Library Management System

Scenario

Design a Library Management System (LMS) that helps librarians and members manage books and transactions efficiently. The system should include the following functionalities:

1. Book Management:

- Librarians can add, update, or remove books with details like book ID, title, author, price, rack number, status (available/issued), edition, and purchase date.
- Use generalization to represent different types of books such as Journals, Magazines and Study Books.
- Both librarians and members can search for books by title, author, keywords, or ID. The system should display detailed book information, including real-time availability status.
- When books are issued or returned, the system updates the book's status accordingly. If a book is unavailable, it should suggest alternative books.

2. Member Management:

- Librarians can register new members by capturing details like member ID, name, contact information, membership date, and type (student, faculty, etc.).
- Use generalization to represent different types of members such as Student and Faculty.
- Members can log in to borrow books. The system verifies their credentials and tracks the number of books each member has borrowed, preventing further borrowing if they exceed their limit.
- Librarians can update or retrieve member information using member ID or other relevant details.

3. Transaction Management:

- Librarians can issue books to members, recording the transaction ID, member ID, book ID, issue date, and due date. The system updates the book's status to 'issued' and tracks the member's current borrowed books.
- When a book is returned, the system updates the book's status to 'available' and adjusts the member's borrowed count.
- If the book is returned late, the system automatically calculates overdue fines and generates a bill.
- All transactions (book issues, returns, and fines) are recorded and can be retrieved as needed. Authorized users can delete transaction records when necessary.

4. Billing Management:

- The system generates bills for overdue fines, including details like bill number, date, member ID, and amount.
- Members can pay their fines, and the system updates the bill status to reflect payment.

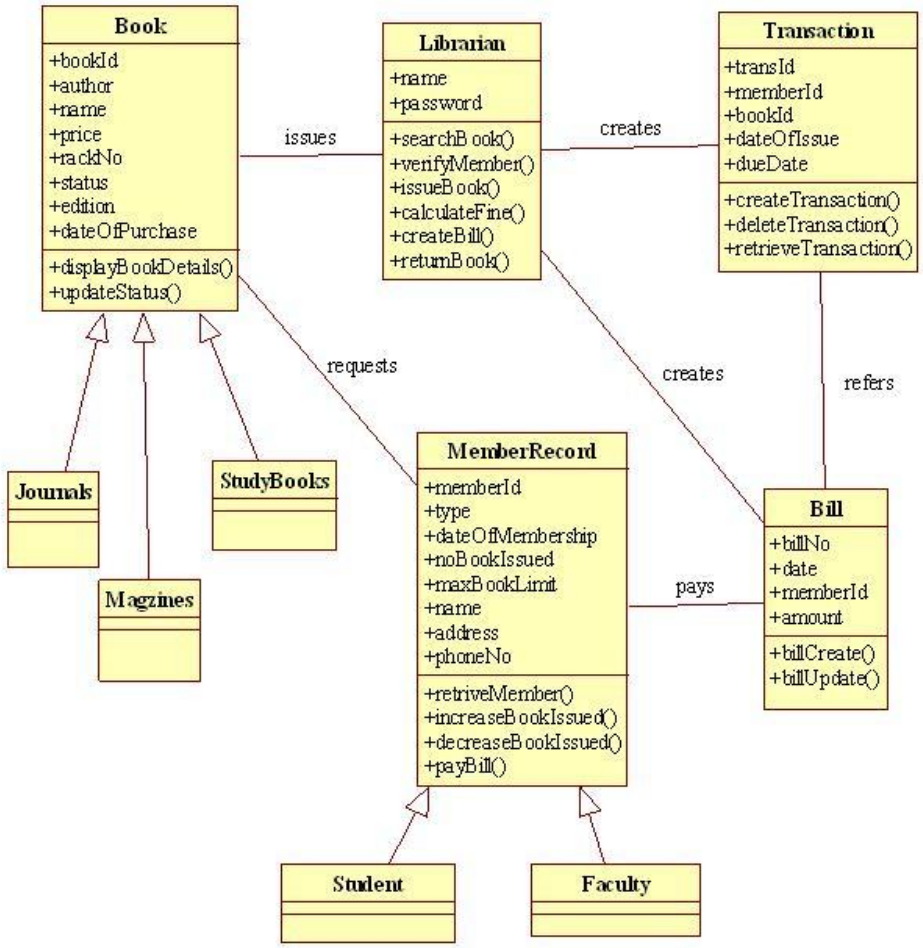
UML Diagrams with plantUML codes

1) Class Diagram

PlantUML code:

```
@startuml


class Book {
  - bookID: int
  - title: string
  - author: string
  - price: float
  - rackNumber: int
  - status: string
  - edition: string
  - purchaseDate: date
}

class Journal
class Magazine
class StudyBook

Book <|-- Journal
Book <|-- Magazine
Book <|-- StudyBook

class Member {
  - memberID: int
  - name: string
  - contactInfo: string
  - membershipDate: date
  # getBooksBorrowed(): int
}

class Student
class Faculty

Member <|-- Student
Member <|-- Faculty


Book "1" -- "*" Member : Currently Borrowed

@enduml
```

Diagram:



2) Sequence Diagram

PlantUML code:

@startuml

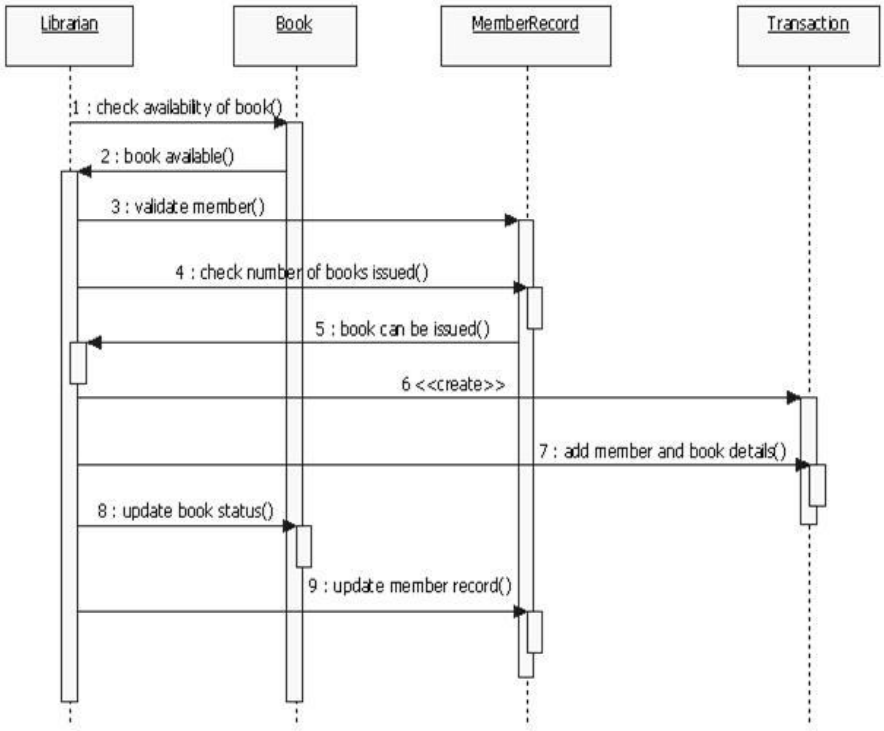
participant Librarian
participant Member
participant Book

Librarian -> Book: Add Book
Book --> Librarian: Book Added

Member -> Book: Issue Book
Book --> Member: Book Issued
Book -> Member: Return Book
Member --> Book: Book Returned

@enduml

Diagram:



3) Use Case Diagram

PlantUML code:

@startuml

actor Librarian as L
actor Member as M

rectangle LibraryManagementSystem {
  usecase "Add Book"
  usecase "Update Book"
  usecase "Remove Book"
  usecase "Search Book"
  usecase "Issue Book"
  usecase "Return Book"
  usecase "Generate Bill"
  usecase "Register Member"
  usecase "Update Member"
  usecase "Search Member"
}

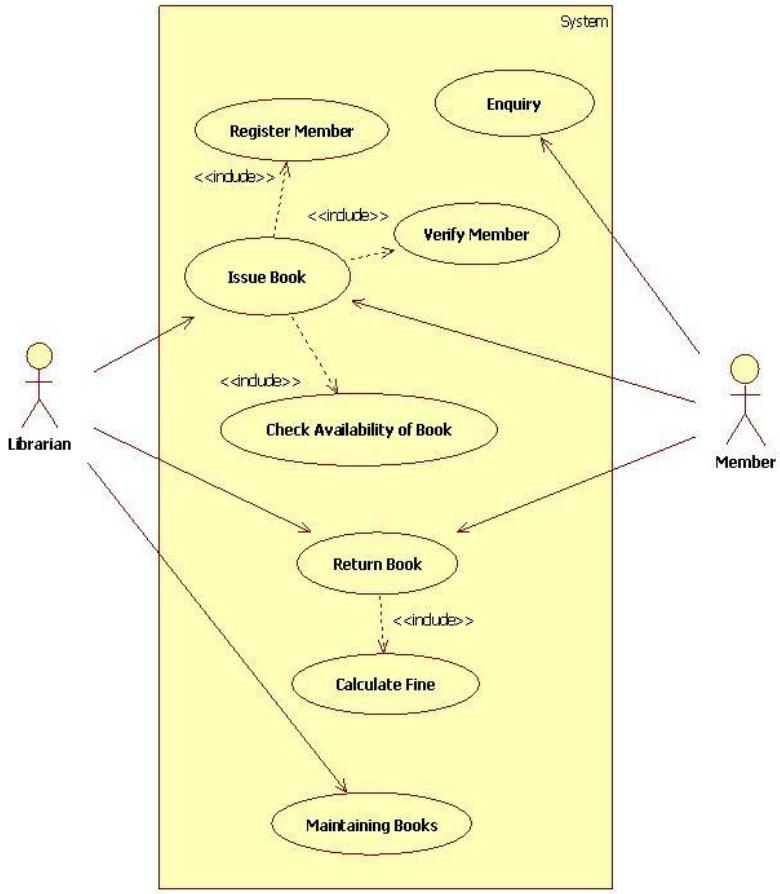
L -- (Add Book)
L -- (Update Book)
L -- (Remove Book)

L -- (Search Book)
L -- (Issue Book)
L -- (Return Book)
L -- (Generate Bill)
L -- (Register Member)
L -- (Update Member)
L -- (Search Member)

M -- (Search Book)
M -- (Issue Book)
M -- (Return Book)

Librarian --> LibraryManagementSystem
Member --> LibraryManagementSystem
@enduml

Diagram:



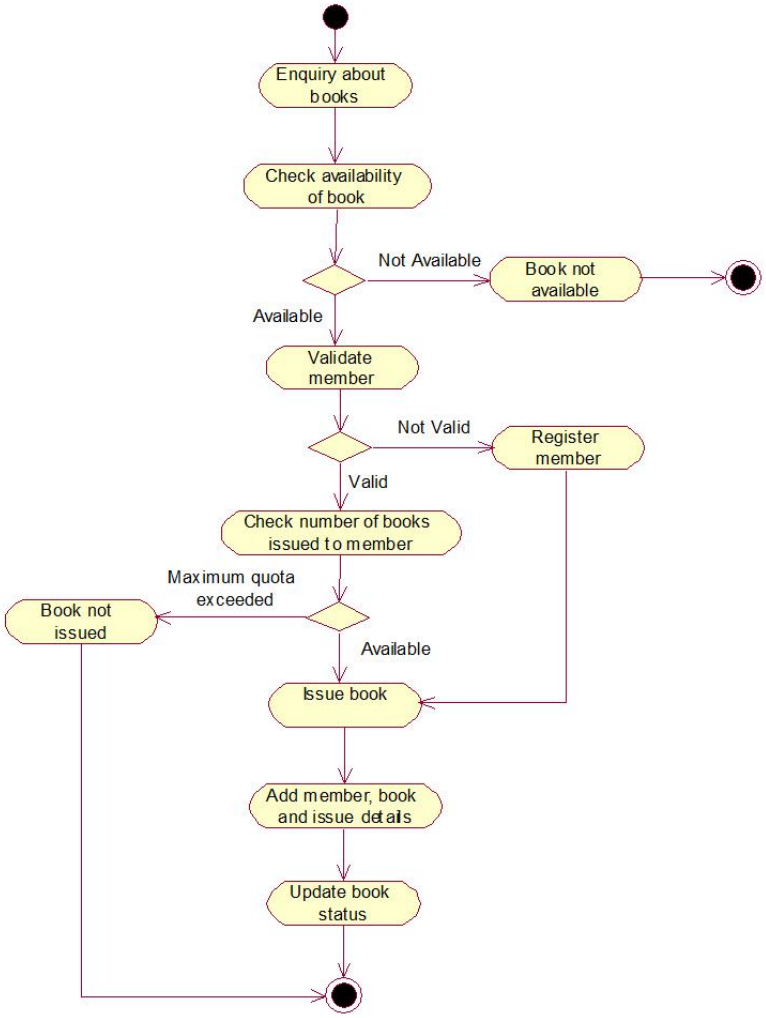
4) Activity Diagram

PlantUML code:

@startuml

```
start

:Search Book;
if (Book Found?) then (Yes)
  :Issue Book;
  :Generate Transaction Record;
else (No)
  :Display Alternatives;
endif

:Return Book;
if (Returned Late?) then (Yes)
  :Generate Bill;
else (No)
  :Update Book Status;
endif

stop
@enduml
```

Diagram:

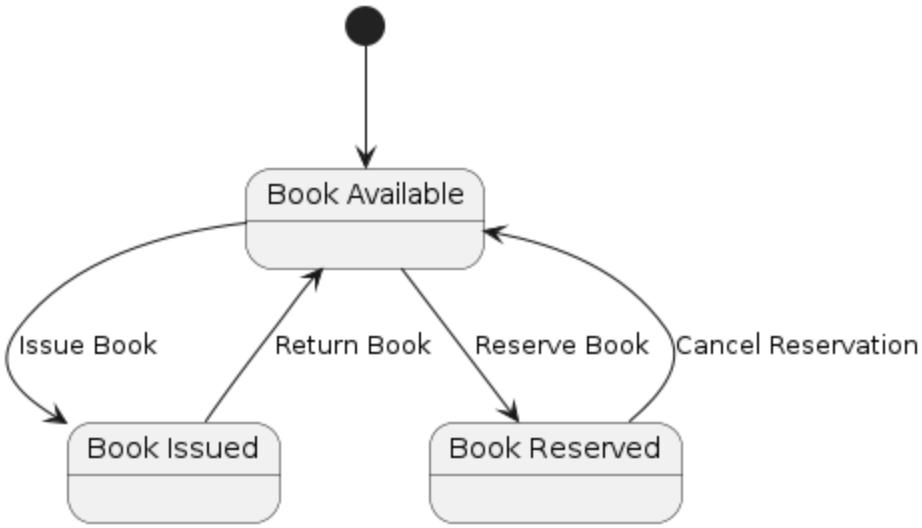


5) State Diagram

PlantUML code:

```
@startuml

[*] --> Available

state Available {
  [*] --> InStock
  InStock --> Issued: Issue
}

state Issued {
  Issued --> Returned: Return
}

state Returned {
  Returned --> Available
}

@enduml
```

Diagram:



6) Collaboration Diagram

PlantUML code:

```
@startuml


object Librarian
object Member
object Book
```

Librarian -> Book: Add Book
Book -> Librarian: Book Added

Member -> Book: Issue Book
Book -> Member: Book Issued
Member -> Book: Return Book
Book -> Member: Book Returned

@enduml

Diagram: