ISMAIL YAQUB

20227028

LAB II (Due date) – 23 May, 2025

Lab Objectives:

The Software Engineering Lab has been developed by keeping in mind the following objectives:

- To have hands on experience in developing a software project by using various software Engineering principles and methods in each of the phases of software development.
- To gain knowledge about open source tools used for implementing software engineering methods.
- To exercise developing product-startups implementing software engineering methods

Precautions:

- Use knowledge from previous courses and reliable sources, but avoid AI chatbots.
- Do not copy work from classmates; ensure originality.
- Communicate ideas clearly and concisely to avoid misunderstandings.
- Answer each question on a fresh page.
- Use all notations where applicable.
- Use this exact file.
- Submit the final report in pdf format

Scenario: You are to design a library management system

A. Draw a use case diagram for the library management system. The use case diagram should depict the relationships, if any, that exist among use cases.

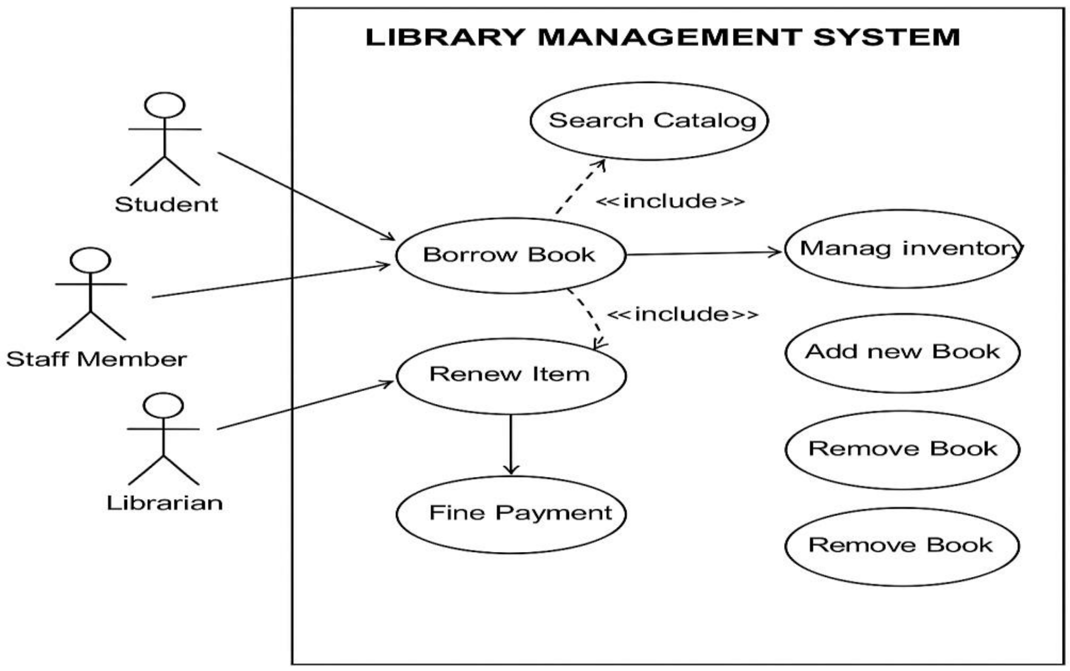
• The library has around 20,000 books and 5000 journals.

• There are three types of members in the library, students, staff members others.

• Each student can borrow up to six books and three journals for three months.

• Each staff member can borrow up to 10 books and 6 journals for 3 months.

• The other members can borrow up to two books for one month.



B. From A above:

i. Describe any two use-cases, with an appropriate template

Use Case Description

Use Case 1: Borrow Book

Use Case name: Borrow Book

Description:

A patron checks out a book from the library. The system looks into whether the member has permission and r updates the records.

Main course:

Demand for members wanting to borrow a book.

System asks for member ID.

System checks borrow limit.

Member selects a book.

System checks availability.

Book is issued.

Due date is set.

Preconditions:

Member is registered.

Book is available.

Borrow limit not reached.

Postconditions:

Book is assigned to member.

Due date recorded.

Alternate courses:

3a. Limit reached → demand for borrowing refused.

5a. Book unavailable → display return date.

Generic test scenarios:

Student borrows a book.

Member at limit → unable to borrow.

Book is not accessible → error will be displayed.

Use Case 2: Return Item

Use Case name: Return Item

Description:

Somebody brings a book or a journal back to the owner. The system flags any delays and refreshes records.

Main course:

Member returns item.

System scans item.

System checks due date.

If return on time → return the counterfoil.

Record updated.

Preconditions:

Item was borrowed.

Postconditions:

Item is returned.

Member record updated.

Alternate courses:

3a. Item late → penalty computed.

4a. Fine shown to member.

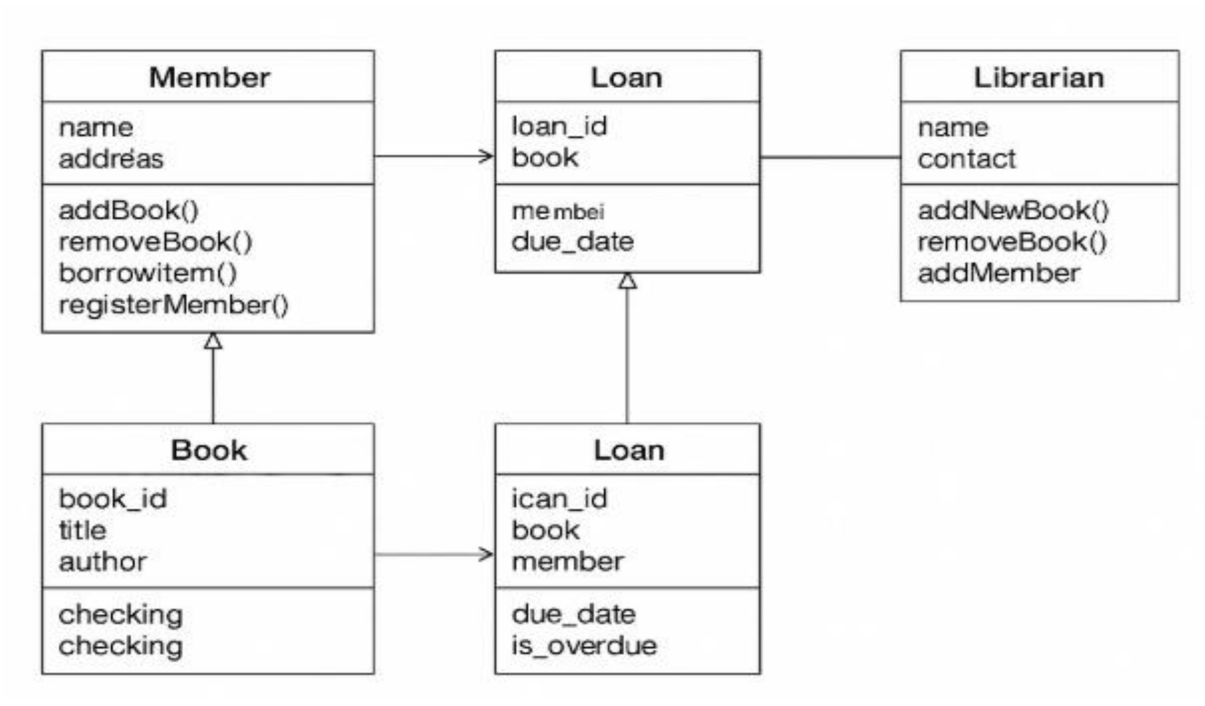
Generic test scenarios:

Member returns on time.

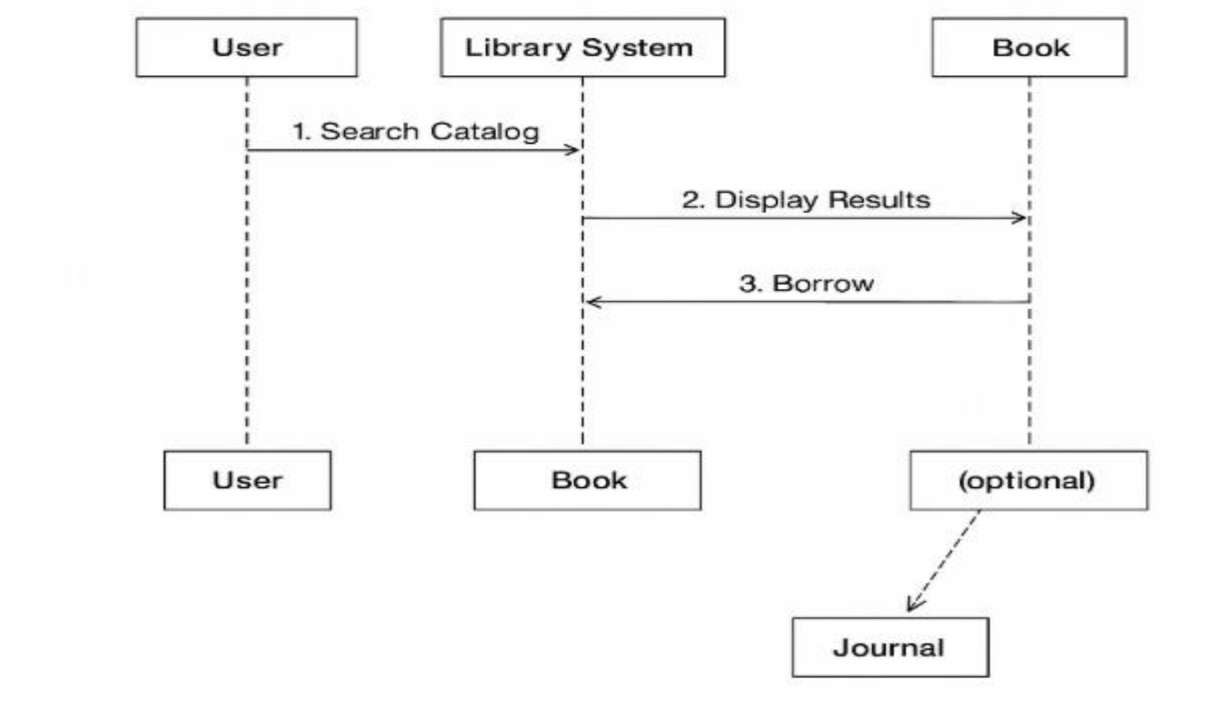
Member is late → Fine included.

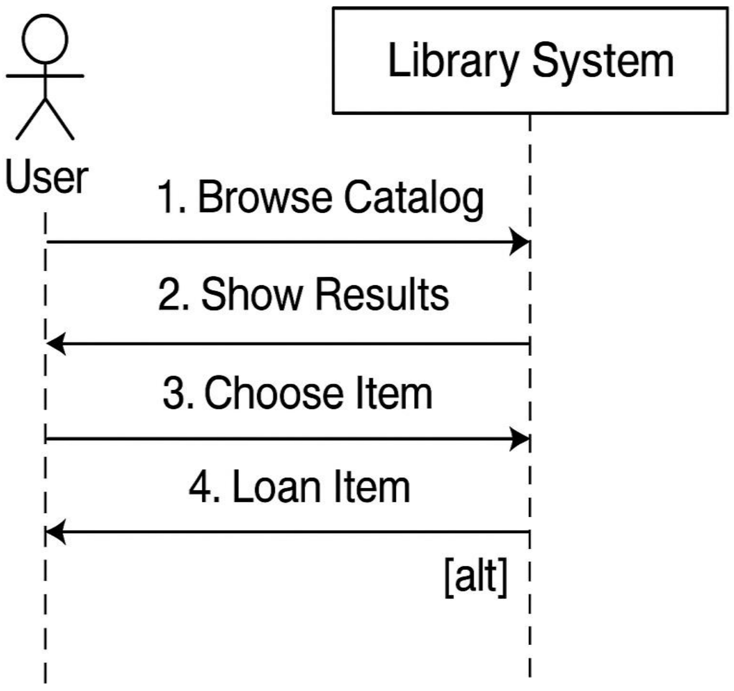
Member returns item not in system > error.

ii. Identify the classes and draw the class diagram



iii. Apply the GRASP principles





iv. Apply the GRASP principles

1.Information Expert

Responsibilities were allocated to classes that had the data required to perform them. For example, the classes such as Member (and its children: Student, Staff, OtherMember) are there to manage the borrowing limits and loan durations because those are the objects that naturally have those data. Likewise, the Book and Journal classes maintain their availability status.

2. Creator

The Library class was left to instantiate Book, Journal, and Member because it is logically the keeper of these collections, and the manager of them. This makes for clear and centralised point of object creation.

3. Controller

A controller class for the system interface (e.g., LibrarySystem) was defined to manage user requests, including borrow, return, and search operations. It serves as the bridge between the UI and the doingness, the user land where the system should behave uniform and predictable.

4. Low Coupling

Responsibilities The responsibilities were assigned in such a way that they needed as few class dependencies as possible. For instance book or journal records themselves are not added to or manipulated in the process of borrowing. Instead, it passes to the controller, which passes off the requisite actions to the right classes. This methodology will support ease of maintenance and flexibility.

5. High Cohesion

Each class in the system has a well-defined and focused purpose. The Transaction class, for example, is solely responsible for tracking issue and return dates, ensuring that it does not take on unrelated responsibilities, which makes the system easier to understand and modify.

6. Polymorphism

To handle the different borrowing rules for various member types (students, staff, others), polymorphism was applied. Each subclass of Member overrides or extends behavior related to borrowing limits and duration, enabling the system to dynamically respond to different user roles without complex conditional logic.

7. Pure Fabrication

The Transaction class is an example of pure fabrication. It doesn't directly represent a real-world object but was introduced to better support the system's functionality by handling borrowing and returning operations cleanly.

8. Indirection

By using a controller (e.g., LibrarySystem) between the UI and the core logic, indirection was introduced to decouple the user interface from the application logic. This improves the modularity and testability of the system