**Software Requirement Specifications**

**Online Library Management System**



**Submitted by**

Warisha Habib

F22BINFT1M01003

**Submitted to**

Syed Ali Nawaz Shah

**Department of Information Technology**

**Faculty of Computing**

**The Islamia University of Bahawalpur**

**Meeting Details**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No** | **Details** | **Date** | **Supervisor Signature** |
| SR01 | Discussion about project scope, objectives, system modules, and required functionalities. | 20-10-25 |  |
| SR02 | Reviewed functional and non-functional requirements, clarified use cases, and confirmed system flow. | 27-10-25 |  |
| SR03 | Discussed UML diagrams, system architecture, and necessary improvements for the final SRS. | 17-11-25 |  |
| Sr04 | |  | | --- | | Design review: UML diagrams, system architecture, and database schema. |  |  | | --- | |  | | 24-11-25 |  |
| Sr05 | |  | | --- | | Final review and approval of SRS draft before submission. |  |  | | --- | |  | | 1-12-25 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Summary**

The **Library Management System** automates key library operations, including book cataloging, member management, and book issuance/returns. Admins can efficiently manage books and members, while users can search for and borrow books with real-time availability updates. The system features role-based access, secure data storage in MySQL, and a responsive web interface built with HTML, CSS, and Flask. It reduces manual errors, improves efficiency, and provides a user-friendly library experience

**Table of Contents**

[**1.** **Introduction** 5](#_Toc182254506)

[**1.1.** **Purpose** 5](#_Toc182254507)

[**1.2.** **Scope** 5](#_Toc182254508)

[**1.3.** **Product Perspective** 6](#_Toc182254509)

[**1.4.** **User Characteristics** 6](#_Toc182254510)

[**1.5.** **Similar apps and systems/Literature Review** 6](#_Toc182254511)

[**1.6.** **Proposed Technologies** 7](#_Toc182254512)

[**2.** **Requirements** 7](#_Toc182254513)

[**2.1.** **Function Requirements** 7](#_Toc182254514)

[**2.1.1.** **Sign Up** 7](#_Toc182254515)

[**2.2.** **Non-Functional Requirements** 10](#_Toc182254516)

[**3.** **Use Cases and Flow of Processes** 10](#_Toc182254517)

[**3.1.** **Use Case 1** 11](#_Toc182254518)

[**4.** **References** 17](#_Toc182254519)

1. **Introduction**

The Library Management System is being developed to support the needs of a typical academic or institutional library. Libraries traditionally manage book records, member registrations, and issue/return transactions using manual methods, which can lead to inefficiencies, data inconsistencies, and difficulty in tracking book availability. As libraries expand their collections and user base, manual record-keeping becomes increasingly time-consuming and prone to errors.

To address these challenges, this project focuses on developing the Book, Member, and Transaction Management Module, which automates cataloging, membership handling, and book circulation processes. The system aims to provide library staff with efficient administrative tools and offer library members an intuitive platform to search for books, check availability, and borrow items easily.

* 1. **Purpose**

The purpose of this project is to design and implement a secure and efficient web-based system that automates major library operations such as book management, member registration, searching, issuance, and return processes. The project aims to reduce manual workload, prevent errors, and provide real-time updates on book availability.

* 1. **Scope**

**In Scope**

* Book catalog management (add, update, delete, search).
* Member account management.
* User authentication with separate admin and member roles.
* Book issuance.
* Availability updates based on real-time transactions.
* Simple and responsive web interface using HTML and CSS.

**Out of Scope**

* Fine calculation for overdue books.
* Digital library integration (PDFs/e-books).
* Inventory management beyond books (e.g., CDs, magazines).
* SMS/email notification system.
* Mobile app version.
  1. **Product Perspective**

The developed module serves as the core of a complete Library Management System. It replaces manual ledger-based processes with automated modules for record storage and transaction tracking. The system interacts directly with the MySQL database for persistent data storage and uses Flask to manage routing, authentication, and communication between the user interface and the database.

* 1. **User Characteristics**

**Admin (Librarian/Library Staff)**

* Moderate computer proficiency.
* Responsible for managing books, members, and all transactions.

**Library Member (Students/Faculty)**

* Basic computer literacy.
* Can search books, view availability, and borrow books through the system.
  1. **Similar apps and systems/Literature Review**
* **KOHA ILS:** A widely used open-source system with extensive modules but complicated installation and heavy resource usage.
* **SLiMS (Senayan Library Management System):** Lightweight and feature-rich but lacks modern UI/UX and customizability.
* **LibraryThing:** Used for cataloging but does not support issue/return workflows for institutional libraries.
  1. **Proposed Technologies**

**Backend:** Flask (Python)

**Database:** MySQL

**Frontend:** HTML, CSS

**Development Tools:** VS Code, GitHub, MySQL Workbench

1. **Requirements**

The Library Management System will automate major library tasks such as managing book records, handling member registration, and processing book issuance and returns. Librarians will be able to add new books and update existing records. Users can log in to search for books, check availability, and borrow them. The system will maintain updated availability statuses and store transaction history securely in the MySQL database. All workflows will be handled through a responsive and user-friendly interface.

* 1. **Function Requirements**

Authors will provide definite number of functional requirements in standard format. These requirements directly in the with functions which are already provided. The requirement format as following.

* + 1. **Sign Up**

**Sign Up**

* **Name:** FR001
* **Purpose:** To allow new members or admin users to register in the system.
* **User(s**): Admin, Library Member
* **Input:**
  + Full Name
  + Email Address
  + Password (minimum 8 characters, must include letters and numbers)
  + Phone Number
  + User Role (Admin / Member)
* **Output:** Successful account creation; user can log in.

**2.1.2 Book Management**

* **Name:** FR002
* **Purpose:** To enable admins to create, update, delete, and view book records.
* **User(s):** Admin
* **Input:**
  + Book Title
  + Author
  + ISBN
  + Category
  + Total Copies
* **Output:** Book details stored or updated in system; availability automatically calculated.

**2.1.3 Member Management**

* **Name:** FR003
* **Purpose:** To manage library member data.
* **User(s):** Admin
* **Input:**
  + Member ID
  + Name
  + Phone Number
  + Email
* **Output:** Member record added, updated, or deleted

**2.1.4 Search Books**

* **Name:** FR004
* **Purpose:** To allow users to search the library catalog.
* **User(s):** Member, Admin
* **Input:**
  + Keywords (title, author, category)
* **Output:** List of matching books with availability status.

**2.1.5 Request Book**

* **Name:** FR008
* **Purpose:** To allow a library member to request a book that is unavailable or requires admin approval.
* **User(s):** Library Member
* **Input:**
  + Book ID
  + Member ID

**2.1.6 Issue Book**

* **Name:** FR005
* **Purpose:** To allow issuing of books to members.
* **User(s):** Admin
* **Input:**
  + Member ID
  + Book ID
  + Issue Date
* **Output:** Book issued; availability reduced by one.

**2.1.7 Return Book**

* **Name:** FR006
* **Purpose:** To record book returns.
* **User(s):** Admin
* **Input:**
  + Book ID
  + Member ID
  + Return Date
* **Output:** Book returned; availability increased; transaction marked complete.
  1. **Non-Functional Requirements**

**Performance:** System should handle at least 200 concurrent requests with response times under 2 seconds.

**Security:**

* Passwords stored using hashing (e.g., bcrypt).
* Role-based access control for admin and members.

**Usability:** Clean, simple UI using HTML and CSS; should be easy to navigate on desktop and mobile.

**Reliability:** MySQL database backups automatically maintained (daily).

**Scalability:** System should support up to 10,000 book records without performance degradation.

**Maintainability:** Code organized using Flask Blueprints for easy updates.

1. **Use Cases and Flow of Processes**

This section describes the core interactions between users and the system.  
It outlines each use case with its corresponding actors, preconditions, and outcomes.  
The flow of processes explains how the system responds to user actions step by step.  
Both normal operations and alternative paths are documented for clarity.  
These use cases ensure that all system functionalities are defined and understood.

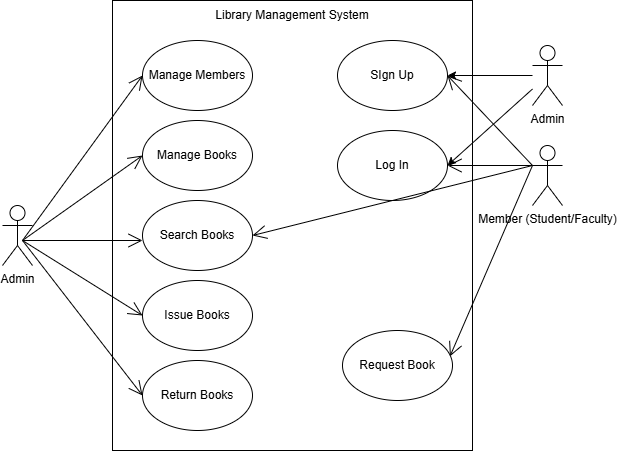
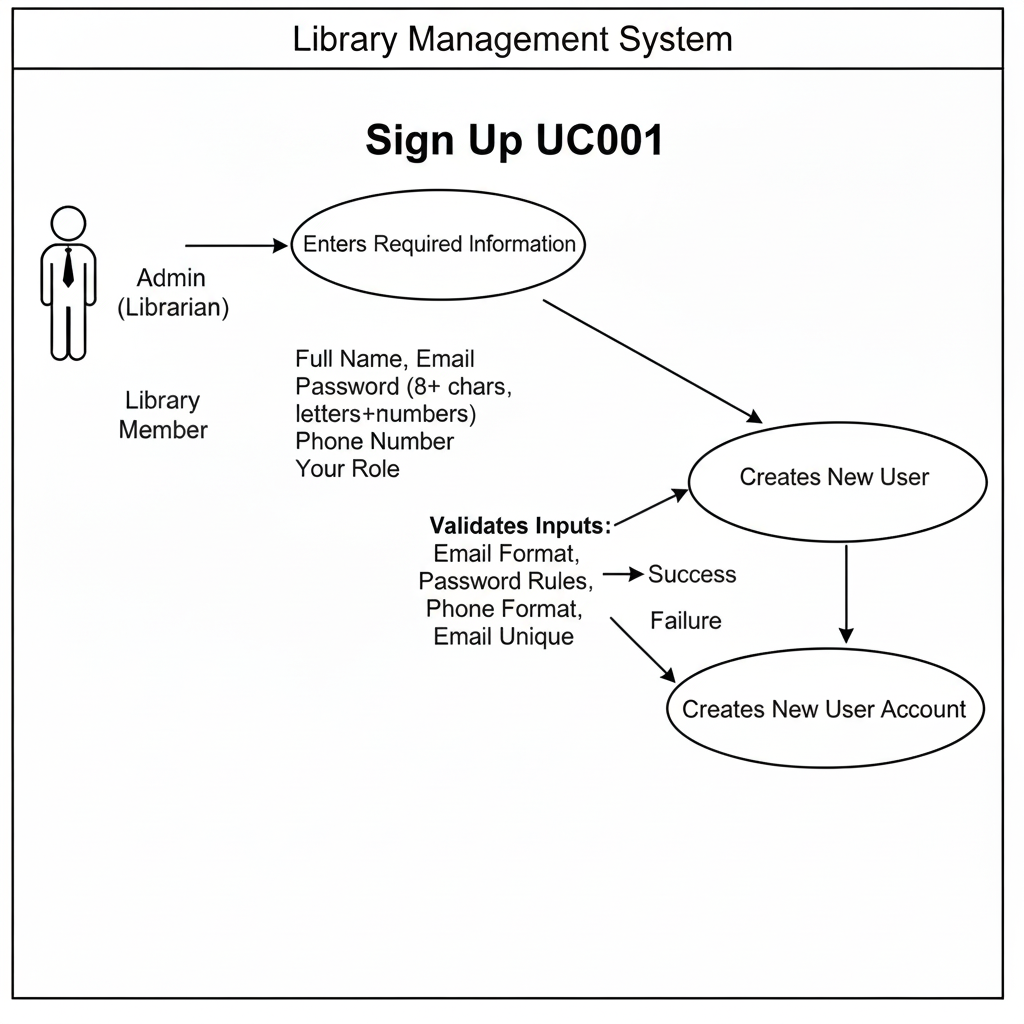


Figure 1: System Level Use Case Diagram

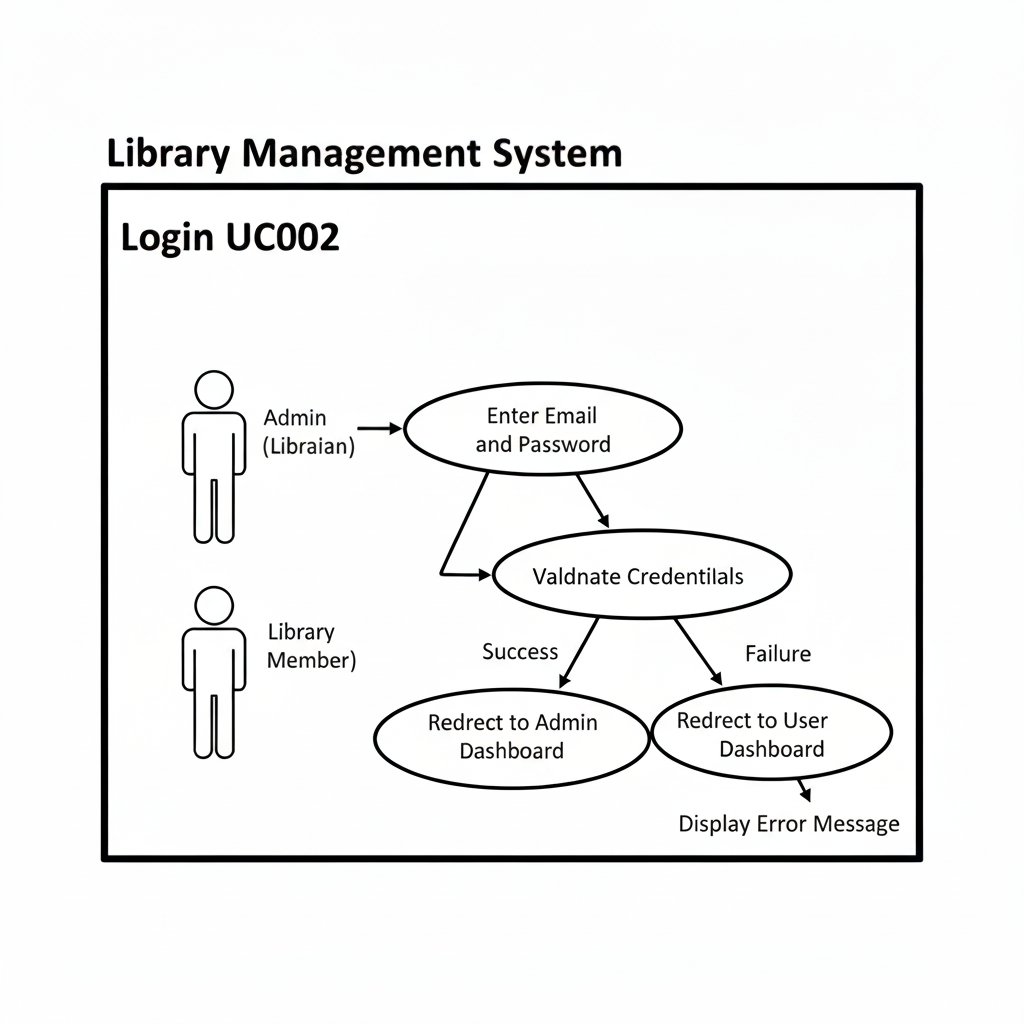
* 1. **Use Case 1: Sign Up**

|  |  |
| --- | --- |
| **ID** | UC001 |
| **Name** | Sign Up |
| **Description** | This use case describes the process of registering a user. |
| **Requirement(s)** | FR001 |
| **Actor(s)** | Admin, Library Member |
| **Precondition** | The actor must not already have an existing account in the system. |
| **Postcondition** | A new user account is created successfully; user can proceed to login. |
| **Basic Flow** | ***Basic Flow***   1. Actor opens the Sign Up page. 2. Actor enters required information.    1. Following fields are required       1. Full Name       2. Email       3. Password       4. Phone Number       5. Your Role    2. Check the proper inputs for following       1. Checks if the email format is valid.       2. Ensures the password meets system rules (minimum 8 characters, letters + numbers).       3. Checks that the phone number is properly formatted.    3. On successful validation the system will create new user   System displays confirmation message.  ***Alternative Flow***   1. Admin Logs In 2. Admin manually creates a new user account by entering their details.   ***Exceptions***   1. Invalid input formats (email, password, phone). 2. Email already registered. 3. System error while creating the account. |

****

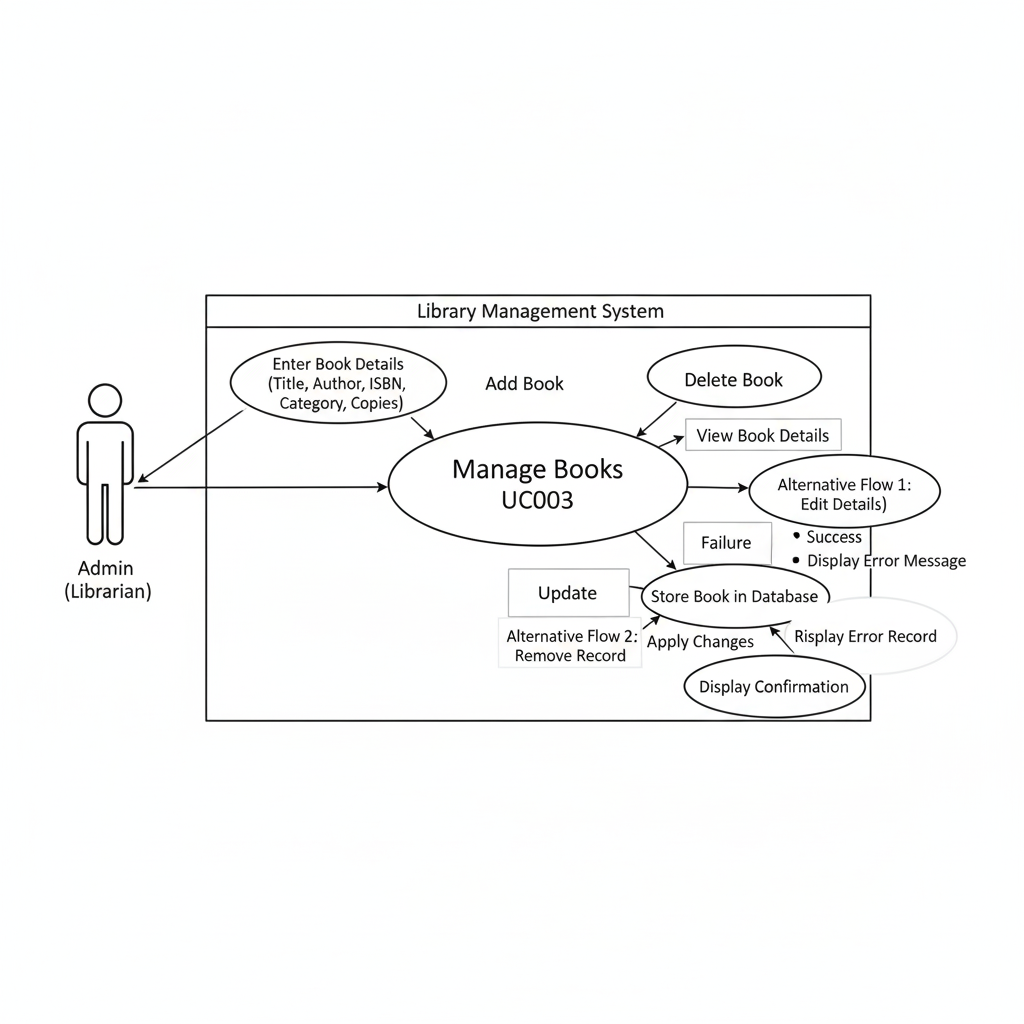
* 1. **Use Case 2: Login**

|  |  |
| --- | --- |
| **ID** | UC002 |
| **Name** | Login |
| **Description** | This use case describes the process of user authentication. |
| **Requirement(s)** | System Function |
| **Actor(s)** | Admin, Library Member |
| **Precondition** | User must have a registered account. |
| **Postcondition** | User is successfully logged into the system. |
| **Basic Flow** | ***Basic Flow***   1. Actor enters email and password. 2. System validates credentials. 3. System verifies user role. 4. System redirects user to their respective dashboard.   ***Exceptions***   1. Incorrect credentials. 2. Account not found. 3. System error. |

****

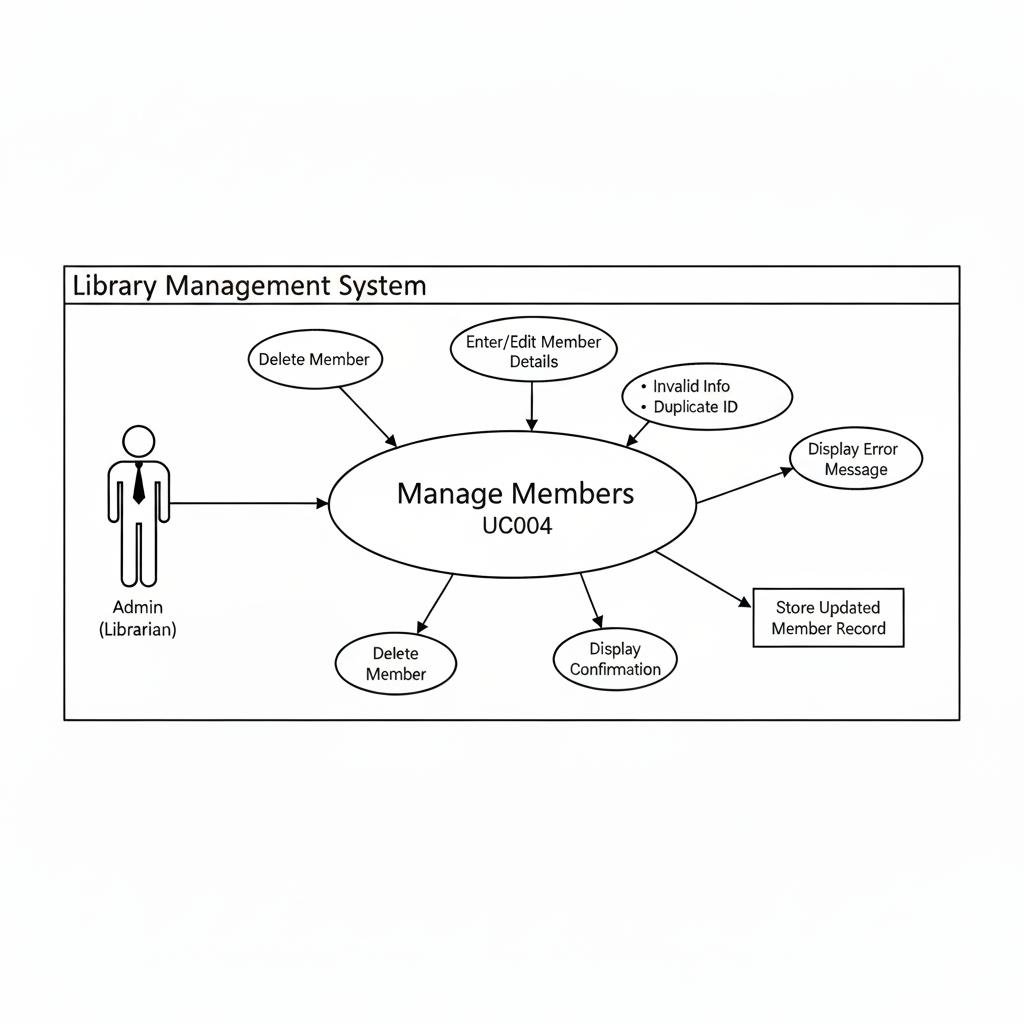
* 1. **Use Case 3: Manage Books**

|  |  |
| --- | --- |
| **ID** | UC003 |
| **Name** | Manage Books |
| **Description** | Admin adds, updates, deletes, or views book records. |
| **Requirement(s)** | FR002 |
| **Actor(s)** | Admin |
| **Precondition** | Admin must be logged in. |
| **Postcondition** | Book records are successfully added, updated, or removed. |
| **Basic Flow** | ***Basic Flow***   1. Admin selects "Add Book". 2. Admin enters book details (Title, Author, ISBN, Category, Copies). 3. System validates details. 4. System stores the book in the database.   ***Alternative Flow***   1. Update Book: Admin selects an existing book → edits details → saves changes. 2. Delete Book: Admin selects a book → deletes it → system removes record.   ***Exceptions***   1. Invalid or missing book details. 2. Duplicate ISBN entered. 3. Database errors. |

****

* 1. **Use Case 4:** **Manage Members**

|  |  |
| --- | --- |
| **ID** | UC004 |
| **Name** | Manage Members |
| **Description** | Admin can add, update, or delete member records. |
| **Requirement(s)** | FR003 |
| **Actor(s)** | Admin |
| **Precondition** | Admin must be logged in. |
| **Postcondition** | Member record is created, updated, or removed. |
| **Basic Flow** | ***Basic Flow***   1. Admin selects "Manage Members". 2. Admin enters or edits member details. 3. System validates data. 4. System stores updated member record.   ***Exceptions***   1. Invalid member information. 2. Duplicate member ID. |

****

* 1. **Use Case 5:** **Search Books**

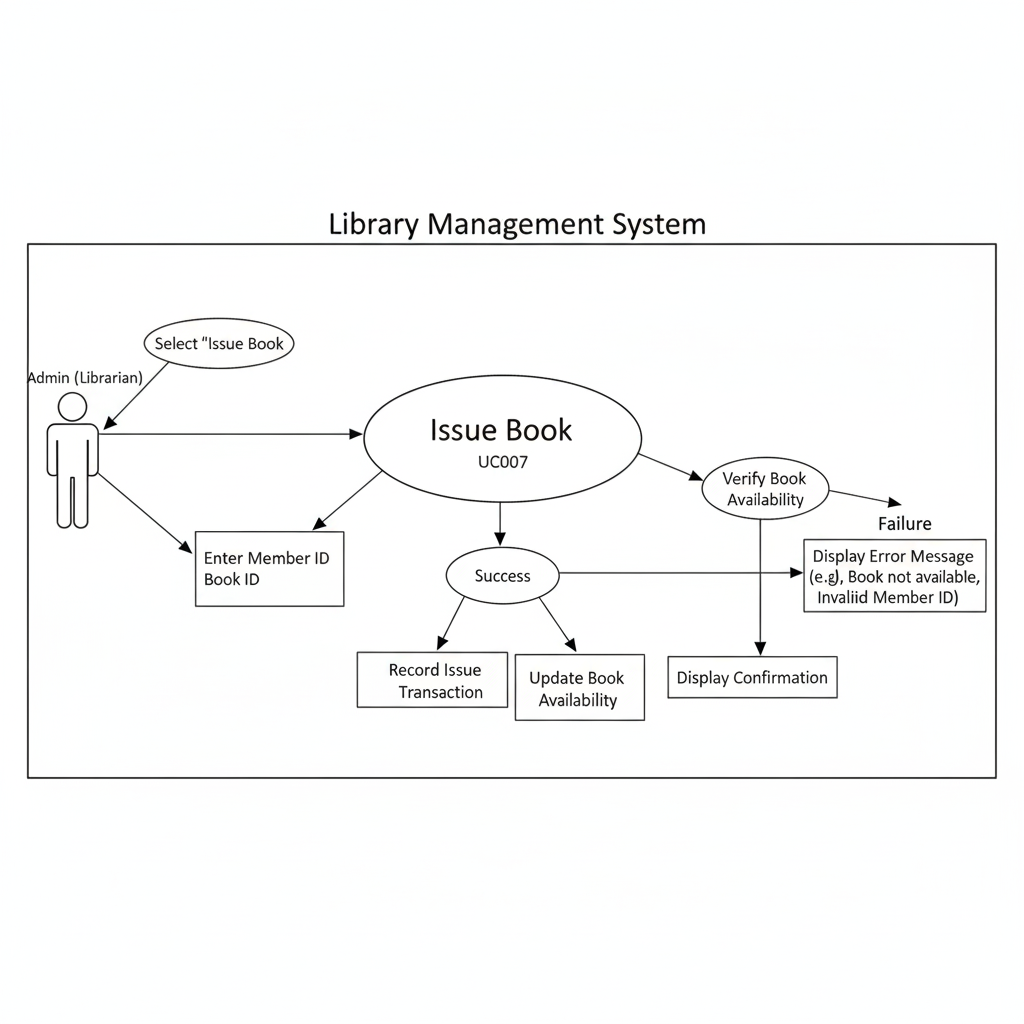
|  |  |
| --- | --- |
| **ID** | UC005 |
| **Name** | Search Books |
| **Description** | User searches for books based on keywords. |
| **Requirement(s)** | FR004 |
| **Actor(s)** | Admin, Member |
| **Precondition** | User is logged in. |
| **Postcondition** | A list of matching books is displayed. |
| **Basic Flow** | ***Basic Flow***   1. Actor enters keyword (title/author/ISBN/category). 2. System searches the database. 3. System displays list of matching books with availability status.   ***Exceptions***   1. No book found. 2. Database connection issue. |

* 1. **Use Case 6:** **Request Book**

|  |  |
| --- | --- |
| **ID** | UC006 |
| **Name** | Request Book |
| **Description** | Member requests a book that is unavailable or requires approval. |
| **Requirement(s)** | FR008 |
| **Actor(s)** | Library Member |
| **Precondition** | Member is logged in. |
| **Postcondition** | Request is recorded and sent to admin. |
| **Basic Flow** | ***Basic Flow***   1. Member selects a book. 2. Member clicks “Request Book.” 3. System records the request. 4. System shows confirmation.  **Exceptions**  1. Member account inactive. 2. Book not found. 3. System error. |

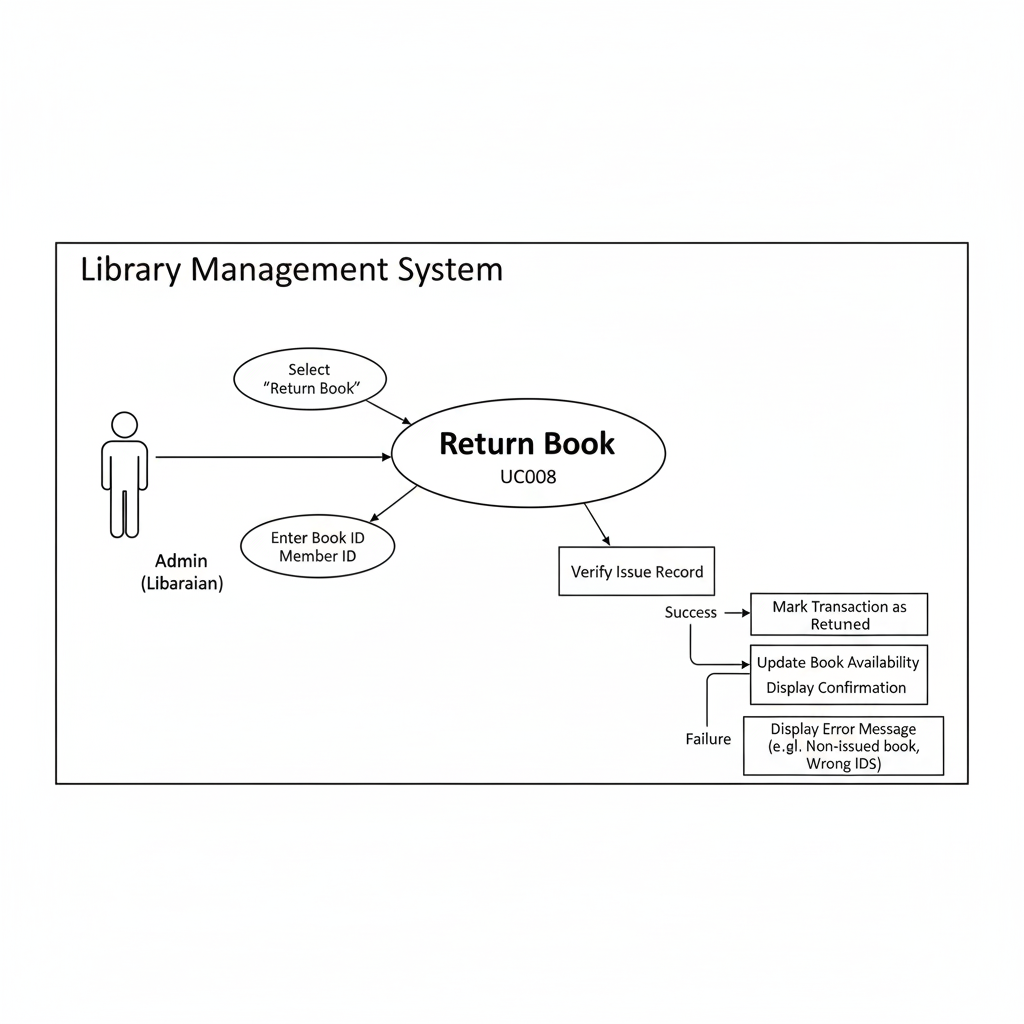
* 1. **Use Case 7:** **Issue Book**

|  |  |
| --- | --- |
| **ID** | UC007 |
| **Name** | Issue Book |
| **Description** | Admin issues a book to a member. |
| **Requirement(s)** | FR005 |
| **Actor(s)** | Admin |
| **Precondition** | Book must be available.  Member must have an active account. |
| **Postcondition** | Book is issued; availability decreases by one. |
| **Basic Flow** | ***Basic Flow***   1. Admin selects “Issue Book.” 2. Admin enters Member ID and Book ID. 3. System verifies book availability. 4. System records issue transaction. 5. System updates availability.   ***Exceptions***   1. Book not available. 2. Member ID invalid. 3. System error during transaction. |

****

* 1. **Use Case 8:** **Return Book**

|  |  |
| --- | --- |
| **ID** | UC008 |
| **Name** | Return Book |
| **Description** | Admin processes book return. |
| **Requirement(s)** | FR006 |
| **Actor(s)** | Admin |
| **Precondition** | Book must have been issued to the member. |
| **Postcondition** | Book is marked returned; availability increases. |
| **Basic Flow** | ***Basic Flow***   1. Admin selects “Return Book.” 2. Admin enters Book ID and Member ID. 3. System verifies issue record. 4. System marks the transaction as returned. 5. System updates availability.   ***Exceptions***   1. Return attempted for a non-issued book. 2. Wrong Book ID or Member ID. 3. System error. |

****

1. **References**
2. [1] I. Sommerville, *Software Engineering*, 10th ed. Boston, MA, USA: Pearson, 2016.
3. [2] IEEE Computer Society, *IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications*. IEEE, 1998.
4. [3] A. Tripathi and A. Kumar, "Library Management System: Design and Implementation," *International Journal of Computer Applications*, vol. 133, no. 14, pp. 1–6, Jan. 2016.