Practical no:1

Aim: write down the problem statement for a Library Management System

Problem Statement: Library Management System

The current library management process at SCCT is largely manual and paper-based, leading to several inefficiencies, including:

Time-consuming book tracking and inventory management.

Difficulty in tracking borrowed and returned books.

Increased likelihood of errors in maintaining book records and user transactions.

Limited access to real-time information for administrators, library staff, and users.

Challenges in generating detailed reports and analysis of library usage.

With an increasing number of books and frequent transactions, the library faces difficulties in maintaining an up-to-date inventory, meeting demand, and ensuring smooth operations. To address these issues, SCCT requires a Library Management System (LMS) that can automate core processes such as book tracking, issue/return management, and reporting, while providing user-friendly access to stakeholders.

Proposed Solution:

The LMS will offer the following features:

Automated Book Inventory Management:

Simplify book cataloging and inventory tracking with automated addition, deletion, and updating of book records.

User Management:

Maintain detailed profiles for each library user (students, faculty, and staff) including borrowing history, fines, and account status.

Book Borrowing and Return System:

Streamline the book borrowing process with automated tracking of borrowed books, due dates, and overdue notifications.

Search and Reservation System:

Provide an efficient search feature for users to find books by title, author, or category and allow reservations for books that are currently unavailable.

Notifications and Alerts:

Send automated email/SMS alerts for due dates, overdue books, and library updates.

Reporting and Analytics:

Generate detailed reports on library usage, popular books, overdue books, and user activity for administrators and library staff.

Constraints:

Budget: The system must be developed within a budget of [insert budget].

Timeline: The project must be completed within [insert timeline] months.

Scalability: The system should accommodate future increases in library resources, such as new books, and potential integration with other educational tools.

User Access: Ensure role-based access for library administrators, staff, and users (students and faculty).

Conclusion:

By implementing this system, SCCT aims to:

Improve operational efficiency in managing library resources.

Reduce the workload and errors associated with manual library management processes.

Enhance user satisfaction with quick and transparent access to library resources.

Facilitate better tracking of book loans and user activity, ensuring timely returns and optimal resource allocation.

Provide a scalable foundation for future library growth and system enhancements.

Practical no:2

Aim: perform requirement analysis and develop software requirement specifications sheet (srs) for library management system

. Introduction

1.1 Purpose

This document outlines the functional and non-functional requirements for the Library Management System (LMS). The system will allow administrators, librarians, and users (students, staff, and faculty) to manage books, track transactions, and generate reports. The goal is to automate key processes and provide a streamlined experience for managing library resources.

1.2 Scope

The system will manage various aspects of a library, including book inventory management, user management, book borrowing and returns, notifications, and report generation. It will provide user-friendly interfaces for all stakeholders: administrators, librarians, and library users. It will be web-based and accessible through modern browsers.

1.3 Definitions, Acronyms, and Abbreviations

LMS: Library Management System

Admin: System user responsible for managing library operations, including user management and reporting.

Librarian: A user who manages book transactions and resources.

User: A student, staff, or faculty member who can borrow and return books.

Book Inventory: A collection of books and resources stored in the system.

1.4 References

IEEE Software Requirements Specification (SRS) Standard

Institution's library policies and procedures

2. Overall Description

2.1 Product Perspective

The Library Management System (LMS) will be a web-based application that interacts with a database to store and retrieve book and user data. It will provide different interfaces for administrators, librarians, and users. The system will focus on providing seamless access to library resources and efficient management of transactions.

2.2 Product Features

User Management: Admins and librarians can manage users (students, faculty, staff).

Book Inventory: Admins and librarians can add, update, or remove books from the inventory.

Book Borrowing/Returning: Users can borrow and return books, with automated tracking of due dates and overdue books.

Search and Reservation: Users can search for books by title, author, or category and reserve unavailable books.

Notifications: Automated notifications for due dates, overdue books, and new book arrivals.

Report Generation: Admins can generate reports on library usage, popular books, overdue books, and user activity.

2.3 User Classes and Characteristics

Admin: Full access to the system, including user management, book management, and reporting.

Librarian: Manages day-to-day library operations, including issuing books, returns, and resource management.

User (Student, Faculty, Staff): Can borrow and return books, view current loans, and receive notifications.

2.4 Operating Environment

Web-based application that can be accessed through modern browsers (Chrome, Firefox, etc.).

Server-side technologies: Python/Django, Node.js, or Java-based frameworks.

Database: MySQL, PostgreSQL, or any relational database.

2.5 Design and Implementation Constraints

The system must comply with institutional data privacy regulations.

The system should be scalable to support future growth in library resources and users.

User authentication and role-based access control will be implemented to ensure data security.

3. System Features

3.1 User Authentication

3.1.1 Description

The system will authenticate users via a login system to ensure proper access control based on roles.

3.1.2 Functional Requirements

Users must log in with a valid username and password.

Admins can reset passwords for any user.

Roles (admin, librarian, user) determine system access levels.

3.1.3 Non-Functional Requirements

The login process should take no more than 3 seconds.

Passwords must be encrypted (e.g., bcrypt or similar encryption methods).

3.2 Book Inventory Management

3.2.1 Description

Admins and librarians will manage the library's book inventory, including adding new books, updating existing ones, and removing outdated books.

3.2.2 Functional Requirements

Admins and librarians can add new books to the inventory.

Books can be updated or removed from the system.

Each book will have a unique ID, title, author, category, and availability status.

3.2.3 Non-Functional Requirements

Book data should be validated (e.g., non-empty fields for title, author, etc.).

The book management interface should be responsive on both desktop and mobile platforms.

3.3 Book Borrowing and Returning

3.3.1 Description

Users can borrow and return books, with automated tracking of borrowed books and due dates.

3.3.2 Functional Requirements

Users can borrow books, and the system tracks the borrowing date and due date.

Users can return books, and the system updates their account and the book's availability status.

The system should alert users for overdue books and send reminders for returns.

3.3.3 Non-Functional Requirements

Borrowing and returning actions should be processed in under 2 seconds.

Users should receive email/SMS notifications for overdue books.

3.4 Report Generation

3.4.1 Description

Admins can generate detailed reports on library usage, including overdue books, user activity, and book availability.

3.4.2 Functional Requirements

Admins can generate reports by user, book category, overdue books, and popular books.

Reports can be exported to PDF, CSV, or Excel formats.

3.4.3 Non-Functional Requirements

Report generation should be completed in under 5 seconds.

4. External Interface Requirements

4.1 User Interfaces

Admin Dashboard: Allows admins to manage users, books, and generate reports.

Librarian Dashboard: Enables librarians to manage book transactions and the inventory.

User Dashboard: Allows users to view borrowed books, search for books, and manage reservations.

4.2 Hardware Interfaces

The system should be operable on standard PCs, tablets, and mobile devices.

The database should run on standard server hardware.

4.3 Software Interfaces

The system will use a relational database like MySQL or PostgreSQL for storing data.

Web application development frameworks like Django (Python) or Express (Node.js) will be used.

5. System Features

5.1 Database Design

Entities: Book, User, Transaction, Category, Report.

Relationships: Users borrow books, books belong to categories, transactions track borrowing and returning of books.

Tables: Users, Books, Transactions, Categories, Reports.

5.2 System Security

Role-based access control will be implemented to ensure that each user type (admin, librarian, user) has appropriate permissions.

Sensitive data (e.g., user details) will be encrypted and stored securely.

System logs will be periodically audited to detect any unusual activity.

6. Non-Functional Requirements

6.1 Performance Requirements

The system should handle up to 1,000 concurrent users.

Page load times for searches and transactions should be under 3 seconds.

6.2 Reliability

The system must have a 99% uptime guarantee.

The system should be able to recover from failures within 10 minutes.

6.3 Scalability

The system should scale to support an increase in the number of books, users, and transactions.

6.4 Security

The system must comply with industry-standard security practices, including data encryption and secure login mechanisms.

6.5 Maintainability

The system should be modular and easy to maintain, with well-documented code and a clear change management process.

6.6 Usability

The system should be user-friendly with a responsive design that works on desktops, tablets, and mobile devices.

It should have intuitive navigation and minimal user training requirements.

7. Appendix

Detailed Database Schema Diagrams

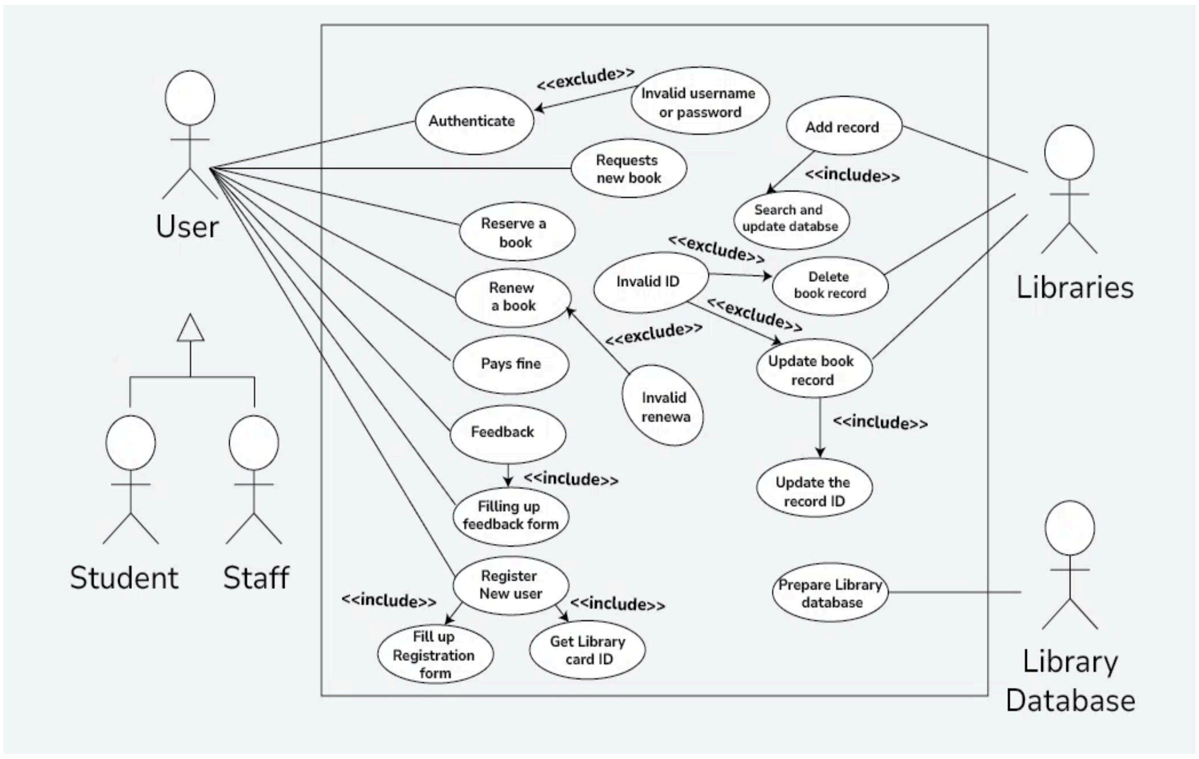
User Interface Wireframes and Mockups

Flowcharts for Key Processes (e.g., book borrowing, book return, report generation)

Practical no: 3

Aim: Draw the user's view analysis for the suggested system: Use case diagram.

Library management system



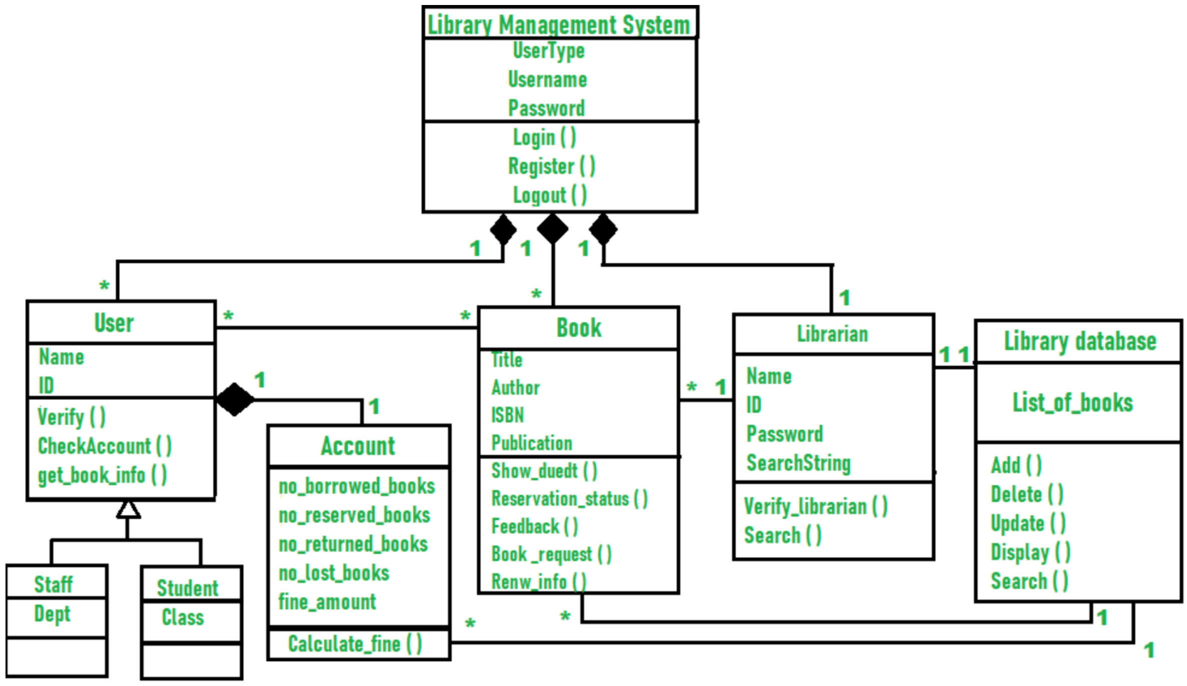
Key Elements in the Diagram:

1. **Actors:**
   - **User** (Students & Staff)
   - **Libraries** (Manage books)
   - **Library Database** (Stores book & user data)
2. **Use Cases (Actions):**
   - **User Actions:** Authenticate, reserve/renew books, pay fines, give feedback, register.
   - **Library Actions:** Add, update, or delete book records.
   - **Database Actions:** Manage book and user records.
3. **Relationships:**
   - **<<include>>**: Must happen (e.g., Register includes filling a form).
   - **<<exclude>>**: Stops action if invalid (e.g., Wrong ID prevents update).

Practical no: 4

Aim:Draw structural view diagram for the system: Class diagram, object diagram.

a]Class diagram



A **Class Diagram** represents the system's structure, defining its **classes, attributes, methods, and relationships**.

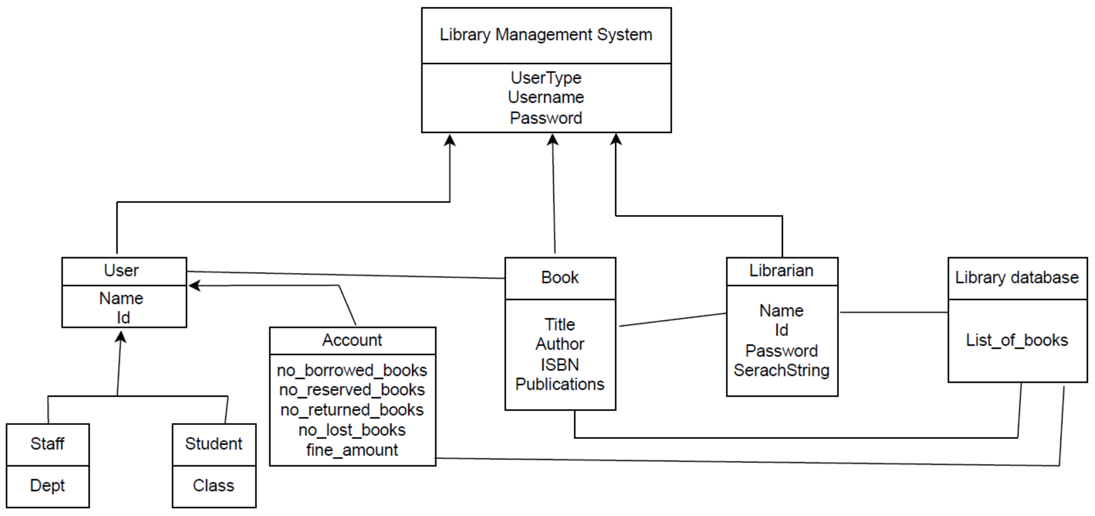
Key Classes & Functions

1. **Library Management System** – Manages users, login, register, and logout.
2. **User (Student/Staff)** – Has attributes (Name, ID) and methods (Verify(), CheckAccount()).
3. **Account** – Tracks borrowed books, fines, and calculates penalties.
4. **Book** – Stores details (Title, Author, ISBN) and functions like Reservation_status(), Book_request().
5. **Librarian** – Manages book search, verification.
6. **Library Database** – Stores book records and handles add, delete, update, and search.

Relationships

- Users interact with **Books, Account, and Librarian**.
- **Librarian** manages books.
- **Library Database** maintains all books.

b] object diagram:-



This **Object Diagram** represents a **Library Management System** and its relationships:

- **Library Management System**: Handles user authentication (`UserType`, `Username`, `Password`).
- **User**: Can be **Staff** (`Dept`) or **Student** (`Class`), identified by `Name` and `Id`.
- **Account**: Tracks user activity (borrowed, reserved, returned, lost books, fines).
- **Book**: Has details like `Title`, `Author`, `ISBN`, and `Publications`.
- **Librarian**: Manages books and the library database.
- **Library Database**: Stores the `List_of_books`.

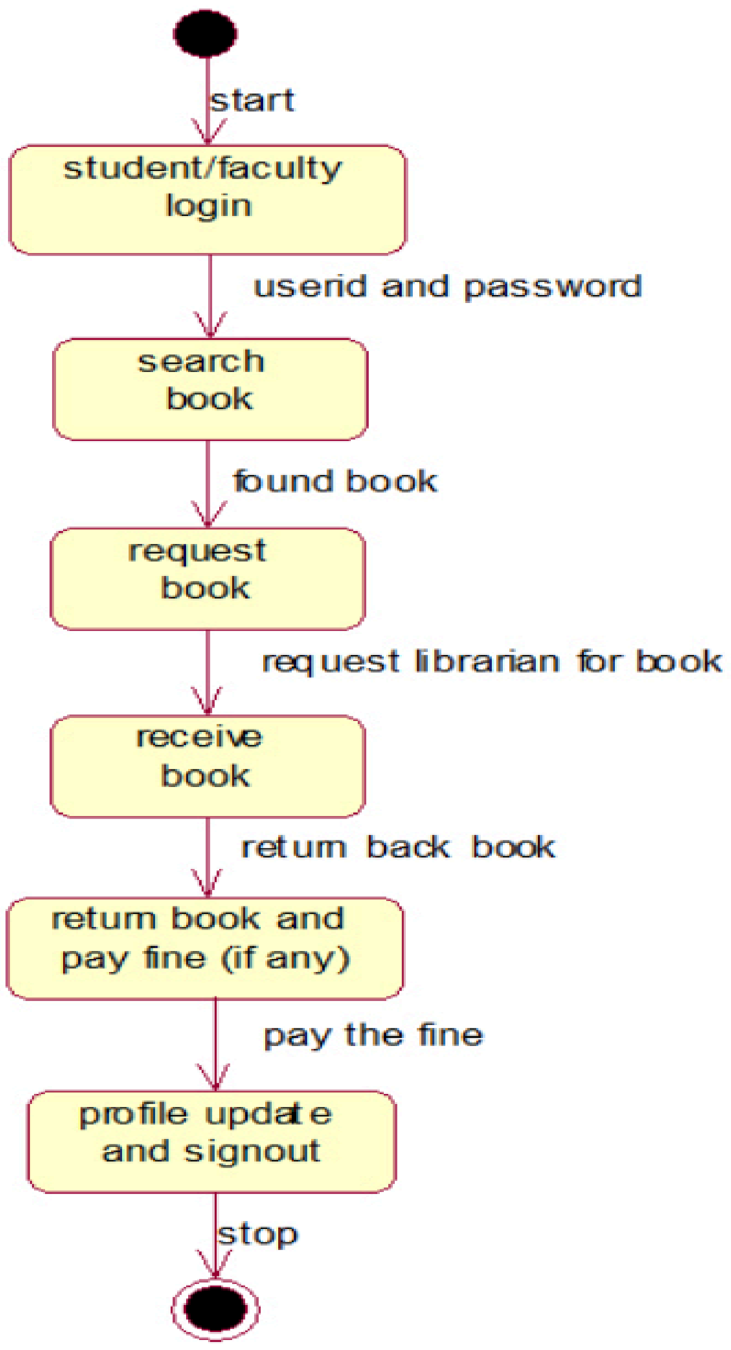
Relationships:

- Users have **Accounts**.
- Librarians manage **Books** and the **Library Database**.
- The system connects users, librarians, and books.

Practical no:5

AIM: Draw the behavioral view diagram : State-chart diagram, Activity diagram

1] State-chart diagram

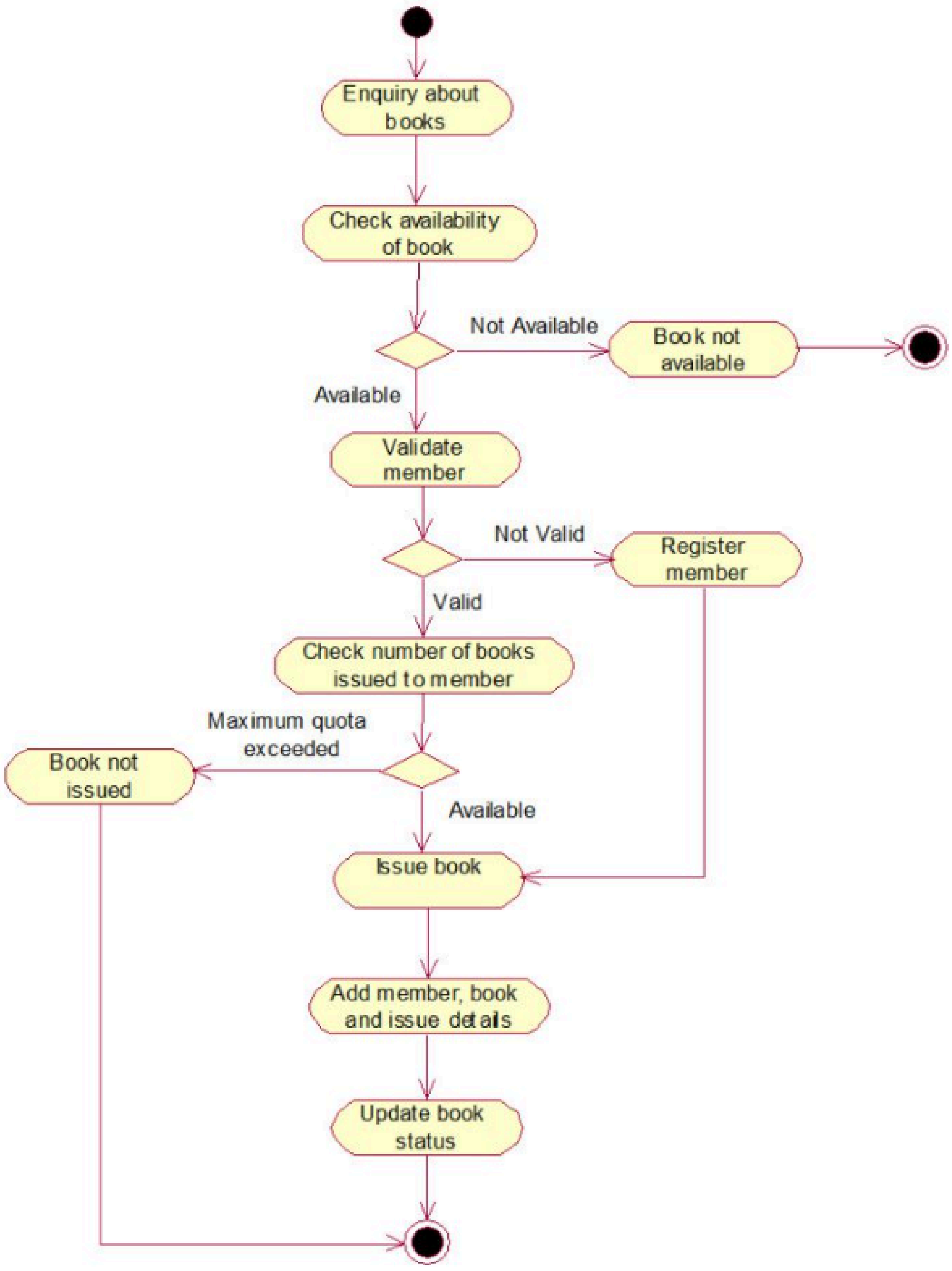


State Chart Diagram Explanation (Minimal Words):

1. **Start** → User logs in (Student/Faculty).
2. **Search Book** → If found, request it.
3. **Request Book** → Ask librarian.
4. **Receive Book** → Borrow the book.
5. **Return Book** → Pay fine if required.
6. **Profile Update & Signout** → End process.

Flow: Login → Search → Request → Receive → Return → Pay Fine → Logout.

2] Activity diagram



1. **Start** → User inquires about books.
2. **Check Availability** →
   - If **not available** → "Book not available" → **End**
   - If **available** → Proceed
3. **Validate Member** →
   - If **not valid** → Register member
   - If **valid** → Proceed
4. **Check Issued Books Limit** →
   - If **quota exceeded** → "Book not issued" → **End**
   - If **within limit** → Proceed
5. **Issue Book** → Add member, book, and issue details
6. **Update Book Status** → **End**

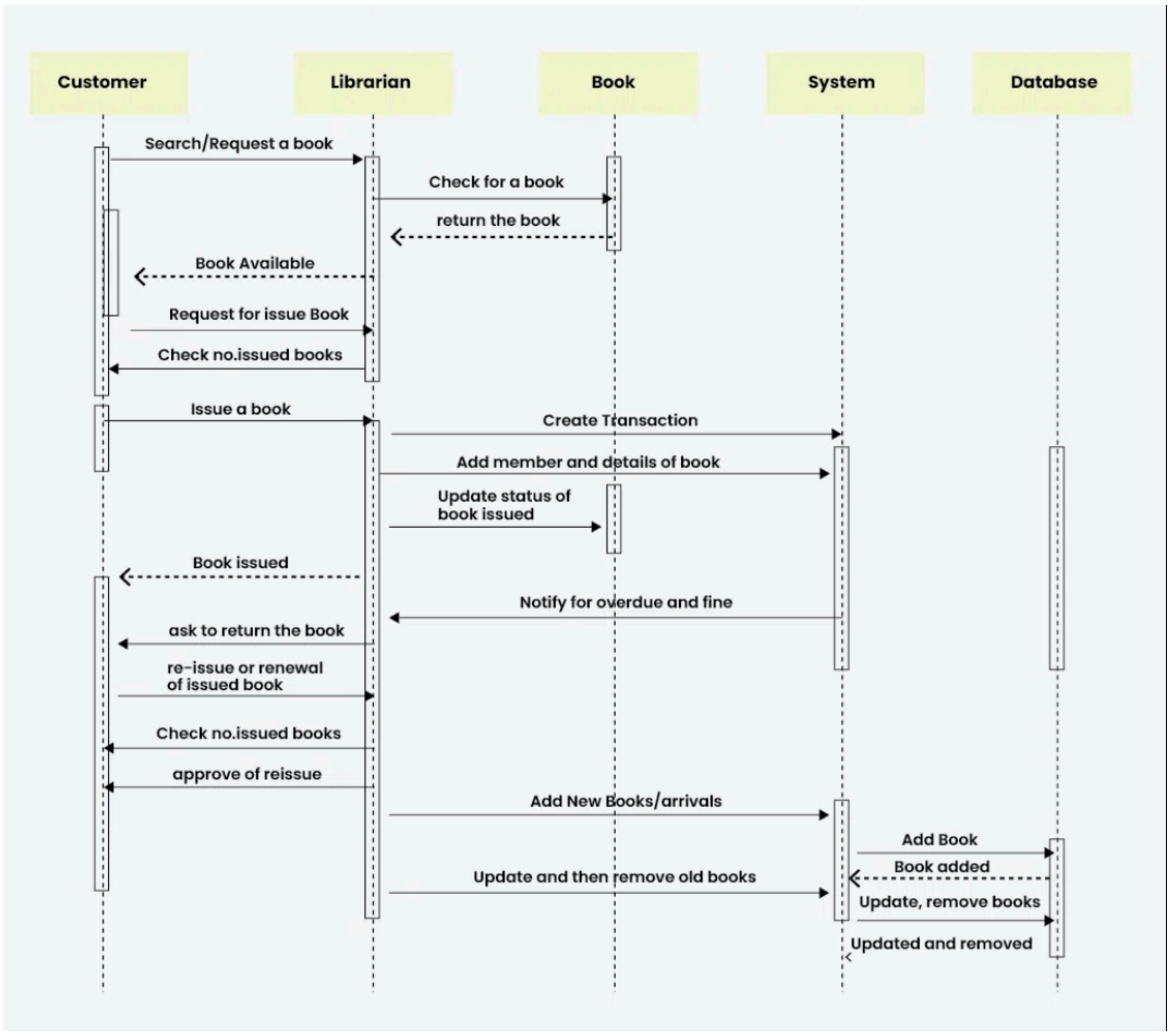
Key Flow:

Inquiry → Availability → Member Validation → Issue Check → Issue Book → Update Status → End

Practical No: 6

Aim:Draw the behavioral view diagram for the suggested system: Sequence diagram, Collaboration diagram

1]Sequence diagra**m**

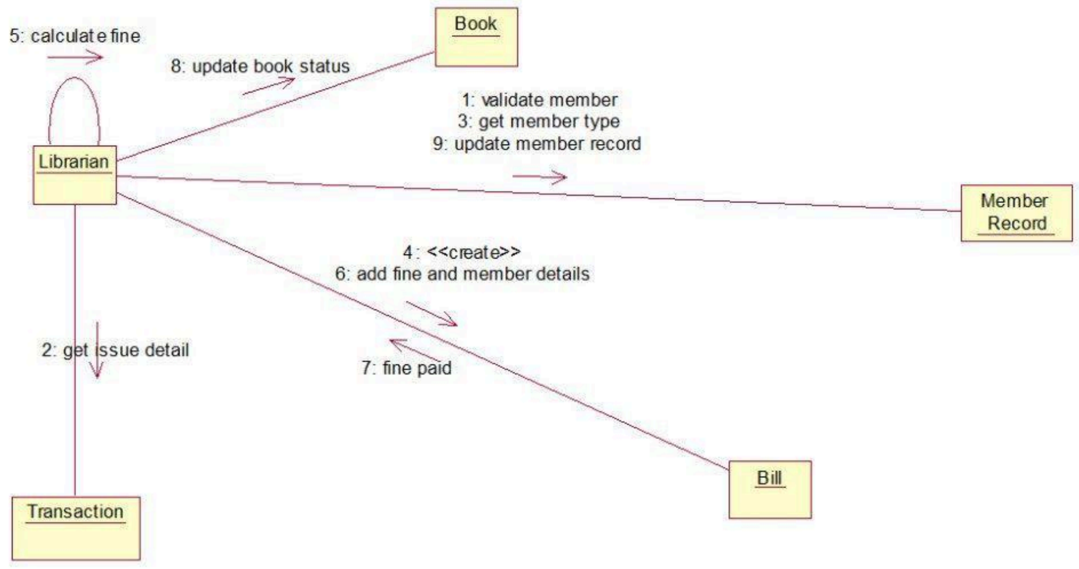


1. **Book Request:**
    - The customer searches or requests a book.
    - The librarian checks the book's availability.
    - If available, the librarian proceeds with issuing the book.
2. **Book Issuance Process:**
    - The librarian checks the number of books already issued to the customer.
    - If within the limit, the book is issued, and transaction details are updated in the system.
3. **Book Return & Renewal:**
    - The customer requests a return or renewal.

- The librarian checks issued book records and approves renewal if applicable.

4. **Fine Notification:**
   - The system tracks overdue books and sends notifications for fines.
5. **Book Management:**
   - New books are added to the system.
   - Old books are updated or removed from the database.

2]collaboration diagram



A **Collaboration Diagram** represents object interactions with a focus on message exchange. It highlights relationships and the flow of operations between objects in a **Library Management System (LMS).**

Key Interactions in the Diagram

1. **Validate Member:**
    - The **Librarian** verifies the **member's identity** using the **Book** system.
    - Retrieves the **member type** and updates the **Member Record** if necessary.
2. **Issue Details Retrieval:**
    - The **Librarian** fetches **book issue details** from the **Transaction** system.
3. **Fine Calculation:**
    - If a book is overdue, the **Librarian** calculates the **fine** for the late return.
4. **Bill Creation:**
    - A **new bill** is generated using the **Bill** system, linking the fine with the **member details.**
5. **Fine Payment:**
    - The **member pays the fine**, updating the **bill status.**
6. **Update Records:**
    - After fine payment, the **Librarian updates the book status** in the **Book** system.
    - The **Member Record** is also updated with new transaction details.