A Report

ON

"ISE Project"

"Library Management System"

Representor

Muhammad Uneeb Khan
(243729)

Semester: 2nd Semester

Instructor: Ms. Fatima Yousaf

Submission Date: 15-June-2025

Department of Computer Science Air University Multan

TABLE OF CONTENT:

Contents

Library Management System

Project Overview:

Objective of the System:

The objective of the Library Management System is to simplify and automate the management of library resources, user activities, and book tracking. It aims to ensure efficient book issuance, returns, reservations, and user management in a secure and user-friendly environment.

Key Features:

- **User Registration & Login:**
  Secure user authentication using credentials and library card ID.
- **Book Management:**
  Admins/librarians can add, remove, and update books.
- **Search and Availability:**
  Users can search for books by title, author, ISBN, or category and check availability.
- **Issue & Return Books:**
  Members can borrow and return books; fines are calculated for late returns.
- **Reservations:**
  Books can be reserved if currently unavailable.
- **Notifications:**
  Automated alerts for due dates, returns, and reservation updates.
- **Role Management:**
  Different roles (Admin, Librarian, Member) with specific permissions.
- **History & Feedback:**
  Users can view their borrowing history and give feedback.

Target Users:

- **Students/Members:**
  Individuals borrowing books and using the system.
- **Librarians:**
  Staff responsible for managing book records and transactions.
- **Administrators**:
  Supervisors who control system settings and user roles.

Functional Requirements:

No	Requirement Description
1	The system should allow users to register and log in using unique credentials.
2	Admins should be able to add new books or remove existing ones from the catalog.
3	Users should be able to search for books using title, author, ISBN, or category.
4	The system should allow users to issue available books.
5	Users should be able to return previously issued books.
6	The system should show the number of available copies of a particular book.
7	Users should be able to reserve books that are currently issued.
8	The system should allow different roles (e.g., admin, librarian, member) with different permissions.
9	Users and admins should be able to view borrowing history.
10	The system should send notifications for due dates, overdue books, and reservation availability.

Non-Functional Requirements:

No	Requirement Description
1	The system should respond to user queries within 2 seconds.
2	The system should handle an increasing number of users and books without performance issues.
3	The system should store passwords securely and prevent unauthorized access.
4	The system should be available at least 99% of the time during working hours.
5	The codebase should be structured for easy maintenance and future updates.
6	The system should operate correctly under expected conditions.
7	The interface should be user-friendly and easy to navigate for all users.
8	The system should automatically back up data and support quick recovery in case of failure.
9	The system should run on different devices (PCs, tablets) and operating systems.
10	The system should comply with relevant data protection laws and institutional policies.

User Case Components

1) Actors for LMS:

a) Primary Actors:

- Member/User (borrower, student, etc.)
- Librarian (staff managing books)

b) Secondary Actors:

- Admin (manages system settings, users)

2) Use Cases for LMS:

- Register/Login (For User to access the system)
- Pay Fine (If user get late to return book)
- Search Book (Used by logged-in user)
- Borrow/Issue Book (Done by members with librarian approved)
- Return Book (Performed by members)
- Feedback (User gives his feedback)
- Reserve Book (Only librarian or admin)
- Add/Remove Book (Only librarian or admin)
- Manage Roles (Only admin)
- View Borrowing History (User and librarian access)
- View Availability (All Users)
- Send Notification (Automated function (extends others)

3) Include:

- Issue Book includes Login (must logged in in to issue a book)
- Register/Login includes Registration form to login.
- Return Book includes Sending Notification (to remind user for returning book)
- Register/Login includes Library Card ID.
- Feedback includes filling Feedback form.
- Reserve Book includes login.
- Add/Remove Book includes Search/Update Record.
- Add/Remove Book includes Invalid delete ID.

4) Extend:

- Reserve Book extends must login.
- Add/Remove extends Invalid delete ID.
- Return Book extends send notification.

5) Generalization:

- Student
- Person

6) User Case Diagram:

Unified Modelling Language

UML:

1) Classes:

- User (Base class)
- Member (inherits User class)
- Librarian (inherits User)
- Admin (inherits User)
- Book
- BorrowRecord
- Fine
- Notification
- Feedback
- Reservation

2) Attributes and Methods:

User

- string userID **(protected)**
- string name **(protected)**
- string password **(protected)**
- string librarycardID **(protected)**
- virtual void login (string id,string pass) **(public)**
- void searchBook (string title) (string title) **(public)**
- void view availability() **(public)**
- void viewBorrowHistory() **(public)**

Member

- void borrowBook (Book b) **(public)**
- void returnBook (Book b) **(public)**
- void payFine(float amount) **(public)**
- void giveFeedback(string msg) **(public)**

Librarian

- void issueBook(Book, Member m) **(public)**
- void acceptReturn(Book b,Member m) **(public)**
- void addBook(Book b) **(public)**
- void removeBook(string bookID) **(public)**
- void reserveBook(Book b, Member m) **(public)**

Admin

- void manageRoles(User u,string newRole) **(public)**
- void reserveBook(Book b,Member m) **(public)**
- void addBook(Book b) **(public)**
- void removeBook(string bookID) **(public)**

Book

- string bookID **(private)**
- string title **(private)**
- string author **(private)**
- bool isAvailable **(private)**
- bool getAvailability() **(public)**
- void updateRecord(string title) **(public)**

BorrowRecord

- Member member **(private)**
- Book book **(private)**
- Date BorrowDate **(private)**
- Date returnDate **(private)**
- Fine fine **(private)**
- float calculate() **(public)**
- void sendReturnNotification() **(public)**

Fine

- float amount **(private)**
- float calculate(int daysLate) **(public)**
- void pay(float amount) **(public)**

Notification

- string message **(private)**
- User receipt **(private)**
- Void send() **(public)**

Feedback

- string feedbackID **(private)**
- User user **(private)**
- string message **(public)**
- void submit() **(public)**

Reservation

- Member member **(private)**
- Book book **(private)**
- Date reserveDate **(private)**
- Void reserveBook() **(public)**

2) Relationships:

Inheritance:

- Librarian class inherits User class

Association:

- Notification class from User class

Composition:

- BorrowRecord from Book, Member

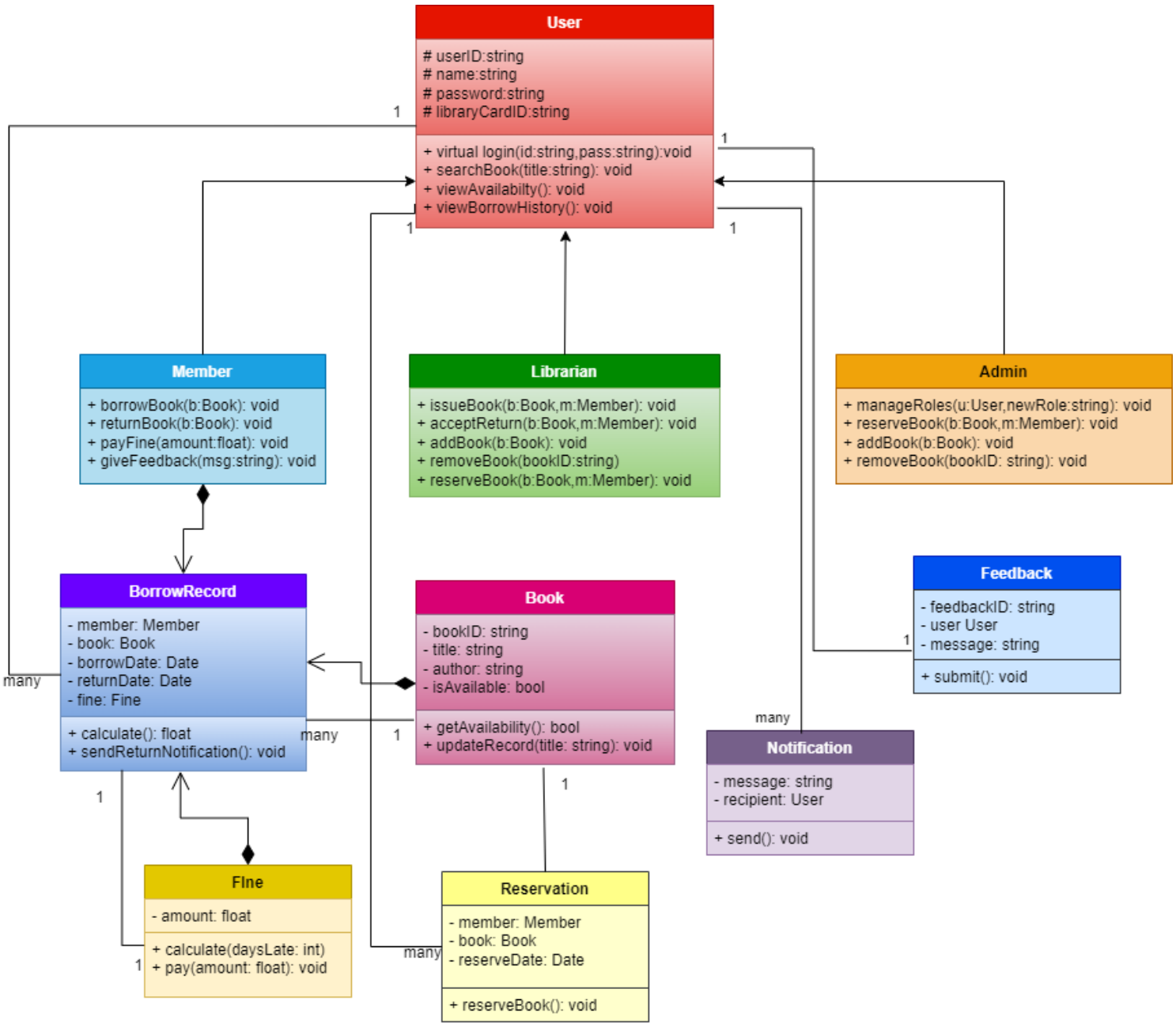
Aggregation:

- BorrowRecord from Fine

Association:

- Reservation from Book, Member

4) UML Diagram:



Sequence Diagram:

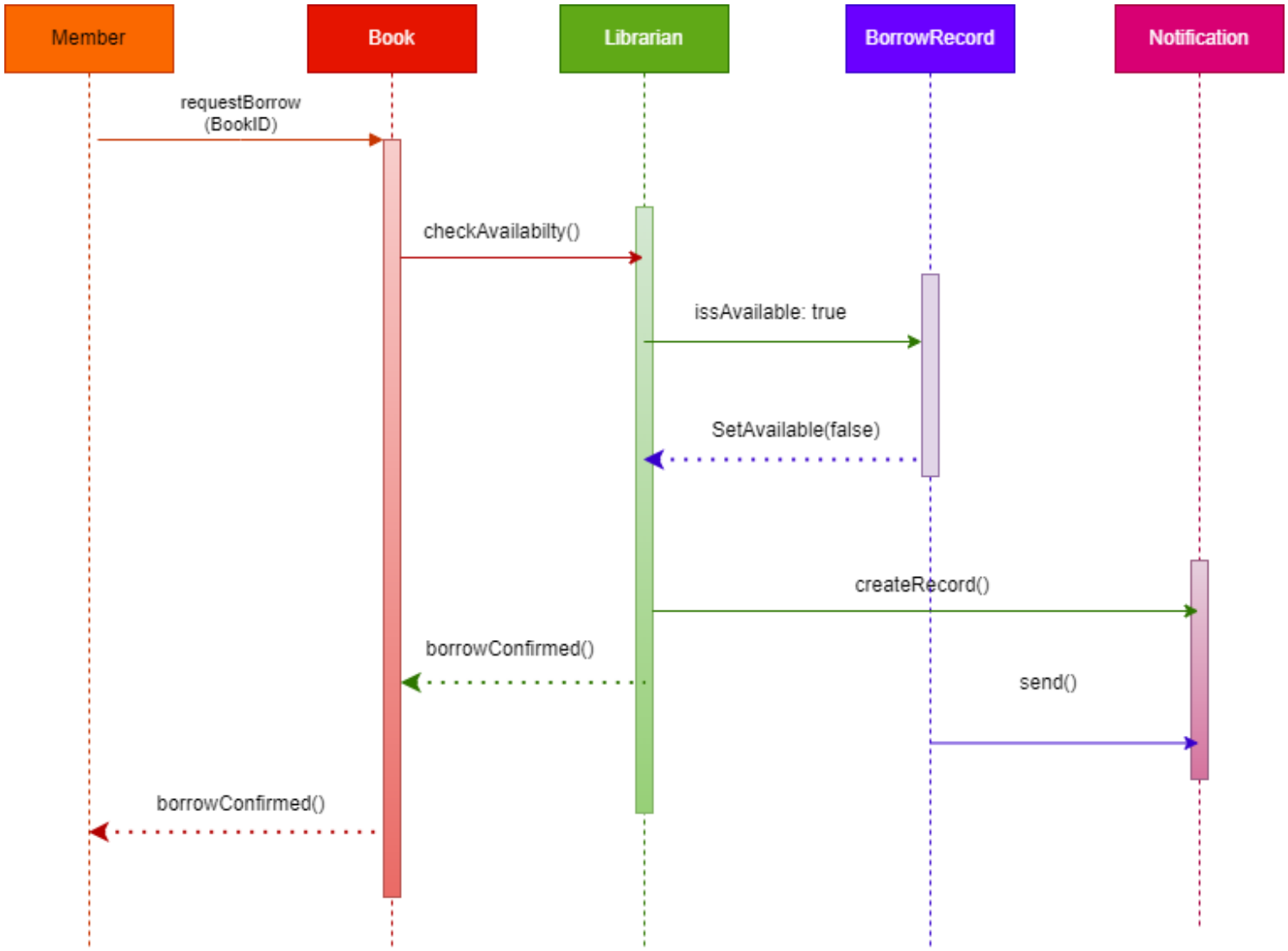
Sequence Diagram:

1) Explanation of Flow:

Scenario: A Member Borrows a Book

- **Member** requests the **Librarian** to borrow a book by providing the Book ID.
- **Librarian** checks the availability by calling `checkAvailability()` on the **Book** object.
- If the book is available, the **Librarian** calls `issueBook(Book, Member)`.
- The **Book** object updates its availability status (e.g., `isAvailable = false`).
- A **BorrowRecord** is created and stores details like member, book, borrow date, etc.
- A **Notification** is generated to inform the member that the book was successfully issued.
- Finally, a confirmation is sent back to the **Member**.

2) Sequence Diagram:



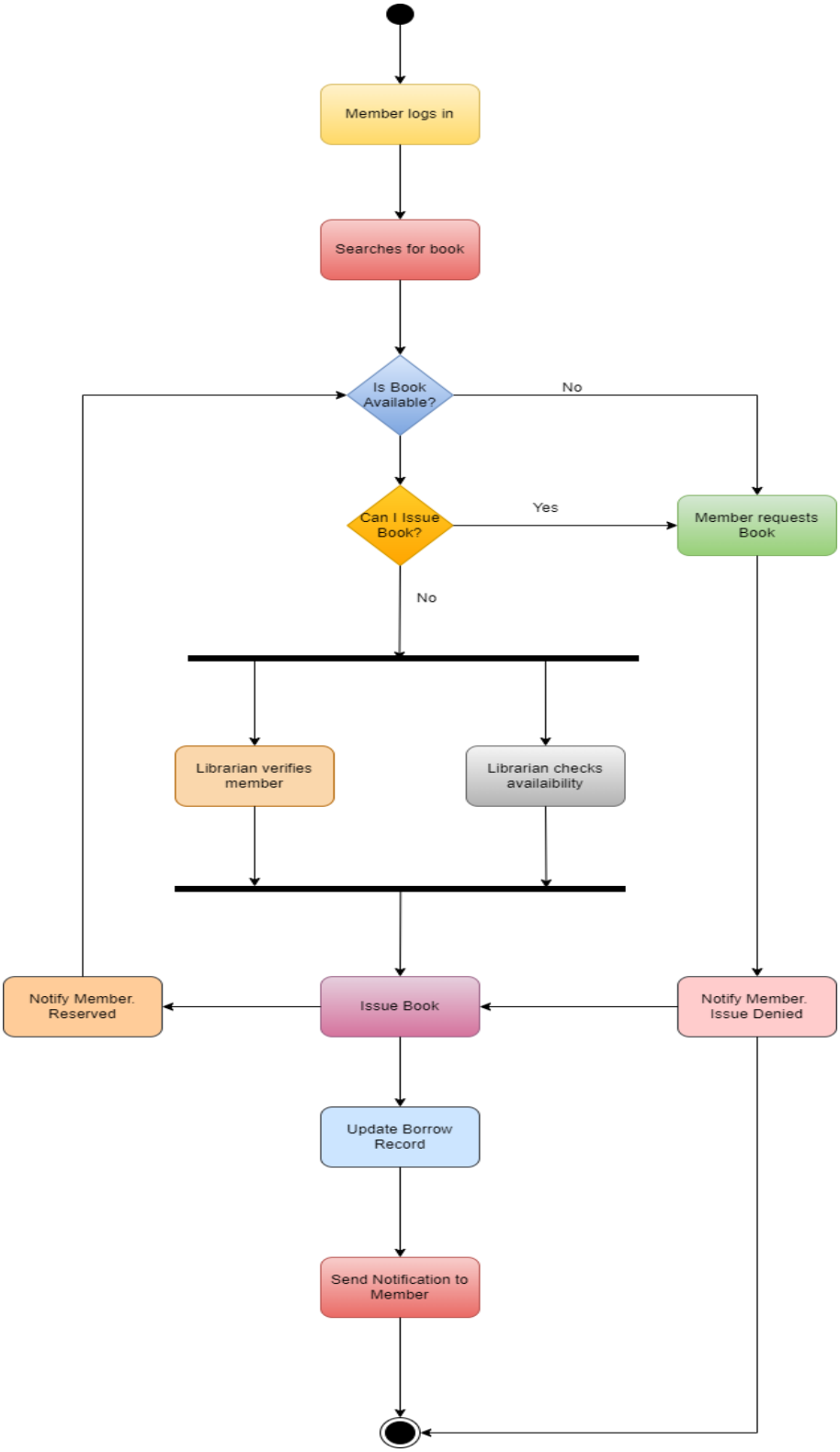
Activity Diagram

Activity Diagram:

1) Explanation of the Scenario:

- **Initial Node**: Member initiates the process by logging in.
- **Activity**: Searches for book, checks availability.
- **Decision**: If the book is available, the borrow process continues. If not, the system goes to the reservation path.
- **Fork Node**: Librarian performs multiple checks in parallel (e.g., verifying member and checking book).
- **Decision Point**: Librarian determines whether the book can be issued.
- **Join Node**: After parallel tasks or reservation handling, the flow converges before ending.
- **Final Node**: Process ends when the book is borrowed or reserved and the member is notified.

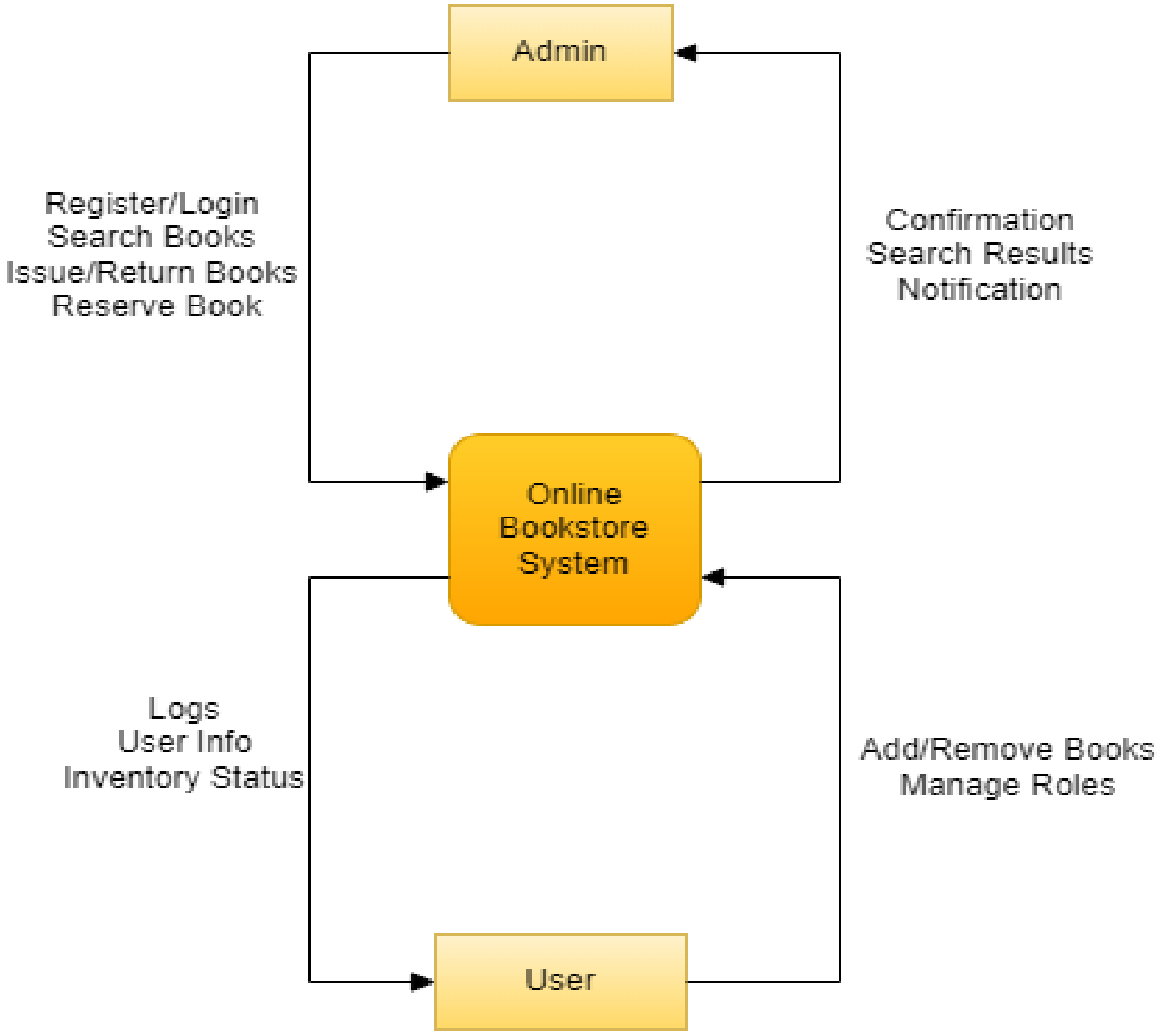
2) Diagram:


Member logs in
Searches for book
Is Book Available?
No
Can I Issue Book?
Yes
Member requests Book
No
Librarian verifies member
Librarian checks availaibility
Notify Member. Reserved
Issue Book
Notify Member. Issue Denied
Update Borrow Record
Send Notification to Member

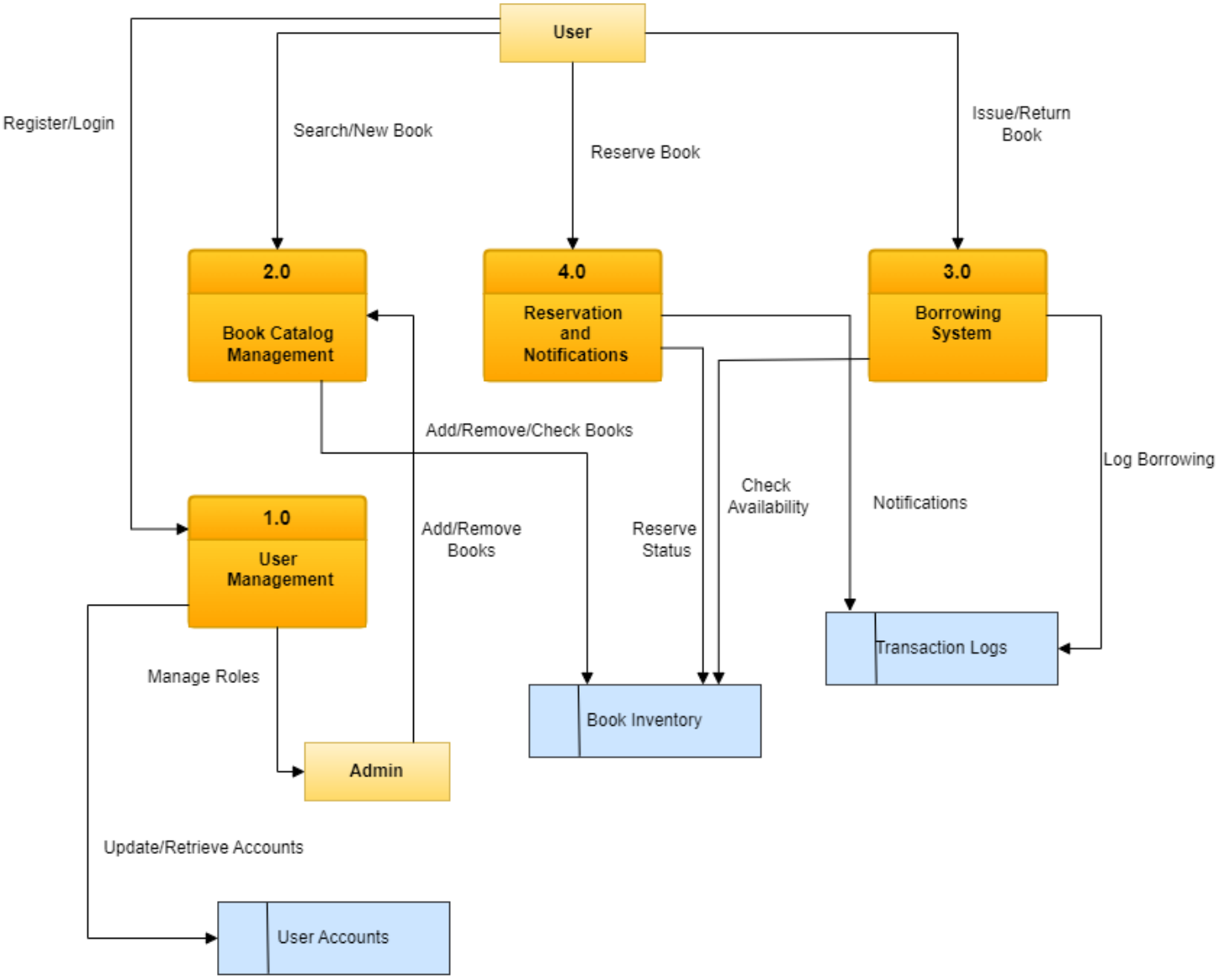
Data Flow Diagram

DFD Diagram:

1) Context Level DFD:

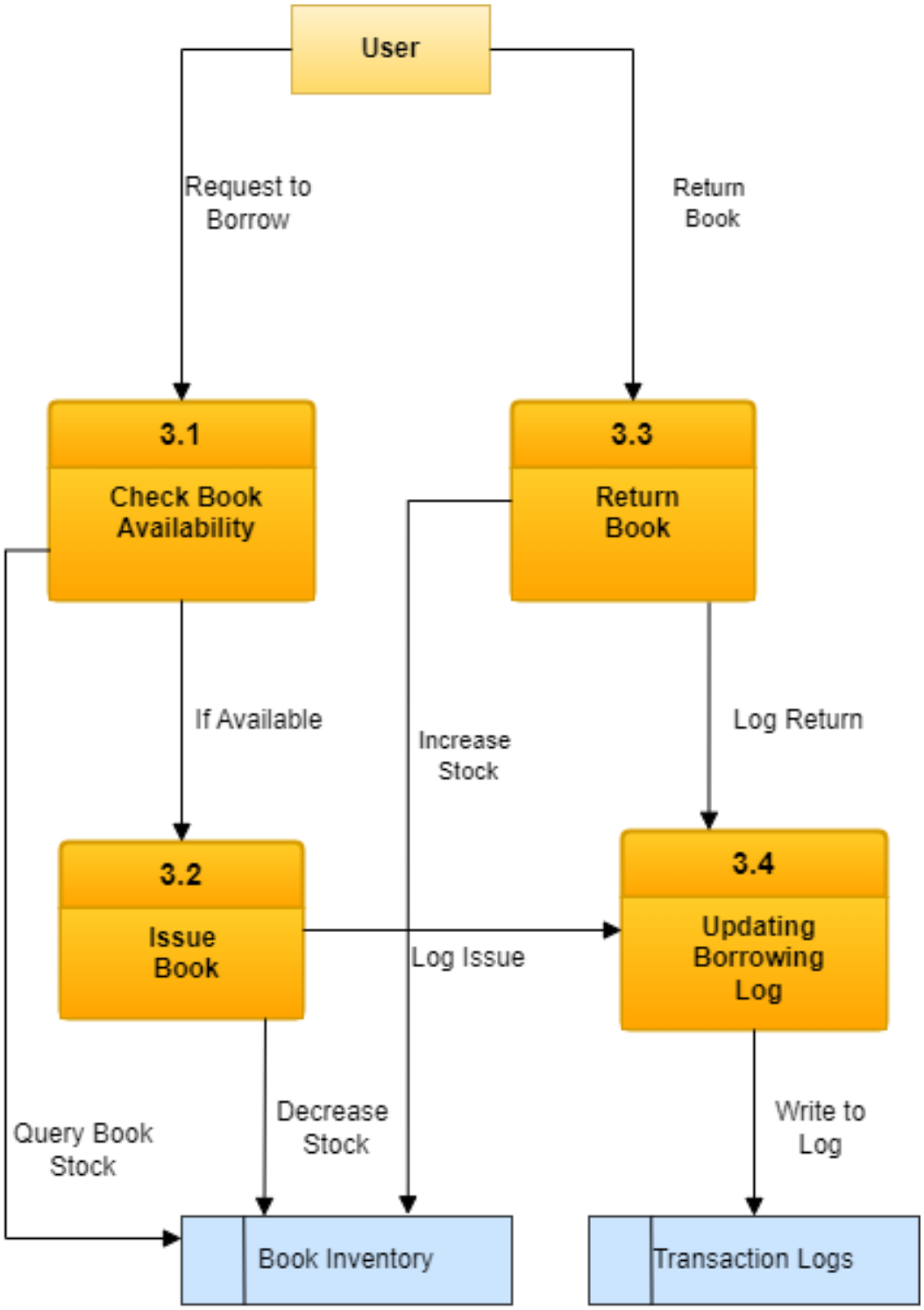


2) Level-0-DFD:


User
Register/Login
Search/New Book
Reserve Book
Issue/Return Book
2.0
Book Catalog Management
4.0
Reservation and Notifications
3.0
Borrowing System
Add/Remove/Check Books
Log Borrowing
1.0
User Management
Add/Remove Books
Reserve Status
Check Availability
Notifications
Transaction Logs
Manage Roles
Book Inventory
Admin
Update/Retrieve Accounts
User Accounts

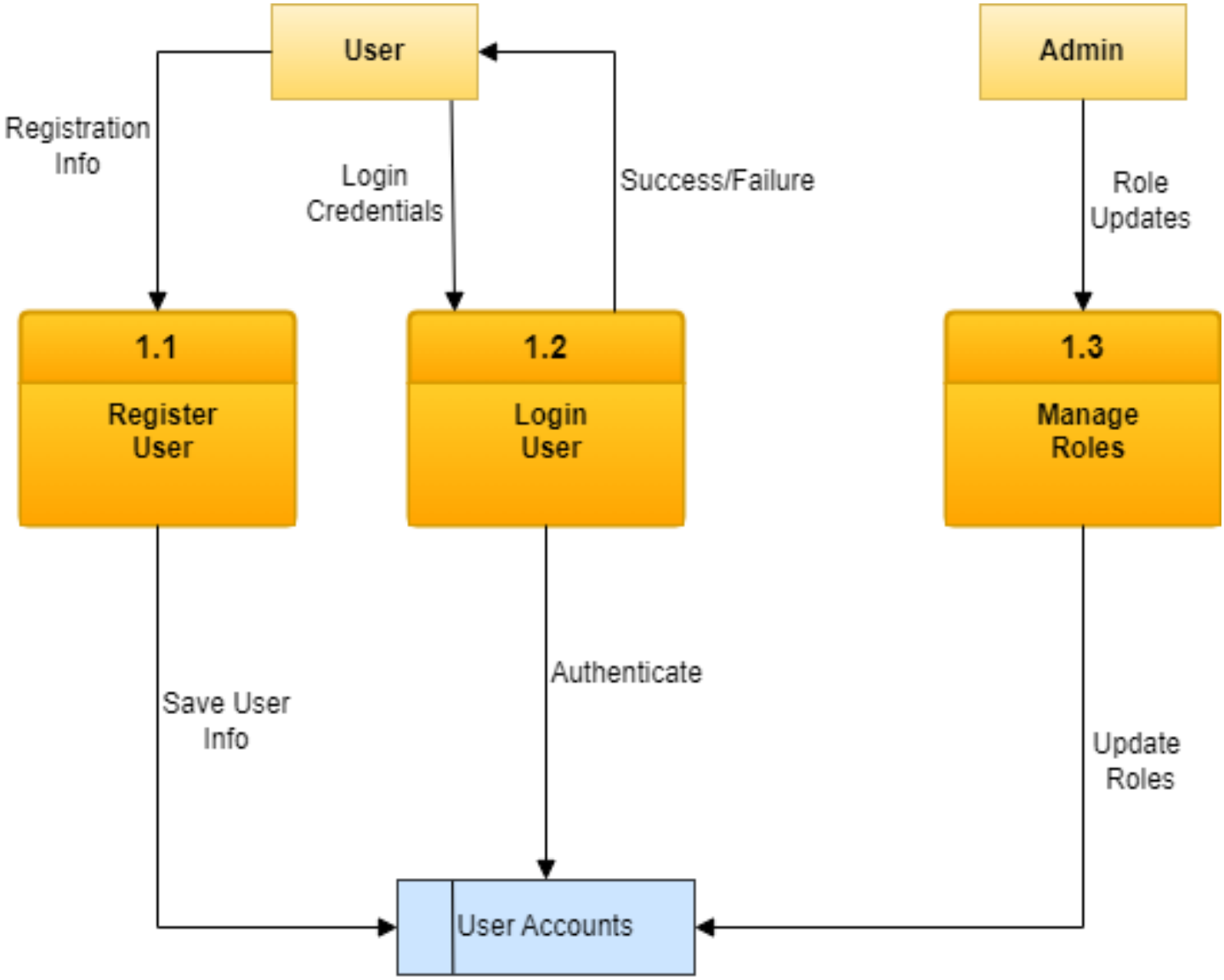
3) Level-1-DFD:

Borrowing System:



4) Level-1-DFD (Optional):

User Management:



Testing Strategy

Testing Strategy:

1) Type of Testing:

- **Unit Testing:**
  Tests individual components or functions such as login(), borrowBook() etc.
- **Integration Testing:**
  Verifies interaction between modules like User login and Borrow module.
- **System Testing:**
  Tests the entire LMS system to validate that it meets the requirements.
- **Acceptance Testing:**
  Validates the system with actual data to ensure it meets user expectations.

2) Test Cases:

Test Case ID	Description	Input	Expected Output	Status
TC-01	Used login with valid credentials	Valid Username & Password	Dashboard appears	Pass
TC-02	User login with invalid credentials	Wrong Username/Password	Error message shown	Pass
TC-03	Book search by title	Book title: "C++ Programming"	Display matching books	Pass
TC-04	Add a new book (Admin)	Book details	Book added successfully	Pass
TC-05	Issue available book	Book ID + User ID	Book issued, status updated	Pass
TC-06	Return book on time	Book ID + User ID	Book returned, no fine	Pass
TC-07	Return book late	Book ID + User ID + Late by 5 days	Fine calculated and notified	Pass
TC-08	Reserve an issued book	Book ID (already issued)	Reservation confirmed	Pass
TC-09	View borrowing history	Logged-in user	List of past issued/returned books	Pass
TC-10	Send notification on due date	Book due today	Email/alert sent to user	Pass

Conclusion:

The Library Management System (LMS) is designed to modernize and streamline the operations of a traditional library. By automating key processes such as user registration, book management, borrowing and returning, fine calculations, and notifications, the system ensures efficient and user-friendly experiences for students, librarians, and administrators alike.

This system addresses both the functional and non-functional requirements needed for reliable operation in real-world environments. With clearly defined user roles and secure data handling, it enhances accountability, reduces manual workload, and minimizes human error. Features like reservation and overdue notifications add value by improving resource accessibility and time management for users.

Expected Impact:

- Increased efficiency in library operations Book Management.
- Enhanced user satisfaction through automated and responsive features.
- Improved resource tracking and data accuracy.
- Simplified user management with role-based access.

Next Steps for Development and Improvement:

- Integration with RFID for automated book scanning and tracking.
- Implementation of a mobile application for easier user access.
- Real-time analytics and reporting dashboards for administrators.
- Multi-language support for broader accessibility.
- Continuous testing and user feedback collection for iterative improvement.