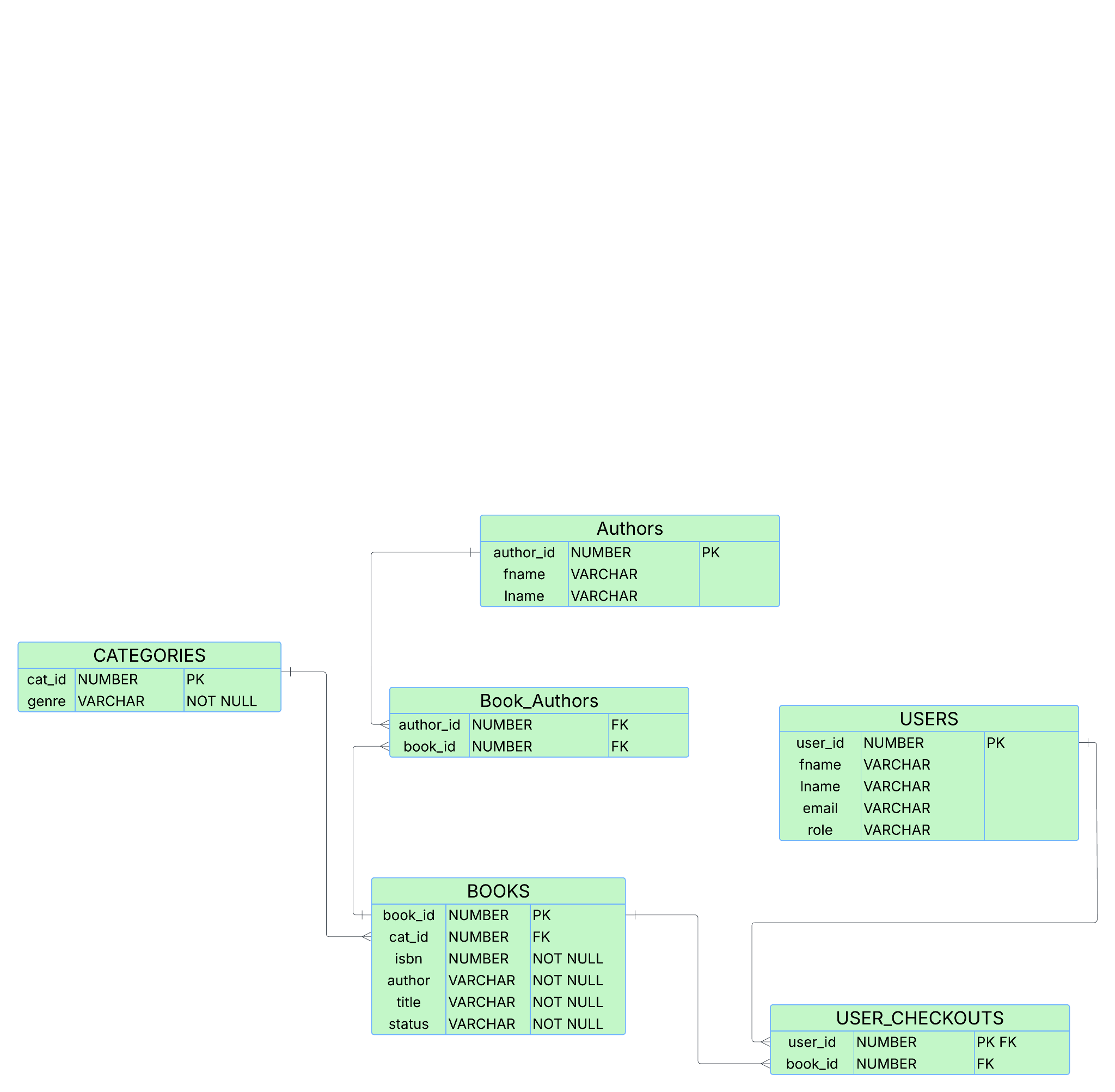
#### **Tables:**

* **Categories:** Stores categories of books.
* **Book\_Authors:** Stores what books are tied to an author.
* **Books:** Stores book information.
* **Authors:** Stores author name.
* **Users:** Stores user information.
* **User\_Checkout:** Stores what books a user has checked out.

**6.5.2.** **Functional Requirements**

* **Registration:** Users can register an account by providing their first and last name, followed with an email.
* **User Checkout:** A book ID will be tied to the User ID that is logged in to track what books that user has checked out.
* **Inventory:** Librarians can track book inventory by querying the Books table to view the amount of books by title, author, and isbn.

**6.5.3. Entity-Relationship Diagram**



**6.6.** **Top-level Classes**

These are the top-level classes and the components each class is designed to implement. Each entry includes:

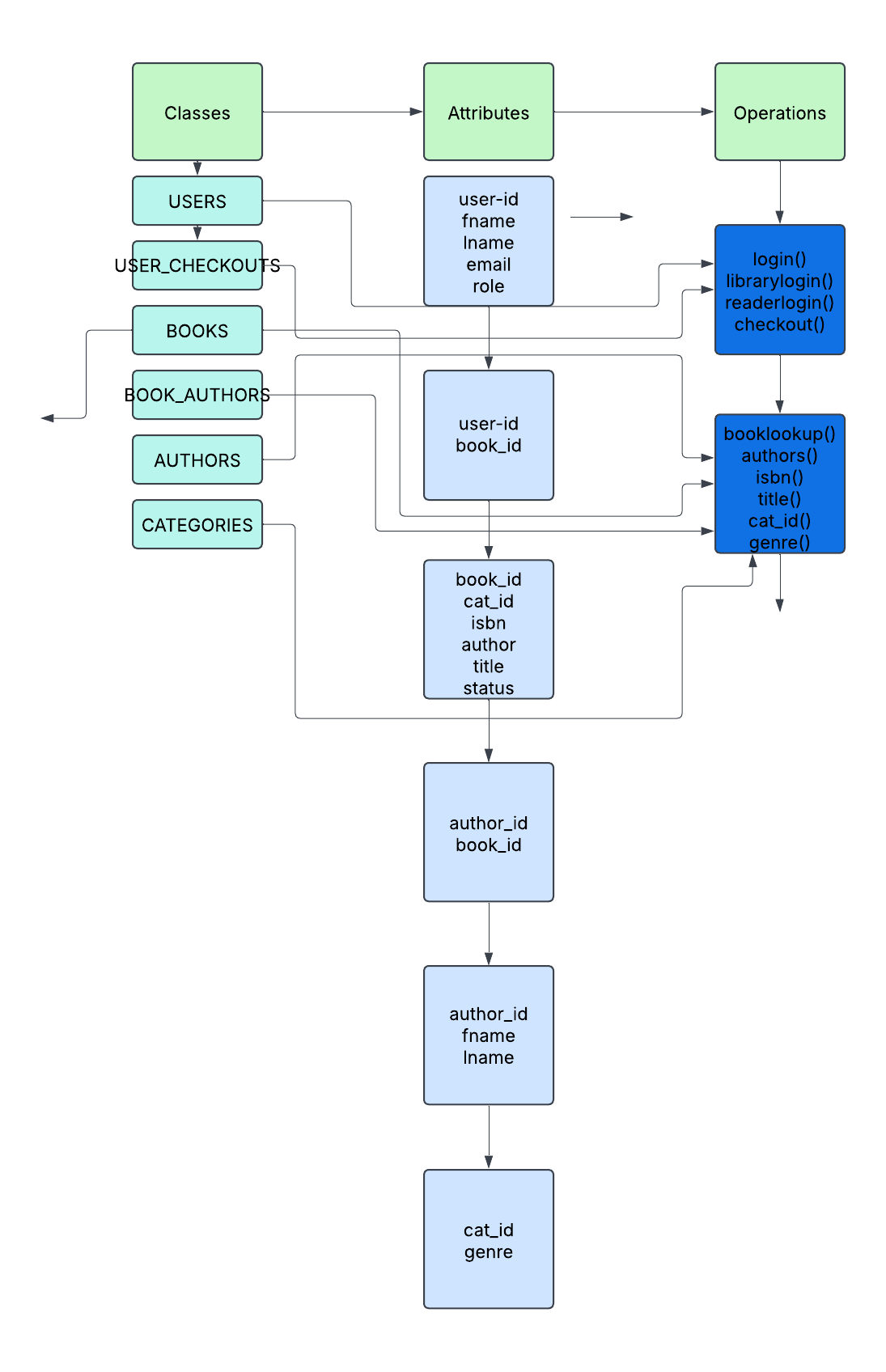
* A **Requirement ID** for easy reference
* A **Description** that defines the purpose of the class
* A **Requirement** that explains what the class should do during operation

This structure improves clarity and helps track the system's functionality through object-oriented design. The class diagram included in the next section offers a visual representation of these top-level class designs.

#### **6.6.2. Class Requirements**

* **Requirement ID:** USER1  
   **Description:** Users  
   **Requirement:** Logs the user in
* **Requirement ID:** USERCHKOUT2  
   **Description:** User\_Checkout  
   **Requirement:** Checks out the user
* **Requirement ID:** BOOKS3  
   **Description:** Books  
   **Requirement:** Looks up book for user
* **Requirement ID:** BOOKAUTH4  
   **Description:** Book\_Author  
   **Requirement:** Looks up book through author
* **Requirement ID:** AUTH5  
   **Description:** Authors  
   **Requirement:** Shows list of authors
* **Requirement ID:** CAT6  
   **Description:** Categories  
   **Requirement:** Gives list of books and availability

**6.6.3. Class Diagram**



**6.7.** **Data Flow and States**

### *Requirement ID: DF1*

*Description*: Users can log in and perform actions based on their roles (Admin or Customer). admins can manage the inventory (add/edit/remove books), while customers can search and borrow books.  
 *Requirement*: The system shall allow authenticated users to perform role-specific actions, with data transitions captured and processed in real-time, reflecting state changes (e.g., book availability, user sessions).

### **6.7.1. Specifications**

* *Login Flow:*
  + User enters credentials.
  + System authenticates the user and starts the session.
  + System checks role (Admin or Customer) and redirects to appropriate dashboard.
* *Admin Book Management Flow:*
  + Admin adds a book → Book is added to the inventory with status “Available”.
  + Admin edits a book → Inventory record is updated in the database.
  + Admin deletes a book → Status is changed to “Removed” or the entry is deleted.
* *Customer Book Interaction Flow:*
  + Customer searches for a book → System filters inventory and returns matching results.
  + Customer borrows a book → Book status changes from “Available” to “Unavailable”.
  + Customer returns a book → Book status changes from “Unavailable” to “Available”.
* *State Changes:*
  + Book states include: Available, Unavailable, Removed.
  + User session states include: Logged Out, Logged In, Session Expired.

**6.7.2** **State Transitions**

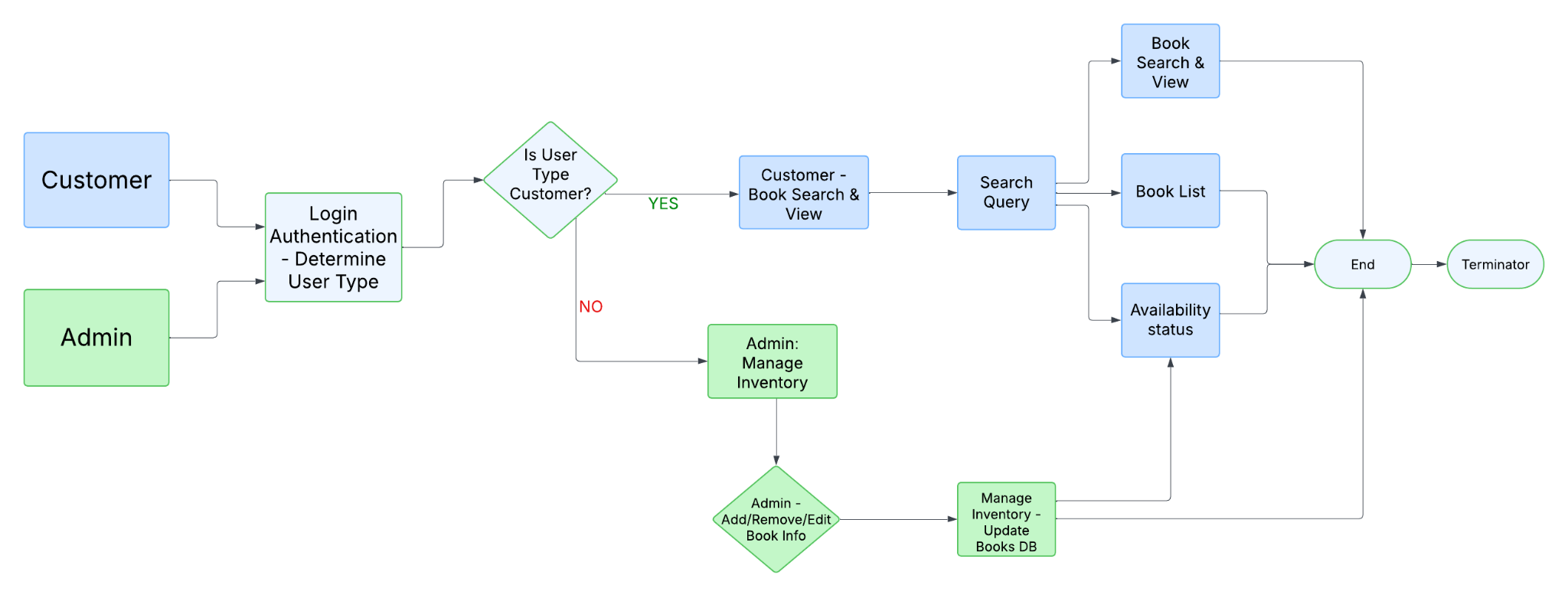
How key entities in the system transition between states based on actions and system events.

**User Session State Transitions:**

| **Current State** | **Action/Event** | **Next State** |
| --- | --- | --- |
| Logged out | User logs in successfully | Logged in |
| Logged in | User logs out | Logged out |
| Logged in | Inactivity Timeout | Logged out |
| Session expired | User logs in again | Logged in |

**Book State Transitions:**

| **Current State** | **Action/Event** | **Next State** |
| --- | --- | --- |
| N/A | Admin adds a book | Available |
| Available | Admin marks book as borrowed | Unavailable |
| Unavailable | Admin marks book as returned | Available |
| Available/Unavailable | Admin deletes book | Removed |

**6.7.3 Data Flow Diagram**

**6.8.** **Reports**

#### **6.8.1. Specification**

The reports listed below are examples of outputs generated by the system. Each report serves a specific purpose and helps users—both librarians and readers—interact with data efficiently. The unique IDs, descriptions, and requirements help define the scope and function of each report.

#### **6.8.2. Report Requirements**

* **Requirement ID:** REPORT1  
   **Description:** Books Report  
   **Requirement:** Displays a list of books that a user has searched for using keywords or filters.
* **Requirement ID:** REPORT2  
   **Description:** Title Report  
   **Requirement:** Displays a list of books that match a specific title input by the user.
* **Requirement ID:** REPORT3  
   **Description:** Author Report  
   **Requirement:** Displays a list of books associated with an author specified by the user.
* **Requirement ID:** REPORT4  
   **Description:** ISBN Report  
   **Requirement:** Displays a list of books that match a specific ISBN number input.
* **Requirement ID:** REPORT5  
   **Description:** Reserve Report  
   **Requirement:** Shows all books that have been reserved by users and their current status.
* **Requirement ID:** REPORT6  
   **Description:** Sales Report  
   **Requirement:** Generates a report showing sales or transaction activity involving books.

**7. Appendices**

### **7.1. Glossary**

| **Term** | **Definition** |
| --- | --- |
| **Admin** | A user role with permissions to manage book inventory, user accounts, and system configurations. |
| **Customer** | A user role with permissions to view, reserve, and borrow books. |
| **ISBN** | International Standard Book Number, a unique identifier for books. |
| **RDBMS** | Relational Database Management System, used to store and manage structured data. |
| **UI** | User Interface, the visual component through which users interact with the system. |
| **UX** | User Experience, the overall experience of a person using the system. |
| **ORM** | Object-Relational Mapping, a technique for converting data between incompatible systems. |
| **API** | Application Programming Interface, used for communication between software layers. |
| **CRUD** | Create, Read, Update, Delete – basic operations for data management. |
| **Wireframe** | A basic layout design of user interface elements before full implementation. |

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### **7.2. Acronyms**

| **Acronym** | **Full Form** |
| --- | --- |
| **UI** | User Interface |
| **UX** | User Experience |
| **API** | Application Programming Interface |
| **RDBMS** | Relational Database Management System |
| **ERD** | Entity-Relationship Diagram |
| **JWT** | JSON Web Token |
| **HTML** | HyperText Markup Language |
| **CSS** | Cascading Style Sheets |
| **SQL** | Structured Query Language |

### 

### 

### 

### 

### 

### **7.4. Supporting Diagrams**

* User Interface Wireframes (Section 6.3.3)
* Class Diagram (Section 6.6.3)
* Entity-Relationship Diagram (Section 6.5.3)
* Data Flow Diagram (Section 6.7.3)
* State Transition Tables (Section 6.7.2)