

**RAJALAKSHMI ENGINEERING  
COLLEGE  
RAJALAKSHMI NAGAR, THANDALAM – 602 105**



**RAJALAKSHMI  
ENGINEERING COLLEGE**

**CB23332  
SOFTWARE ENGINEERING LAB**

**Laboratory Record Note Book**

Name : .....

Year / Branch / Section : .....

Register No. : .....

Semester : .....

Academic Year : .....

**RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)**  
**RAJALAKSHMI NAGAR, THANDALAM – 602-105**

**BONAFIDE CERTIFICATE**

**NAME:** \_\_\_\_\_ **REGISTER NO.:** \_\_\_\_\_

**ACADEMIC YEAR:** 2024-25 **SEMESTER:** III **BRANCH:** \_\_\_\_\_ B.E/B.Tech

This Certification is the bonafide record of work done by the above student in the

**CB23332-SOFTWARE ENGINEERING - Laboratory** during the year 2024 – 2025.

Signature of Faculty -in – Charge

Submitted for the Practical Examination held on \_\_\_\_\_

Internal Examiner

External Examiner

# INDEX

S.No.	Name of the Experiment	Expt. Date	Faculty Sign
1.	Preparing Problem Statement		
2.	Software Requirement Specification (SRS)		
3.	Entity-Relational Diagram		
4.	Data Flow Diagram		
5.	Use Case Diagram		
6.	Activity Diagram		
7.	State Chart Diagram		
8.	Sequence Diagram		
9.	Collaboration Diagram		
10.	Class Diagram		

## 1. PREPARING PROBLEM STATEMENT

### **Aim :**

Traditional library management systems face numerous challenges, including manual processes, lack of real-time information, and limited online functionality. Users find it difficult to track book availability, manage returns, and renew books. The absence of an efficient online renewal system leads to overdue fines and inconveniences. Additionally, physical and digital resources are not seamlessly integrated, making it harder for users to access materials. A Smart Library Management System (SLMS) is required to address these issues by automating library functions, integrating physical and digital resources, and introducing an online book renewal feature. The system will provide users with real time updates on resource availability, automated notifications for due dates, and convenient online renewal options, significantly improving user experience and operational efficiency.

### **Algorithm :**

1. Initialize system and load the library database, including both physical and digital resources.
2. User Login:
  - Allow users to log in to their account using credentials.
3. Book Search:
  - User searches for a book by title, author, or category.
  - The system displays real-time availability of the book (physical and digital copies).
4. Book Issue/Reserve:
  - If available, the user can issue or reserve the book online.
  - The system records the transaction and updates the database.
5. Online Book Renewal:
  - Users can select issued books and request an online renewal if allowed.
  - The system checks renewal eligibility and, if valid, extends the due date.
  - If the book is not renewable (e.g., reserved by another user), a notification is sent to the user.
6. Notifications:
  - Send automated notifications for due dates, successful renewals, or new arrivals.
7. Return Process:
  - When the user returns a book, the system updates the availability and transaction logs.
8. Reporting:
  - Generate reports on book usage, availability, overdue books, and system performance.

### **INPUTS FOR THE SMART LIBRARY MANAGEMENT SYSTEM:**

#### 1. User Details:

\* Username and password for login.

- Contact information for notifications.

#### 2. Book Information:

- Book title, author, ISBN, category, and availability status (physical/digital). Transaction Details:

- Book issue date, due date, and return/renewal requests.

### 3. Online Renewal Requests:

- User-initiated requests to extend the due date of borrowed books.

### 4. Notifications and Alerts:

- Inputs for automated notifications, such as due date reminders and renewal confirmations.

These elements work together to provide a seamless, user-friendly library management system.

## Stakeholder Problem Statement: Smart Library Management System

### Problem:

Current library management systems lack automation and online services, making it difficult for users to track book availability, renew borrowed books online, and access both physical and digital resources efficiently. Manual processes lead to delays, errors, and poor user experience, including missed due date reminders and overdue fines.

### Background:

The library at [Institution Name] is facing issues with outdated processes that are not user friendly or efficient. Students and staff have expressed frustration over the manual issuing and returning of books, limited digital resource integration, and the inability to renew books online. These inefficiencies affect library usage and user satisfaction.

### Relevance:

An efficient library management system is critical for academic success. Providing real time book availability, online renewal options, and automated notifications will greatly enhance the user experience. A smart system will save time, reduce errors, and make library resources more accessible to the academic community.

### Objectives:

9. Automate book issuing, returning, and tracking.

10.Enable online book renewals for user convenience.

11.Provide real-time updates on book availability (both physical and digital). 12.Send automated reminders for due dates and new arrivals.

13.Improve overall library resource management and user satisfaction.

## **Result :**

The problem statement was written successfully by following the steps described above.

## **Ex.NO. 2 : Write the software requirement specification document**

### **Aim :**

The aim of the Smart Library Management System (SLMS) is to streamline and automate core library functions, including book issuing, returning, and online renewals. The system will provide real-time updates on book availability, send automated notifications for due dates, and integrate both physical and digital resources. This will enhance the user experience and improve the overall efficiency of library operations.

### **Algorithm :**

#### **1. Introduction**

##### **1.1 Purpose**

To design and implement a **Smart Library Management System** that streamlines library operations such as book management, borrowing, and returning, while enhancing user experience through automation.

##### **1.2 Scope**

Defines the functionalities and limitations of the system, including book issuing, returning, and online renewal, real-time availability updates, and integration of physical and digital resources.

##### **1.3 Definitions, Acronyms, and Abbreviations**

A list of terms, acronyms, and abbreviations used in the document for clarity.

##### **1.4 References**

Lists any reference documents, standards, or tools used to gather requirements, such as IEEE SRS Template and API documentation for client-server systems.

##### **1.5 Overview**

A summary of the document's structure, explaining each section and its contents.

#### **2. Overall Description**

##### **2.1 Product Perspective**

Describes the context of the system, its integration with the existing library setup, and how it enhances current operations.

##### **2.2 Product Features**

Key features include:

- Book issuing, returning, and online renewal.
- Real-time updates on book availability.
- Automated notifications for due dates, renewals, and new arrivals.
- Integration of digital resources like e-books and journals.

##### **2.3 User Classes and Characteristics**

Identifies user groups such as:

- **Students and Staff:** Primary users who interact with the library system.
- **Librarians and Admins:** Manage system operations, book transactions, and user management.

##### **2.4 Operating Environment**

The system will operate in a **web-based** environment with mobile access through Android/iOS apps. It uses a cloud-hosted database and interacts with physical devices like barcode scanners and self-service kiosks.

##### **2.5 Design and Implementation Constraints**

- Compatibility with standard web browsers and mobile operating systems (Android, iOS).
- Must adhere to library management standards and data integrity requirements.

##### **2.6 Assumptions and Dependencies**

- Assumes users have basic digital literacy.
- Requires internet connectivity for full system access.
- Depends on institutional databases and digital content platforms for integrated resources.

### **3. System Features**

#### **3.1 User Account Management**

Users can create accounts, log in, and manage their profiles.

#### **3.2 Book Search and Availability Tracking**

Users can search for books and view their real-time availability.

#### **3.3 Book Issuing and Reservation**

Describes the process of issuing books and making reservations.

#### **3.4 Online Book Renewal**

Allows users to renew borrowed books online.

#### **3.5 Notifications (Due Dates, Renewals, New Arrivals)**

Automated notifications for due dates, renewal reminders, and new book arrivals.

#### **3.6 Digital Resource Integration**

Integrates digital resources like e-books and journals into the system.

#### **3.7 Reporting and Analytics**

Provides library staff with reports on usage, overdue books, and transaction statistics.

### **4. External Interface Requirements**

#### **4.1 User Interfaces**

Describes web and mobile user interface design, focusing on accessibility and ease of use.

#### **4.2 Hardware Interfaces**

Specifies the interactions with hardware such as barcode scanners and kiosks for book returns.

#### **4.3 Software Interfaces**

Details the integration with institutional databases and digital content platforms.

#### **4.4 Communication Interfaces**

Explains the communication protocols or APIs for interactions with external systems.

### **5. System Requirements**

#### **5.1 Functional Requirements**

- **FR-01:** User login and registration.
- **FR-02:** Book search and availability tracking.
- **FR-03:** Online book renewal feature.

#### **5.2 Non-Functional Requirements**

- **Performance:** System response time should be less than 2 seconds for key operations.
- **Security:** User data must be encrypted with AES-256 encryption and role-based access control.
- **Usability:** The system should have an intuitive UI for ease of use by non-technical users.
- **Reliability:** The system must have 99.9% uptime and quick recovery from failures.
- **Scalability:** The system should be able to handle a growing number of users and larger datasets.

### **6. Use Case Diagrams**

#### **6.1 Use Case 1: User Login and Registration**

Illustrates the process of user registration and login.

#### **6.2 Use Case 2: Book Search and Reservation**

Describes the flow of searching for and reserving books.

#### **6.3 Use Case 3: Book Issuing and Returning**

Explains the process of issuing and returning books.

#### **6.4 Use Case 4: Online Book Renewal**

Depicts the steps for renewing a book online.

#### **6.5 Use Case 5: Notification**

Shows how notifications for due dates, renewals, and new arrivals are triggered.

## **7. System Models**

### **7.1 Data Flow Diagram (DFD)**

A visual representation of how data moves within the system.

### **7.2 Entity-Relationship Diagram (ERD)**

Illustrates the relationships between different entities like users, books, and transactions.

### **7.3 State Chart Diagram**

Shows the different states of key entities in the system, such as "Book Issued" and "Book Returned."

### **7.4 Sequence Diagram**

Depicts the sequence of interactions between system components for key processes.

## **8. Performance Requirements**

### **8.1 Response Time**

The system should respond to key operations (like book search and renewals) within 2 seconds.

### **8.2 System Uptime**

The system should maintain an uptime of 99.9%, with acceptable downtime for maintenance.

### **8.3 Resource Handling Capacity**

The system should be capable of handling thousands of concurrent users or transactions.

## **9. Security and Privacy**

### **9.1 Authentication and Authorization**

The system will authenticate users and control access to resources using role-based access.

### **9.2 Data Encryption**

User data will be encrypted both in transit (using HTTPS) and at rest.

### **9.3 User Data Privacy**

Measures will be in place to protect user privacy and adhere to data protection regulations.

## **10. Maintenance and Support**

### **10.1 System Maintenance Requirements**

Regular updates and bug fixes to ensure the system remains functional and secure.

### **10.2 Backup and Recovery Procedures**

Regular data backups and recovery methods in case of system failure.

### **10.3 System Monitoring**

Continuous monitoring to track performance, errors, and security issues.

## **11. Appendices**

### **11.1 Glossary of Terms**

Definitions of technical terms used in the document.

### **11.2 Document References**

Lists additional documents or resources referenced in the SRS.

## **Distributed Database Diagram (PlantUML Code)**

puml

Copy code

@startuml

actor Client

node "Load Balancer" {

[Web Server]

[Mobile Server]

}

node "Data Center 1" {

database "Primary Database" as DB1 [App Server 1]



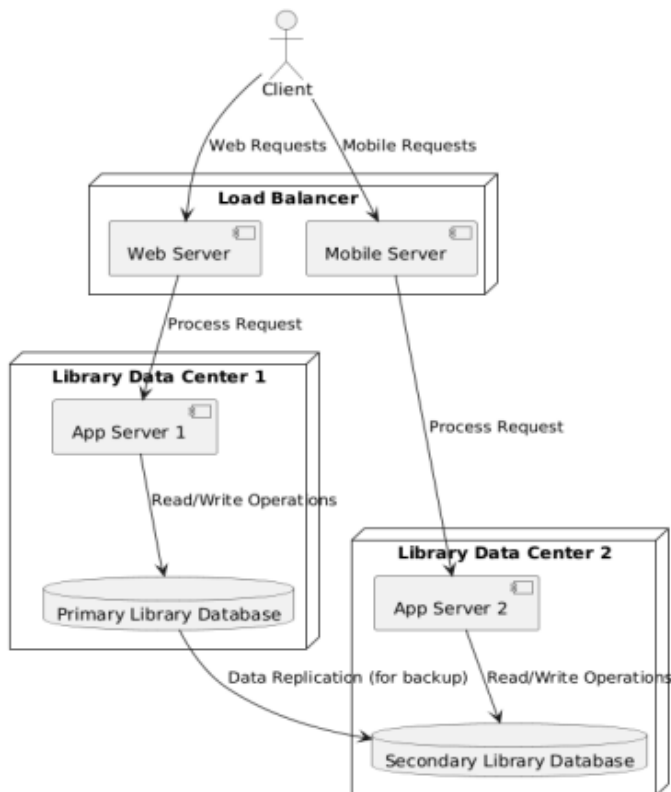
```

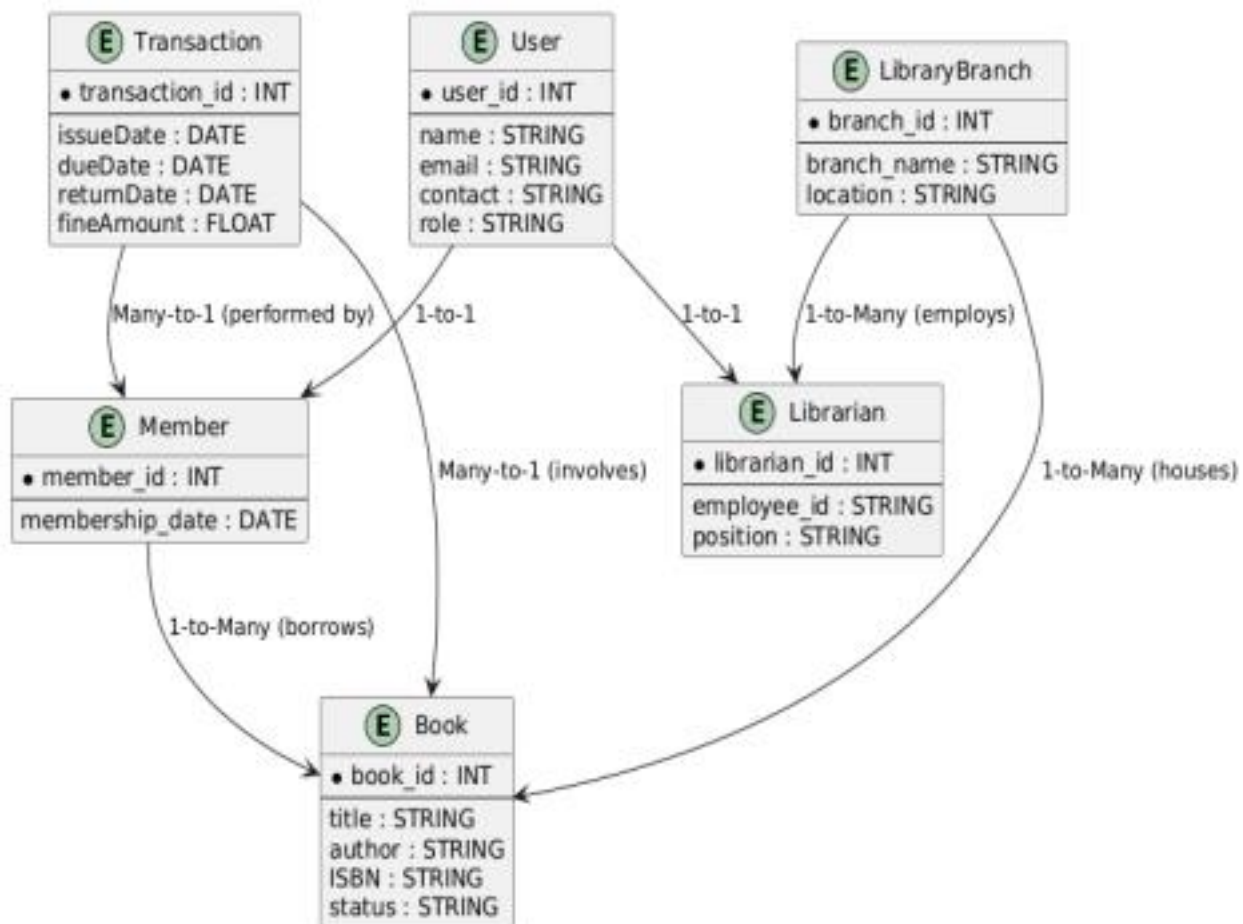
}
node "Data Center 2" {
  database "Secondary Database" as DB2 [App Server 2]
}
Client --> "Web Server" : Request
Client --> "Mobile Server" : Request
"Web Server" --> "App Server 1" : API Call
"Mobile Server" --> "App Server 2" : API Call
"App Server 1" --> DB1 : Read/Write
"App Server 2" --> DB2 : Read/Write
DB1 --> DB2 : Data Replication
@enduml

```

### Explanation:

- **Client** interacts with the system via web and mobile interfaces.
- **Load Balancer** directs requests to the appropriate servers.
- **App Servers** handle business logic and connect to **Primary** and **Secondary Databases**.
- **Data Replication** ensures high availability and keeps databases in sync.





## **Result :**

The SRS was made successfully by following the steps described above.

### 3. ENTITY RELATIONSHIP MODEL

#### **Aim :**

To Draw the Entity Relationship Diagram for smart library management system..

#### **Algorithm :**

Step 1: Mapping of Regular Entity Types

Step 2: Mapping of Weak Entity Types

Step 3: Mapping of Binary 1:1 Relation Types

Step 4: Mapping of Binary 1: N Relationship Types.

Step 5: Mapping of Binary M: N Relationship Types.

Step 6: Mapping of Multivalued attributes.

#### **Input :**

# Entities

# User

# Book

# Loan

#### **2. Entity Relationship Matrix**

Entities	User	Book	Loan
User	-	1	1
Book	M:1	-	1
Loan	M:1	M:1	-

#### **3. Primary Keys**

User: `user\_id`

Book: `book\_id`

Loan: `loan\_id`

#### **4. Attributes**

User: `user\_id`, `username`, `email`, `password`

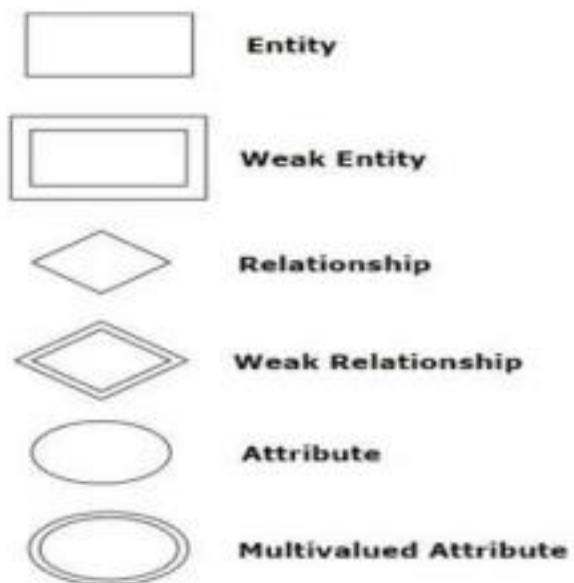
Book: `book\_id`, `title`, `author`, `isbn`, `published\_date` Loan: `loan\_id`, `user\_id`, `book\_id`,  
`loan\_date`, `return\_date`

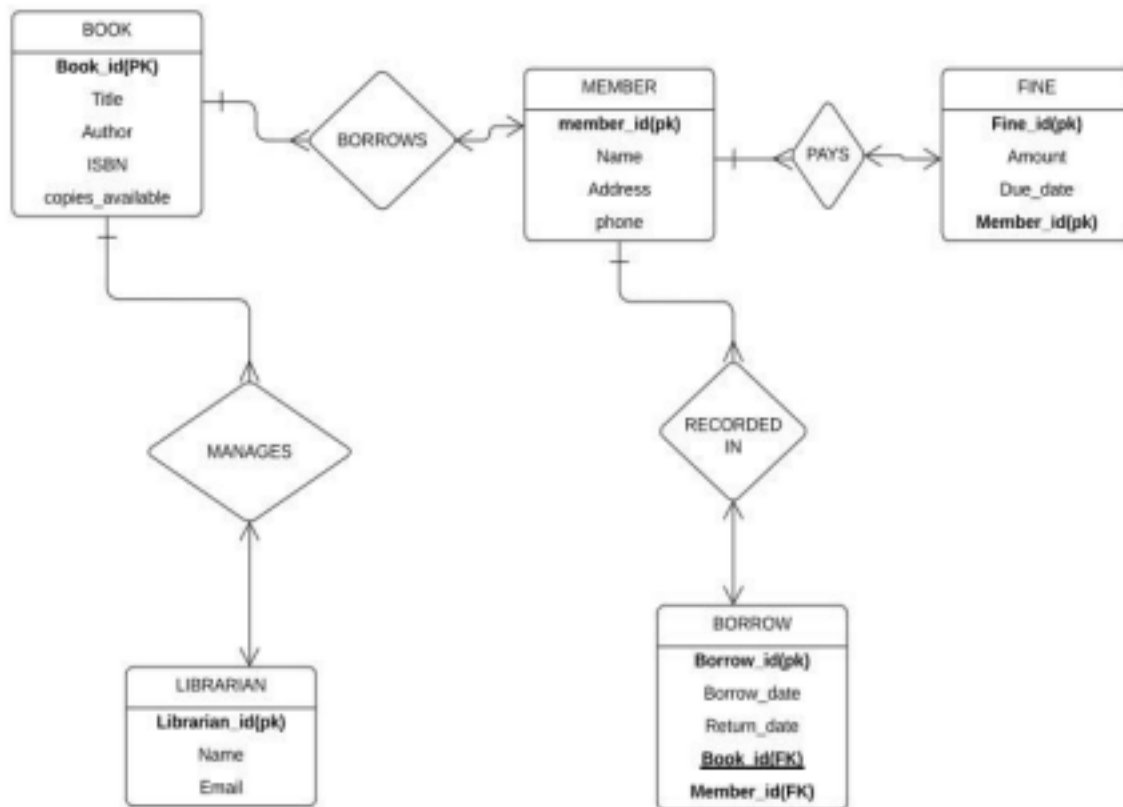
### 5. Mapping of Attributes with Entities

Entity	Relationship
User	User_id, username, email, password
Book	Book_id, title,author,isbn,published_date
Loan	Loan_id,user_id,book_id,loan_date,return_date

### ER DIAGRAM:

#### SYMBOLS:





**Result:** The entity relationship diagram was made successfully by following the steps described above.

#### 4. DATA FLOW DIAGRAM

**Aim :** To Draw the Data Flow Diagram for SMART LIBRARY MANAGEMENT SYSTEM and List the Modules in the Application.

#### Algorithm :

1. Open Visual Paradigm or Lucidchart.
2. Select a DFD Template and name it "Smart Library Management System."
3. Add External Entities: Include 'Library User' and 'Librarian.'
4. Add Processes: Create processes like Login, Book Search, Book Issuing, Book Return, Online Book Renewal, and Fine Calculation.
5. Add Data Stores: Include stores like Library Database, User Account Data Store, and Book Inventory Data Store.
6. Connect Entities, Processes, and Data Stores: Draw data flows like "Login Credentials," "Book Details," and "Issue Record."
7. Label Data Flows: Clearly label the data exchanged between entities and processes.
8. Customize: Adjust colors, fonts, and titles for clarity.
9. Export/Share: Export or share the completed DFD.



#### Input :

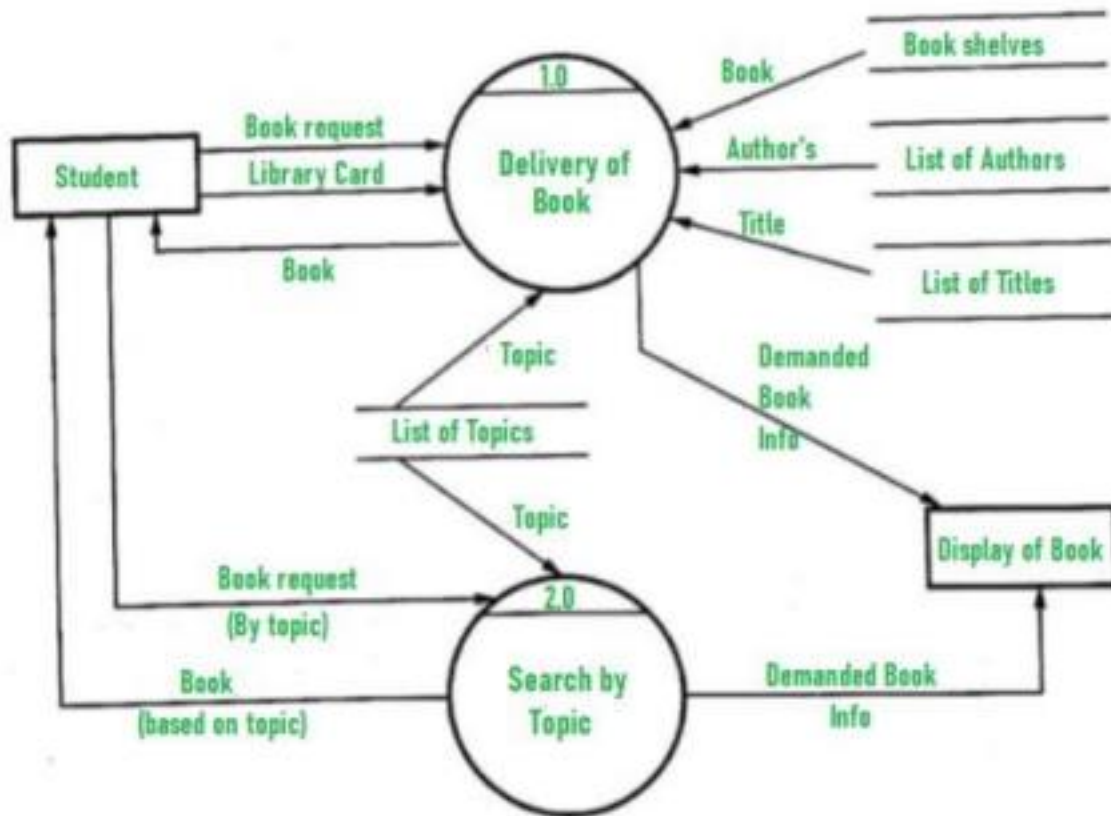
##### **Processes:**

• Login, Book Search, Book Issuing, Book Return, Online Book Renewal, Fine Calculation.

##### **Data Stores:**

Library Database, User Account Data Store, Book Inventory Data Store. • **External Entities:** Library User, Librarian.

Notation	Yourdon & De Marco	Gene & Sarson	SSADM	Unified
External Entity				
Process				
Data Store				
Data Flow				



**Level 1 DFD**

**Result:** The Data Flow diagram was made successfully by following the steps described above.

## 5. USE CASE DIAGRAM

**Aim :** To Draw the Use Case Diagram for smart library management system.

**Algorithm :**

- ☐ Identify Actors:
  - Library User
  - Librarian
  - Admin
- ☐ Identify Use Cases:
  - Login
  - Search Book
  - Issue Book
  - Return Book
  - Renew Book Online
  - Pay Fines
  - Manage Books (Librarian)
  - Manage Users (Admin)
- ☐ Connect Actors to Use Cases:
  - Library User → Login, Search Book, Issue Book, Return Book, Renew Book, Pay Fines
  - Librarian → Manage Books, Issue Book, Return Book
  - Admin → Manage Users
- ☐ Add System Boundary:
  - Label the boundary as "Smart Library Management System."
- ☐ Define Relationships:
  - Use include or extend where applicable (e.g., "Issue Book" might include "Login").
- ☐ Review and Refine:
  - Ensure all relevant actors and use cases are connected.
- ☐ Validate Diagram Accuracy:
  - Double-check for completeness and correct relationships.

**Input :**

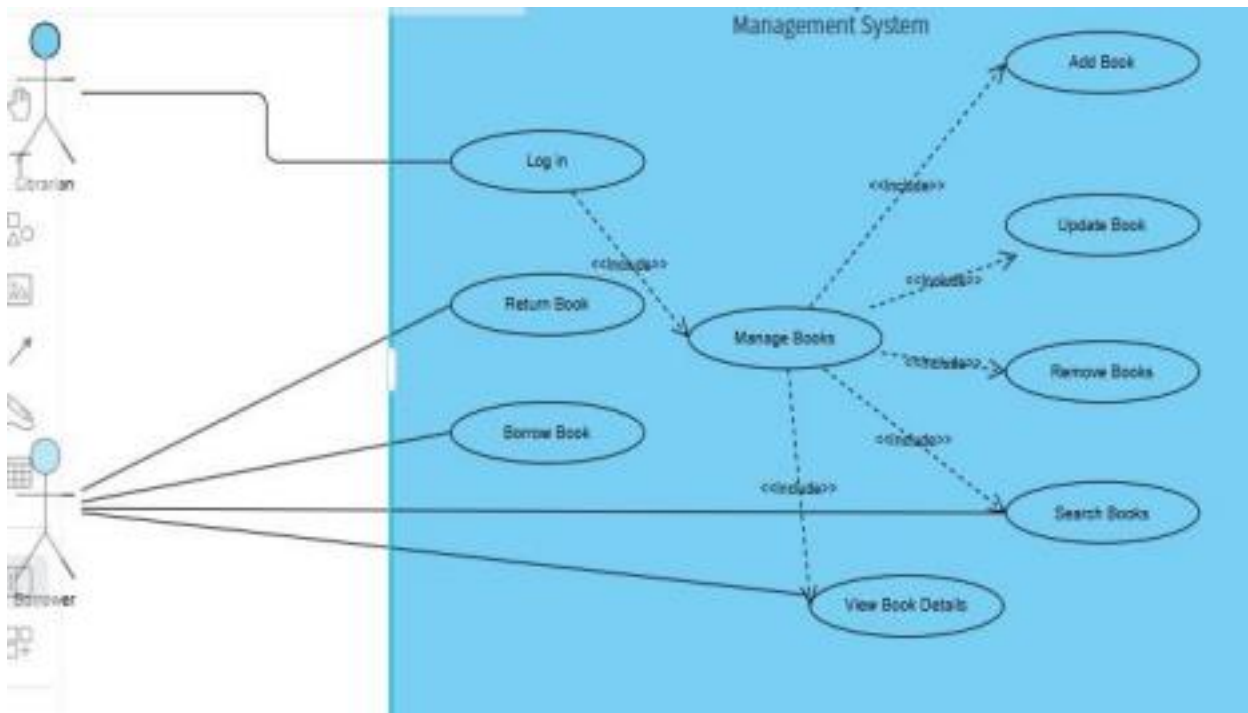
Actors

Use Cases

Relations

**Sample Output :**





### **Result :**

The use case diagram has been created successfully by following the steps given.

## 6. ACTIVITY DIAGRAM

### Aim:

To Draw the activity Diagram for SMART LIBRARY MANAGEMENT SYSTEM

### Algorithm :

- ☐ Identify Initial and Final States:
  - Initial State: System Idle / Logged Out
  - Final State: User Logs Out / System Shutdown
- ☐ Identify Intermediate Activities:
  - User Logs In
  - Search Book
  - Issue Book
  - Return Book
  - Renew Book Online
  - Pay Fines
- ☐ Identify Conditions or Constraints:
  - Successful Login
  - Book Availability
  - Valid Renewal
  - Fine Payment Completion
- ☐ Draw the Diagram:
  - Use ovals for states (e.g., Login, Search, Issue).
  - Use arrows for transitions between states.
  - Use initial state symbol (solid circle) for the start and final state symbol (circle with a border) for the end.

### Inputs :

Activities

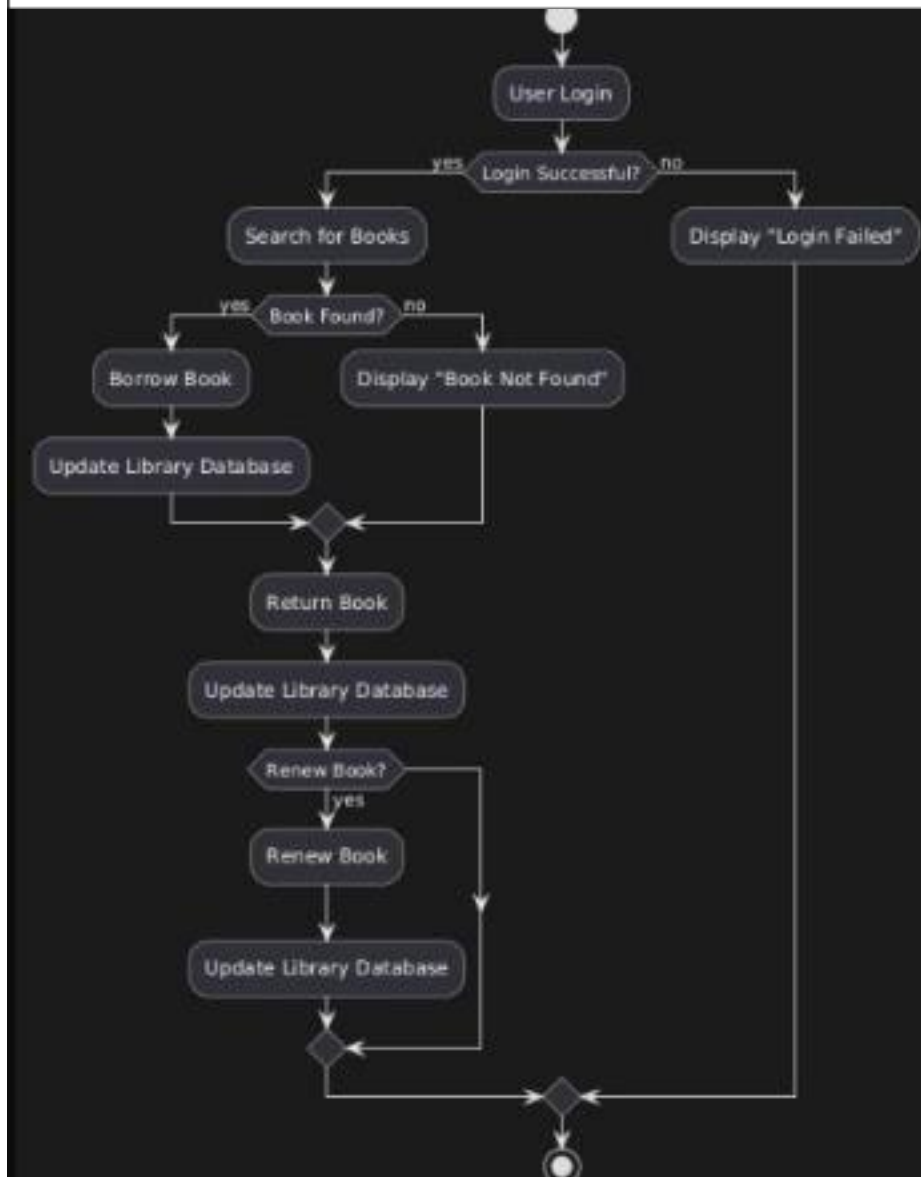
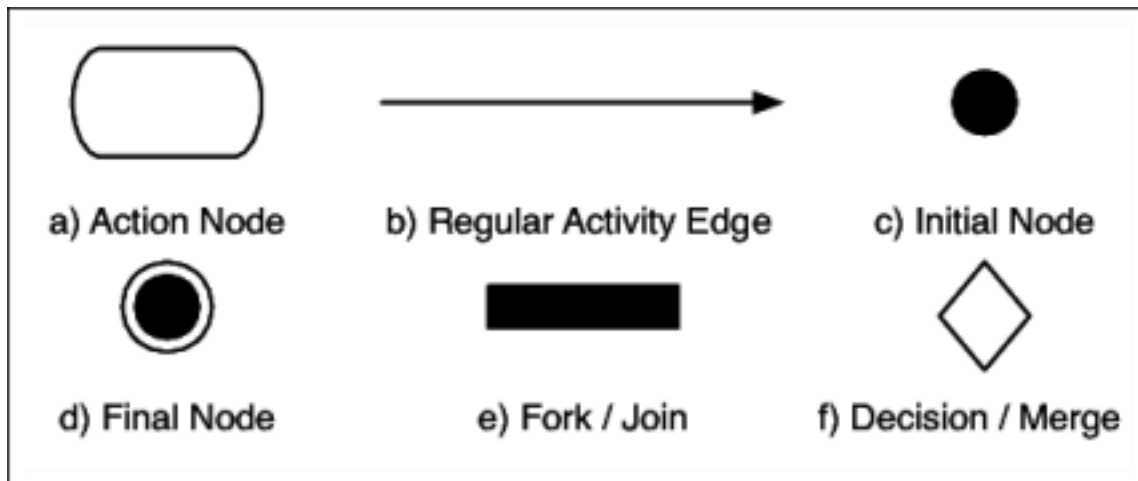
Decision Points

Guards

Parallel Activities

Conditions

### Sample Output :



**Result :** The Activity diagram has been created successfully by following the steps given.

## 7. STATE CHART DIAGRAM

### Aim :

To Draw the State Chart Diagram for SMART LIBRARY MANAGEMENT SYSTEM.

### Algorithm :

1. Identify Important Objects

- Library User, Book, Librarian, System.

2. Identify the States

- Logged Out, Logged In, Searching Book, Issuing Book, Returning Book, Renewing Book, Paying Fines.

3. Identify the Events

- User Login, Book Search, Book Issue, Book Return, Renewal Request, Fine Payment.

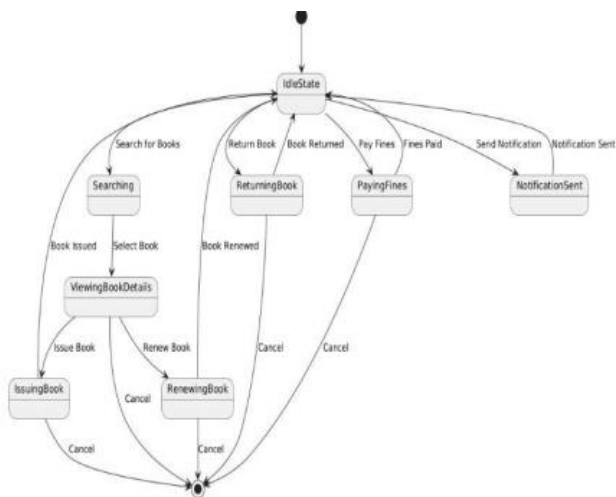
### Inputs :

Objects

States

Events

### Sample Output :



**Result :** The State Chart diagram has been created successfully by following the steps given.

## 8. SEQUENCE DIAGRAM

### **Aim :**

To Draw the Sequence Diagram for smart library management system.

### **Algorithm :**

1. **Identify the Scenario:**
  - User issuing a book.
2. **List the Participants:**
  - Library User, Librarian, System.
3. **Define Lifelines:**
  - Vertical lines for each participant.
4. **Arrange Lifelines:**
  - Place in order of interaction (User, Librarian, System).
5. **Add Activation Bars:**
  - Show active periods with vertical bars.
6. **Draw Messages:**
  - Arrows for messages exchanged (e.g., request, confirmation).
7. **Include Return Messages:**
  - Dashed arrows for responses (e.g., success, failure).
8. **Indicate Timing and Order:**
  - Messages in sequential order.
9. **Include Conditions and Loops:**
  - Use brackets for conditional logic (e.g., availability check).
10. **Consider Parallel Execution:**
  - Represent simultaneous actions if needed.
11. **Review and Refine:**
  - Check for accuracy.
12. **Add Annotations:**
  - Clarify complex interactions.
13. **Document Assumptions:**
  - Note assumptions made.
14. **Use a Tool:**
  - Create the diagram using software (e.g., Lucidchart).

### **Inputs :**

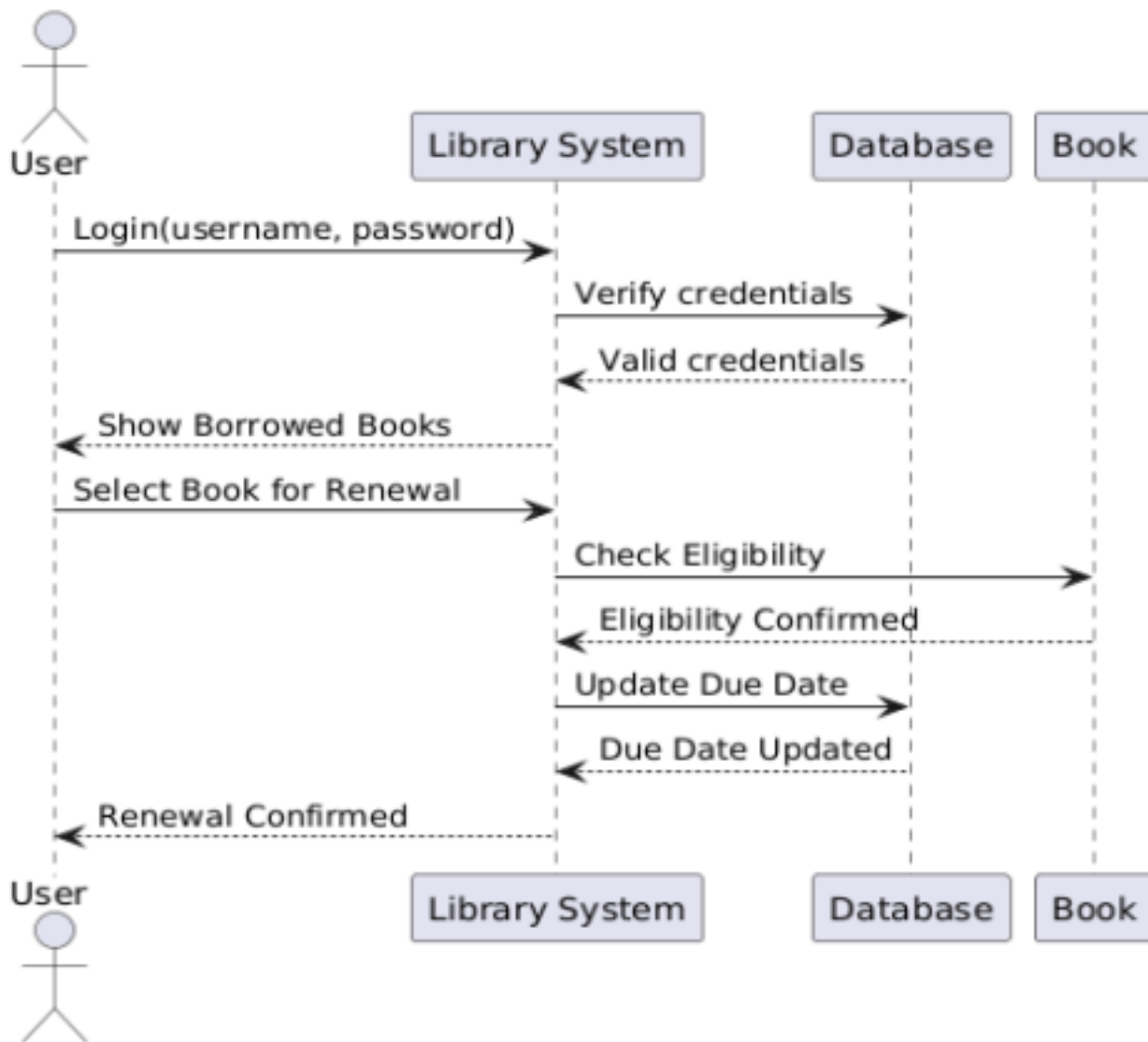
Objects taking part in the interaction.

Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

### **Sample Output :**



### **Result :**

The Sequence diagram has been created successfully by following the steps given.

## 9. COLLABORATION DIAGRAM

### **Aim :**

To Draw the Collaboration Diagram **for SMART LIBRARY MANAGEMENT SYSTEM.**

### **Algorithm :**

- ☐ Identify Objects/Participants:
  - Library User, Librarian, System.
- ☐ Define Interactions:
  - Library User requests to issue a book, Librarian processes the request, System checks book availability.
- ☐ Add Messages:
  - Messages exchanged: "Request Issue," "Confirm Availability," "Issue Book Confirmation."
- ☐ Consider Relationships:
  - Library User interacts with System for book availability, Librarian manages issuing process.
- ☐ Document the Collaboration Diagram:
  - Create a diagram showing objects and messages exchanged, with arrows indicating interaction sequence and object relationships.

### **Inputs :**

Objects taking part in the interaction.

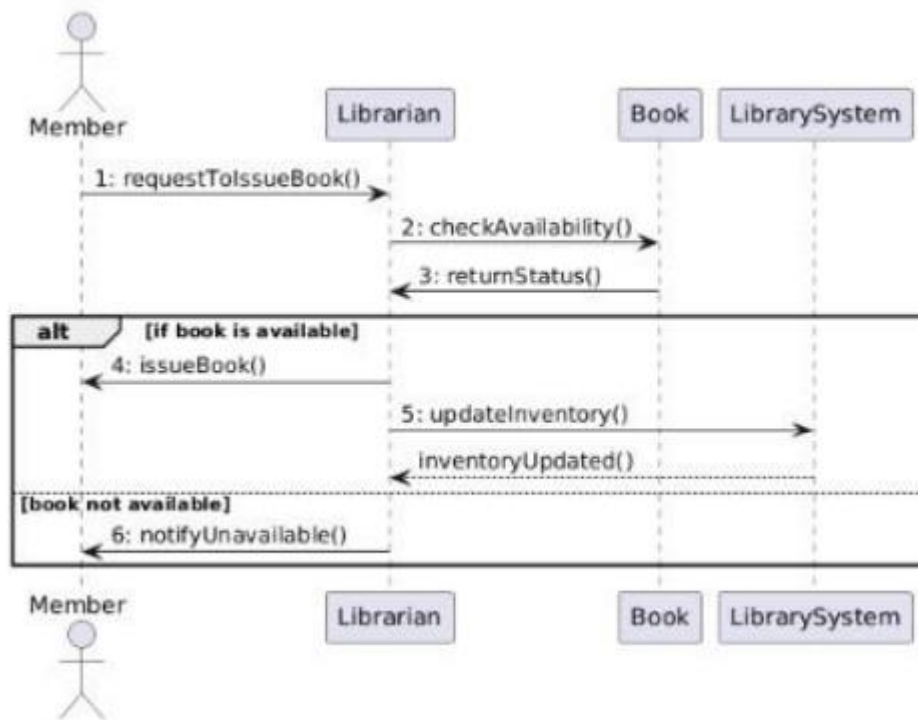
Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

### **Sample Output :**

**Collaboration Diagram for Issuing a Book**



### **Result :**

The Collaboration diagram has been created successfully by following the steps given.



## 10. CLASS DIAGRAM

### **Aim:**

To Draw the Class Diagram for SMART LIBRARY MANAGEMENT SYSTEM

### **Algorithm :**

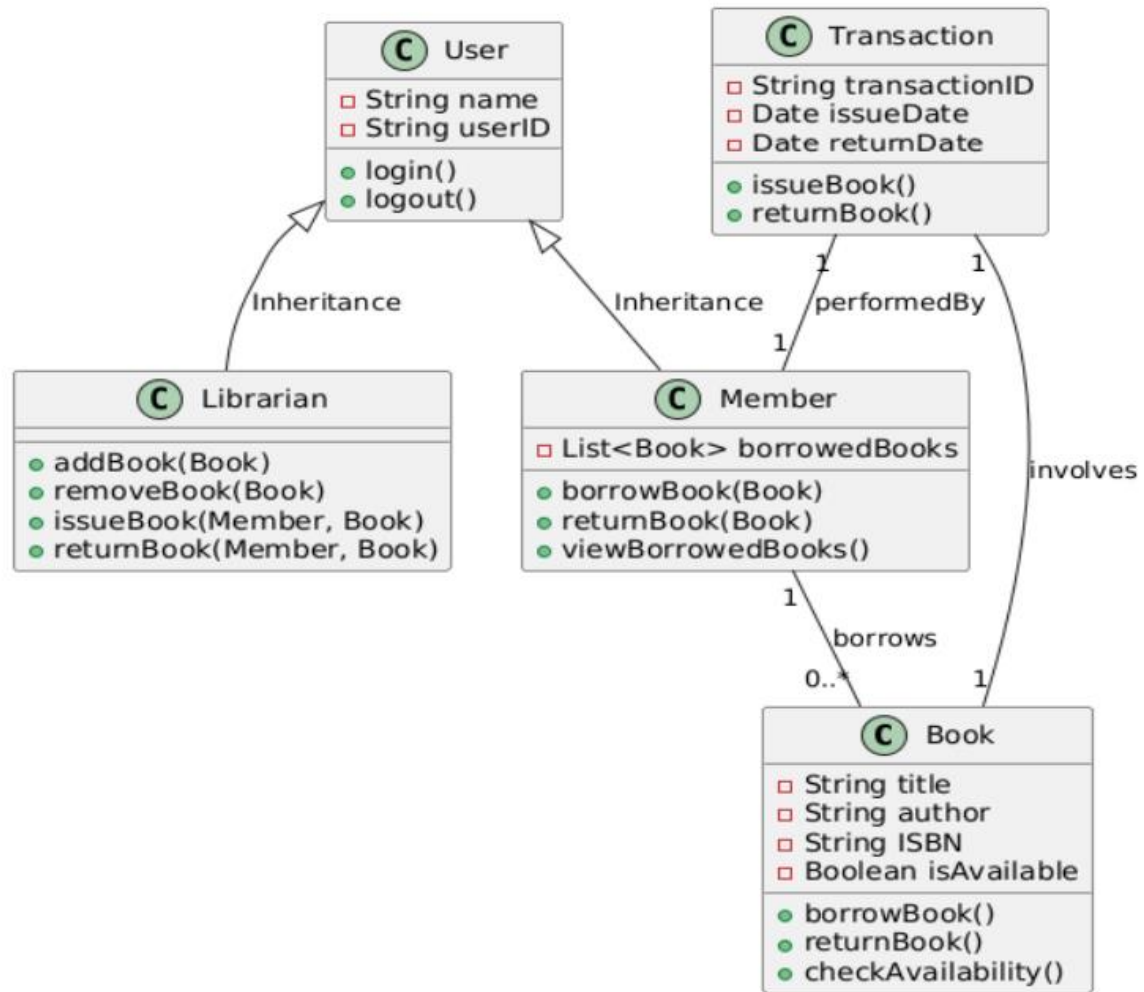
- ☐ Identify Classes:
  - Library User, Librarian, Book, Transaction, System.
- ☐ List Attributes and Methods:
  - Library User: userID, name, email; methods: login(), searchBook(), issueBook().
  - Librarian: librarianID, name; methods: manageBooks(), checkOutBook().
  - Book: bookID, title, author, availability; methods: checkAvailability(), updateStatus().
  - Transaction: transactionID, date, userID, bookID; methods: recordTransaction(), calculateFine().
- ☐ Identify Relationships:
  - Library User ↔ Transaction (a user can have many transactions).
  - Librarian ↔ Book (librarians manage books).
  - Transaction ↔ Book (each transaction involves a book).
- ☐ Create Class Boxes:
  - Draw separate boxes for each class and include their attributes and methods.
- ☐ Draw Relationships:
  - Connect classes with lines to show relationships (e.g., association).
- ☐ Label Relationships:
  - Label relationship types (e.g., one-to-many, many-to-many).
- ☐ Review and Refine:
  - Ensure accuracy and completeness.
- ☐ Use Tools for Digital Drawing:
  - Create the diagram using digital tools like Lucidchart.

### **Inputs :**

1. Class Name
2. Attributes
3. Methods
4. Visibility Notation

### **Sample Output :**

231401099



## **Result:**

The Class diagram has been created successfully by following the steps given.

## CODE FOR MINI PROJECT

### **CODE:**

### **IMPLEMENTATION CODE USING PYTHON**

```
import datetime
# Dummy data for users and books
users = {
    'user1': {'password': 'password1', 'contact': 'user1@example.com'}, 'librarian':
    {'password': 'libpass', 'contact': 'lib@example.com'}, 'admin': {'password':
    'adminpass', 'contact': 'admin@example.com'} }
books = {
    'The Great Gatsby': {'author': 'F. Scott Fitzgerald', 'available': 5, 'issued_to': []},
    '1984': {'author': 'George Orwell', 'available': 3, 'issued_to': []}, 'Python
Programming': {'author': 'John Doe', 'available': 2, 'issued_to': []} }
# Function to log in
def login(username, password):
    user = users.get(username)
    if user and user['password'] == password:
        return True
    return False
# Function to search for books
def search_book(title):
    book = books.get(title)
    if book:
        return f"Title: {title}, Author: {book['author']}, Available: {book['available']}"
    return "Book not found."
# Function to issue a book
def issue_book(title, username):
    if title in books and books[title]['available'] > 0:
        books[title]['available'] -= 1
        books[title]['issued_to'].append(username)
        print(f"{title} issued to {username}.")
    else:
        print(f"{title} is not available for issuance.")
# Function to return a book
def return_book(title, username):
    if title in books and username in books[title]['issued_to']: books[title]['available'] +=
    1
    books[title]['issued_to'].remove(username)
    print(f"{title} returned by {username}.")
    else:
        print(f"{title} was not issued to {username}.")
# Function to renew a book
```

```

def renew_book(title, username):
    if title in books and username in books[title]['issued_to']:
        print(f'{title} renewed for {username}.')
    else:
        print(f'{title} cannot be renewed as it was not issued to {username}.')
# Function to send notifications
def send_notification(username):
    user_contact = users[username]['contact']
    print(f'Notification sent to {user_contact}: Please return your books on time.')
# Main program
def main():
    print('Welcome to the Smart Library Management System')
    username = input('Enter username: ')
    password = input('Enter password: ')
    if login(username, password):
        print('Login successful!')
        while True:
            print("\nMenu:")
            print('1. Search Book')
            print('2. Issue Book')
            print('3. Return Book')
            print('4. Renew Book')
            print('5. Send Notification')
            print('6. Exit')
            choice = input('Choose an option: ')
            if choice == '1':
                title = input('Enter book title to search: ') print(search_book(title))
            elif choice == '2':
                title = input('Enter book title to issue: ') issue_book(title, username)
            elif choice == '3':
                title = input('Enter book title to return: ') return_book(title, username)
            elif choice == '4':
                title = input('Enter book title to renew: ') renew_book(title, username)
            elif choice == '5':
                send_notification(username)
            elif choice == '6':
                print('Exiting system.')
                break
            else:
                print('Invalid choice. Please try again.')
        else:
            print('Invalid username or password.')
if __name__ == "__main__":
    main()

```

#### **INPUT:**

**Enter user name:**

**Enter password:**

**OUTPUT:**

**Enter user name:Nalini Enter password:rec  
Valid user name or password..**