**University Library Management System**

**Table of Contents**

1. Software Requirements Specification (SRS)

1. Introduction

2. Overall Description

3. External Interface Requirements

4. System Features (Functional Requirements)

5. Non-Functional Requirements

6. Other Requirements

Appendix: Terminology / Glossary / Definitions List

2. Data Flow Diagrams

i. Level 0

ii. Level 1

iii. Level 2

3. Entity Relationship Diagram

4. Data Dictionary

5. Structured Chart

6. Use Case Diagram

7. Use Case Diagram Scenarios

8. Link to code

9. Test Report Analysis

**SOFTWARE REQUIREMENTS SPECIFICATION**

1. Introduction  
 1.1 Purpose  
 1.2 Scope  
 1.3 Intended Audience  
 1.4 Additional Information  
 1.5 Contact Information / SRS Team Members  
 1.6 References

2. Overall Description  
 2.1 Product Perspective  
 2.2 Product Functions  
 2.3 User Classes and Characteristics  
 2.4 Operating Environment  
 2.5 User Environment  
 2.6 Design / Implementation Constraints  
 2.7 Assumptions and Dependencies

3. External Interface Requirements  
 3.1 User Interfaces  
 3.2 Hardware Interfaces  
 3.3 Software Interfaces

4. System Features  
 4.1 System Feature A  
 4.1.1 Description and Priority  
 4.1.2 Action / Result  
 4.1.3 Functional Requirements

4.2 System Feature B  
 4.2.1 Description and Priority  
 4.2.2 Action / Result  
 4.2.3 Functional Requirements

4.3 System Feature C  
 4.3.1 Description and Priority  
 4.3.2 Action / Result  
 4.3.3 Functional Requirements

4.4 System Feature D  
 4.4.1 Description and Priority  
 4.4.2 Action / Result  
 4.4.3 Functional Requirements

5. Other Non-Functional Requirements  
 5.1 Performance Requirements  
 5.2 Safety Requirements  
 5.3 Security Requirements  
 5.4 Software Quality Attributes

6. Other Requirements

Appendix: Terminology / Glossary / Definitions List

**1. Introduction**

**1.1 Purpose**

The University Library Management System is designed to automate and streamline book borrowing, returning, and inventory management for both students/faculty and librarians. It ensures efficient handling of library resources, accurate tracking of borrowing activity, and systematic fine calculation — all while providing a seamless web-based user experience.

**1.2 Scope**

This system provides:

● A secure platform for user and librarian login with email and password.

● User dashboard with book search, filter, request, and view borrowed books status features.

● Librarian dashboard for book inventory management, request approval, return status updates, and fine calculations.

● Automatic handling of return dates (30 days from issue) and fine computation (₹1/day after the due date).

● Real-time updates on book availability and return prediction when a book is out of stock.

**1.3 Intended Audience**

● University students and faculty members.

● Library administrative staff (librarians).

**1.4 Additional Information**

The system ensures secure, real-time handling of book transactions and smooth role-based access. It automates fines, manages book request approvals, and provides real-time visibility into book availability and user’s borrowing history.

**1.5 Contact Information / SRS Team Members**

**1.6 References**

● SE Lab - Experiment 2.pdf

● Google

● Wikipedia

**2. Overall Description**

**2.1 Product Perspective**

The University Library Management System is a full-stack web application designed to manage book transactions and inventory.

It features:

● Secure login and authentication for students/faculty, and librarians.

● Book search and filter functionalities.

● A request-approval flow between users and librarians.

● Automatic fine and return date calculation based on transaction logs.

● An intuitive interface for both user and librarian interactions.

**2.2 Product Functions**

**User Functions:**

● Login with a valid ‘srmap.edu.in’ email and password.

● Search and filter books.

● Request books, view borrowed books, return dates, and fines.

● Get expected availability date if a requested book is out of stock.

**Librarian Functions:**

● Login with librarian ID and password.

● Manage book inventory (add/update/remove books).

● View and manage student requests, approve or reject them.

● Update return status, track issued books, and calculate fines.

● View user database (user ID, name, email, phone number).

**2.3 User Classes and Characteristics**

● **User (Student/Faculty):**

○ Search, request, and manage their borrowed books.

○ View fines and return dates.

● **Librarian:**

○ Approve/reject requests.

○ Manage book inventory and update transactions.

○ Track fines and maintain the student inventory log.

**2.4 Operating Environment**

● **Frontend:** ReactJS, HTML, CSS, JavaScript.

● **Backend:** Node.js, Express.js.

● **Database:** MySQL.

● **Platform:** Cross-platform browser compatibility (Chrome, Firefox, Edge, etc.).

● **OS Support:** Windows, macOS, Linux, Android, iOS.

**2.5 User Environment**

● Stable internet connection is required.

● Users access the system via modern browsers.

● Email and password are necessary for authentication.

**2.6 Design / Implementation Constraints**

● Students and faculty must log in with a valid ‘srmap.edu.in’ email ID.

● Books can only be requested if the inventory count is above zero.

● Book requests must be approved by librarians.

● Return dates are automatically calculated (30 days from issue).

● Fine is calculated at ₹1 per day for overdue books.

● Only librarians can modify inventory and transaction records.

**2.7 Assumptions and Dependencies**

● Users possess valid college-issued email IDs and credentials.

● Internet access is mandatory for using the system.

● MySQL will handle persistent data storage for all entities.

● Changes in the database schema will directly affect application logic.

● System must maintain high availability during academic hours.

● In a real-time university library management system, user email IDs, passwords, and user details would typically be fetched directly from the university's central authentication and user database. However, for this project, since live university database integration is not possible, a mock database with a limited set of sample entries is used to simulate real-world behaviour.

**3. External Interface Requirements**

**3.1 User Interfaces**

● Role-based login pages for students/faculty and librarians.

● User dashboard: Book search, filter, request, and borrowed books list.

● Librarian dashboard: Book management, user management, and transaction handling.

● Request approval panel for librarians.

**3.2 Hardware Interfaces**

● Server machine hosting Node.js backend and MySQL.

● Client devices with minimum 4GB RAM and internet access.

● Compatible across desktops, laptops, tablets, and smartphones.

**3.3 Software Interfaces**

● MySQL Database Server.

● ReactJS frontend.

● Node.js with Express.js backend.

● Web browsers for accessing the application.

**4. System Features**

**4.1 System Feature A: Login System**

**4.1.1 Description and Priority** Secure login system with role-based access for students/faculty and librarians.

**4.1.2 Action / Result** Action: Users enter credentials.  
 Result: Credentials are validated against the login database. Access is granted or denied.

**4.1.3 Functional Requirements**

● Login system checks email format for srmap.edu.in domain (users) and admin ID for librarians.

● Password validation against stored database records.

● Session-based authentication and access control.

**4.2 System Feature B: User Functionalities**

**4.2.1 Description and Priority**  Allows users to interact with the library by searching, requesting books, and tracking their borrowed books status.

**4.2.2 Action / Result**  Action: User logs in and performs book-related actions.  
 Result: System handles requests, updates user dashboards, and manages fine and availability status.

**4.2.3 Functional Requirements**

● Search and filter books by subject etc.

● Request a book (request is sent for librarian approval).

● View borrowed books list, return dates, and fines.

● Notification of next available date if a book is unavailable.

**4.3 System Feature C: Librarian Functionalities**

**4.3.1 Description and Priority** Empowers librarians to manage book records, approve requests, and monitor user activity.

**4.3.2 Action / Result** Action: Librarian adds/removes/updates book records, approves requests, updates return status, and checks fines.  
 Result: Database updates inventory, request status, and transaction records accordingly.

**4.3.3 Functional Requirements**

● Add, update, and remove books from the inventory.

● View user database (User ID, Email, Name, Phone Number).

● Approve/reject book requests.

● Update the return status of borrowed books.

● Monitor fines for each user.

**4.4 System Feature D: Approval System**

**4.4.1 Description and Priority** Manages the approval flow of book borrowing requests by the librarian.

**4.4.2 Action / Result** Action: User submits a book request; librarian approves or rejects it.  
 Result: If approved, the issue date is set, the return date is calculated (30 days later), and the inventory count is reduced by one.

**4.4.3 Functional Requirements**

● New requests automatically created when users submit a request.

● Librarian manually approves or rejects each request.

● Issue and return dates are generated post-approval.

● Fine calculation based on actual return status and delay days.

**5. Other Non-Functional Requirements**

**5.1 Performance Requirements**

● Support multiple concurrent user sessions.

● Book search results should load within 3-5 seconds.

● Reliable uptime, especially during peak academic hours.

**5.2 Safety Requirements**

● Passwords must be securely hashed in the database.

● Role-based access prevents unauthorized data manipulation.

● Periodic database backups to prevent data loss.

**5.3 Security Requirements**

● Secure password storage and validation.

● Input validation to prevent SQL injection and XSS attacks.

● User data confidentiality must be maintained.

**5.4 Software Quality Attributes**

● Scalability for future student and book growth.

● Maintainability for ease of bug fixes and feature upgrades.

● Portability across devices and browsers.

**6. Other Requirements**

Appendix: Terminology / Glossary / Definitions List

● **Inventory Management:** Tracking the count and status of books in the library.

● **Authentication:** The verification process for users and librarians before accessing the system.

● **Role-Based Access Control (RBAC):** Access restrictions based on user role (User or Librarian).

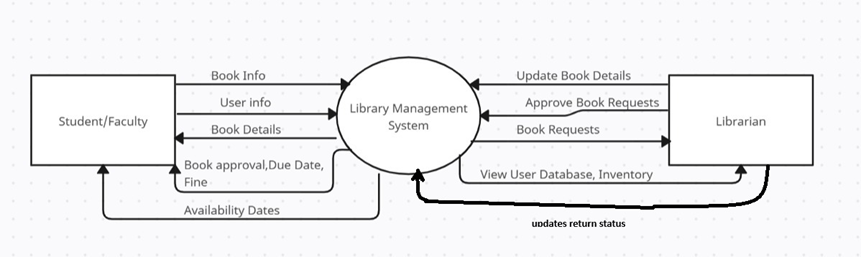
● **Student Inventory:** Database log storing user-borrowed books, issue date, return date, and fine.

● **Request System:** Mechanism for users to request books, pending librarian approval.

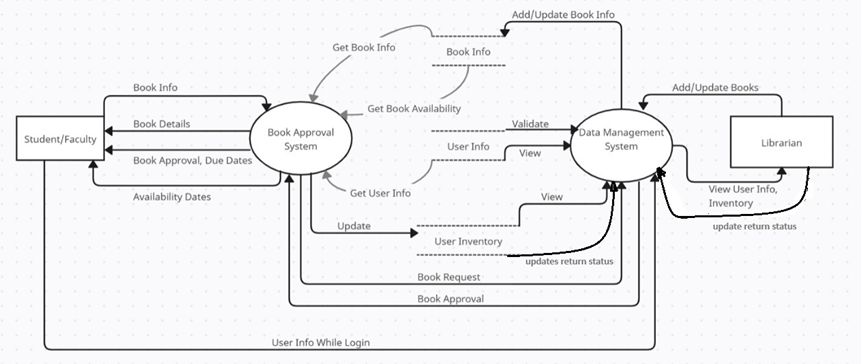
● **Fine Calculation:** ₹1 charged for each day after the return date until the book is returned.

**Data Flow Diagrams (DFD)**

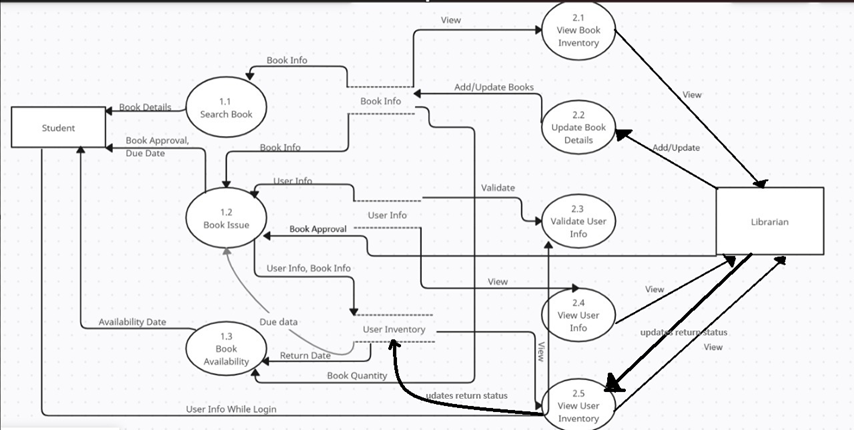
**Context Diagram (Level 0):**

****

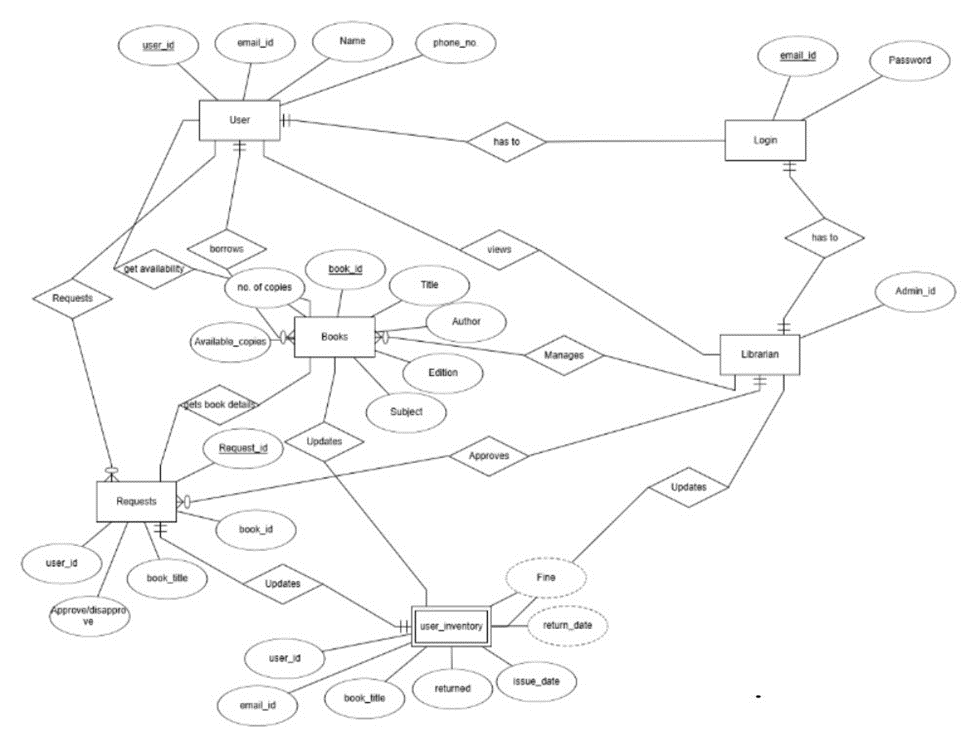
**Level 1 (DFD):**

****

**Level-2 DFD:**

****

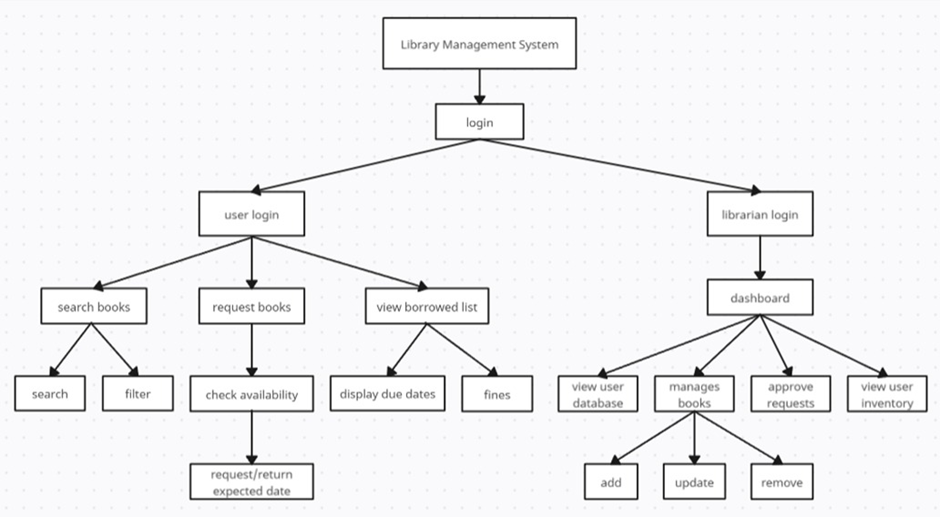
**Entity Relationship Diagram**

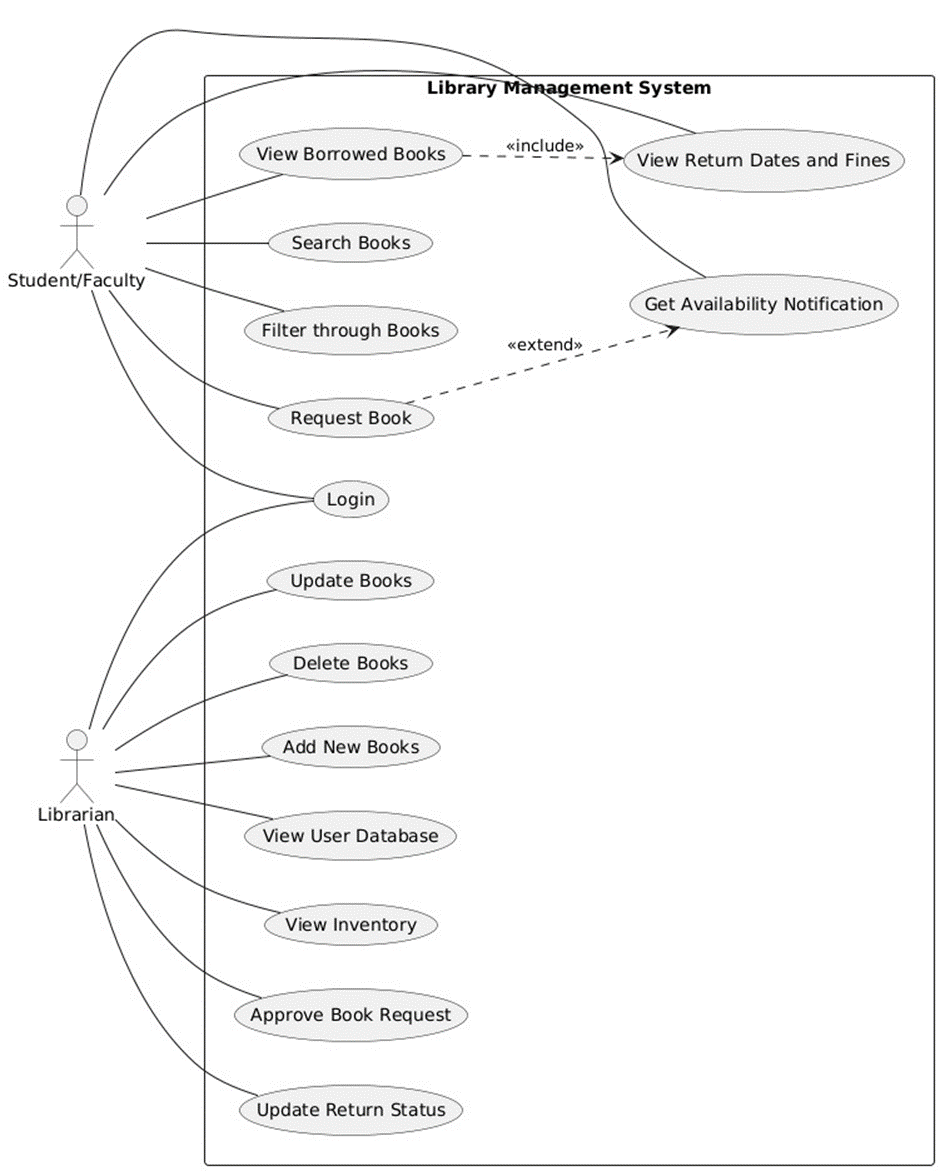
****

**Data Dictionary**

| **Table Name** | **Field Name** | **Data Type** | **Description** | **Constraints** |
| --- | --- | --- | --- | --- |
| **UserLogin** | email\_id | VARCHAR (100) | User's college-issued email address | Primary Key |
|  | password | VARCHAR (100) | Encrypted user password | Not Null |
| **LibrarianLogin** | admin\_id | VARCHAR (50) | Unique Librarian ID | Primary Key |
|  | password | VARCHAR (100) | Encrypted librarian password | Not Null |
| **UserDatabase** | user\_id | INT | Unique user ID | Primary Key, Auto Increment |
|  | email\_id | VARCHAR (100) | User’s college email address | Unique, Not Null |
|  | name | VARCHAR (100) | name of the user | Not Null |
|  | phone\_Number | VARCHAR (15) | Contact number of the user | Not Null |
| **Books** | book\_id | INT | Unique book ID | Primary Key, Auto Increment |
|  | book\_title | VARCHAR (200) | Title of the book | Not Null |
|  | author | VARCHAR (100) | Author of the book | Not Null |
|  | edition | VARCHAR (50) | Edition information | Not Null |
|  | subject | VARCHAR (100) | Subject or category | Not Null |
|  | no\_of\_copies | INT | Total number of copies held by library | Not Null |
|  | available\_books | INT | Number of copies currently available for borrowing | Not Null |
| **Requests** | request\_id | INT | Unique request ID | Primary Key, Auto Increment |
|  | user\_id | INT | Reference to user | Foreign Key |
|  | book\_id | INT | Reference to book | Foreign Key |
|  | book\_title | VARCHAR (200) | Title of the requested book | Redundant storage for reference |
|  | approve\_status | BOOLEAN | Approval status by librarian (TRUE=approved, FALSE=pending) | Default FALSE |
| **Student\_Inventory** | si\_id | INT | Unique ID for each borrowed entry | Primary Key, Auto Increment |
|  | user\_id | INT | Reference to user | Foreign Key |
|  | user\_id | INT | Reference to user | Foreign Key |
|  | book\_id | INT | Reference to book | Foreign Key |
|  | book\_title | VARCHAR (200) | Title of the borrowed book | Not Null |
|  | issue\_date | DATE | Date when the book was issued | Auto-Generated on Approval |

**Structured Chart**

****

**Use case Diagram**

**Use Case Scenarios**

**Use Case Scenario: Borrowing a Book (Request Book)**

| **Use Case ID** | UC-001 |
| --- | --- |
| **Use Case** | Borrowing a Book (Request Book) |
| **Actors** | Student/Faculty, Librarian |
| **Description** | This use case allows a student or faculty member to search for and request a book. The request is then reviewed and approved by the librarian. |
| **Pre-Conditions** | The user must be logged in through the login page to access the system. |
| **Flow of Events** | 1. The student/faculty logs into the system.  2. The student searches for a book using the search feature.  3. The system displays available books.  4. The student selects a book and clicks the Request Book button.  5. The system sends the request to the librarian for approval.  6. The librarian reviews the request and approves it.  7. The student is notified that the book has been successfully borrowed.  8. The student can view the borrowed book with its return date under the Borrowed Books section. |
| **Post-Conditions** | The student has borrowed the book and can view it under the borrowed books list. The system updates the book’s status and return date. |
| **Alternative Flows** | 1. If the book is unavailable, the system notifies the student of the availability date.  2. If the librarian rejects the request, the student Borrow book section will not be updated.  3. If the student is not logged in, they are redirected to the login page. |

**Use Case: View Borrowed Books**

| **Use Case ID** | UC-002 |
| --- | --- |
| **Use Case** | View Borrowed Books |
| **Actors** | Student/Faculty |
| **Description** | Allows the student/faculty member to view the list of books they have borrowed, including the return dates and fines. |
| **Pre-Conditions** | The user must be logged in. |
| **Flow of Events** | 1. The student/faculty logs into the system.  2. The student/faculty navigates to the View Borrowed Books section.  3. The system displays a list of all borrowed books, including the borrow date, return date, and fines if applicable. |
| **Post-Conditions** | The student/faculty is able to see all borrowed books with relevant details. |
| **Alternative Flows** | If no books are borrowed, the system shows a message saying "No borrowed books." |

**Use Case: Librarian Approves Book Request**

| **Use Case ID** | UC-003 |
| --- | --- |
| **Use Case** | Approves Book Request |
| **Actors** | Librarian |
| **Description** | This use case allows the librarian to approve or reject a book request made by a student or faculty member. |
| **Pre-Conditions** | The librarian must be logged in and have pending book requests to approve. |
| **Flow of Events** | 1. The librarian logs into the system.  2. The librarian navigates to the Pending Requests section.  3. The librarian reviews the requests and clicks Approve or Reject.  4. If approved, the student’s borrowed books section is updated.  5. If rejected, the student’s borrowed books section is not updated. |
| **Post-Conditions** | The student’s borrowed books section is updated. |
| **Alternative Flows** | If the librarian rejects the request, the student’s borrowed books section is not updated. |

**Use Case: Librarian Manges Books**

| **Use Case ID** | UC-004 |
| --- | --- |
| **Use Case** | Add, Delete, and Update Books in the Inventory |
| **Actors** | Librarian |
| **Description** | This use case allows the librarian to add new books to the library system, update details of existing books, and delete books from the system. |
| **Pre-Conditions** | The librarian must be logged into the system and have proper authorization to manage books. |
| **Flow of Events** | 1. The librarian logs into the system.  2. The librarian navigates to the Manage Books section.  Add Book:  3. The librarian clicks on Add New Book.  4. The librarian enters the book details (title, author, ISBN, category, etc.).  5. The librarian clicks Save to add the book to the system.  Update Book:  6. The librarian selects an existing book from the list.  7. The librarian updates the necessary book details.  8. The librarian clicks Save to update the book information.  Delete Book:  9. The librarian selects a book to delete from the list.  10. The librarian confirms the deletion.  11. The book is removed from the inventory and no longer available for borrowing. |
| **Post-Conditions** | After adding a book, the new book is available in the library inventory. After updating, the book information is updated in the system. After deleting, the book is no longer in the system. |
| **Alternative Flows** | 1. If any book detail is missing or incorrect (e.g., no title or author), the system prompts the librarian to fill in the missing information. |

**Link to code**

**Testing Report Analysis**

**Testing Objectives**

● Validate that all system functionalities meet the specified requirements.

● Identify and fix any bugs or logical errors.

● Ensure the system is secure and performs well under expected user loads.

● Verify smooth integration between frontend, backend, and database.

**Testing Tools**

● Postman (API testing).

● MySQL Workbench (Database validation).

● React Testing Library (Frontend unit testing).

● Manual testing for UI flow.

**Testing Strategies**

● **Unit Testing:** Focused on individual components such as login, book search, request handling, inventory updates, and fine calculation.

● **Integration Testing:** Checked the flow of data between frontend, backend, and database.

● **System Testing:** Verified the complete system for functional and non-functional requirements.

● **User Acceptance Testing (UAT):** Tested by mock users (students and librarians) to ensure the system meets user expectations.

● **Performance Testing:** Evaluated system response time, especially for database-heavy operations like search and book inventory updates.

**Test Cases Overview**

| **Test Case** | **Description** | **Expected Result** | **Status** |
| --- | --- | --- | --- |
| Login Authentication | User enters valid and invalid credentials. | Grant access / Deny access. | Pass |
| Book Search | Search by title, author, or category. | Returns the correct book list. | Pass |
| Book Request | User requests an available/unavailable book. | Request submitted or waitlisted. | Pass |
| Inventory Update | Librarian updates book count. | Updated count reflected correctly. | Pass |
| Book Return and Fine Calculation | Late return triggers fine calculation. | Fine = ₹1 per overdue day. | Pass |
| Request Approval / Rejection | Librarian processes pending book requests. | Request status updated. | Pass |
| Database Consistency | Cross-check inventory and request updates. | Data remains consistent. | Pass |

**Defect Summary**

Most defects discovered were minor and related to input validation and UI feedback messages. No critical issues were found during system and UAT testing.

● Critical Severity:  
 No critical defects were found during testing.

● High Severity:  
 No high severity bugs were identified.

● Medium Severity:

2 bugs identified:

Input validation errors in forms, allowing invalid data submission.

● Low Severity

3 bugs identified:  
 Minor UI misalignments on various screens

**Conclusion**

The Library Management System successfully passed all major functional and non-functional test cases. The system is stable, user-friendly, and meets all outlined requirements. It is now ready for deployment in an educational institution environment.