BUSINESS REQUIREMENTS

INTRODUCTION :

- Designing a Library Management System is a multi-step process that includes gathering requirements, identifying entities and relationships, visualizing system architecture, breaking down the system into smaller modules, creating class diagrams, identifying functionalities and algorithms, and selecting technologies, frameworks, and libraries.
- By following this step-by-step approach, a functional and efficient Library Management System can be created that meets the library's and its patrons' needs.

GATHERING REQUIREMENT:

- Gathering the functional and non-functional requirements of a Library Management System is the starting point in designing the system.

Functional Requirements

- Functional requirements are like a set of instructions for a system.
- They describe what the system should do and how it should do it.
- They cover what the system should accept as input, what it should produce as output, and how it should interact with users and other methods.
- These requirements are essential because they help ensure that the result is what everyone wants and needs.

Functional Requirements for a Library Management System

The following are the functional requirements for a Library Management System:

- Ability to add and remove books from the library Ability to search for books in the library.
- Ability to check out and return books.

- Ability to display a list of all books in the library.Ability to store and retrieve information about library patrons, including their name and ID number.
- Ability to track which books are currently checked out and when they are due to be returned
- Ability to generate reports on library usage and checkoutsNon-Functional Requirements.

NON FUNCTIONAL REQUIREMENTS :

- Non-functional requirements describe how a system should work, not just what it should do. They include how fast it should run, how secure it should be, and how easy it should be to use.
- They help make sure the system works well and meets people's expectations.

Non-Functional Requirements for a Library Management System

The following are the non-functional requirements for a Library Management System:

- User-friendly interface for easy navigation and use.
- High performance and scalability to handle large amounts of data.
- Data security and protection to ensure the privacy and confidentiality of library patrons and their information.
- Compatibility with various operating systems and devices.
- Ability to handle multiple users and concurrent access to the system.
- Compliance with relevant laws and regulations regarding library management and data privacy.Regular updates and maintenance to ensure the system remains functional and secure over time.

Entities and their Relationships

Entities

- In system design, entities are objects or concepts with distinct characteristics.

- They represent real-world things and are used to organize data within the system.
- Entities provide a structure for the data and relationships and define the data fields and connections within the system.

Entities for a Library Management System

- The following are the entities for a Library Management System:

Book: Each book in the library is represented by a Book entity.

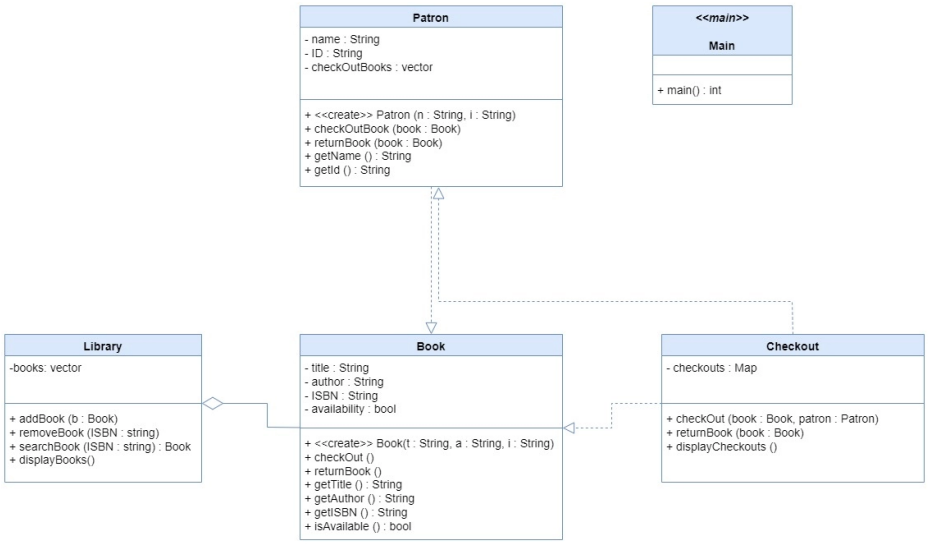
Library: The collection of books in the library is represented by the Library entity.

Patron: Each library patron is represented by a Patron entity.

Checkout: Each checkout transaction is represented by a Checkout entity

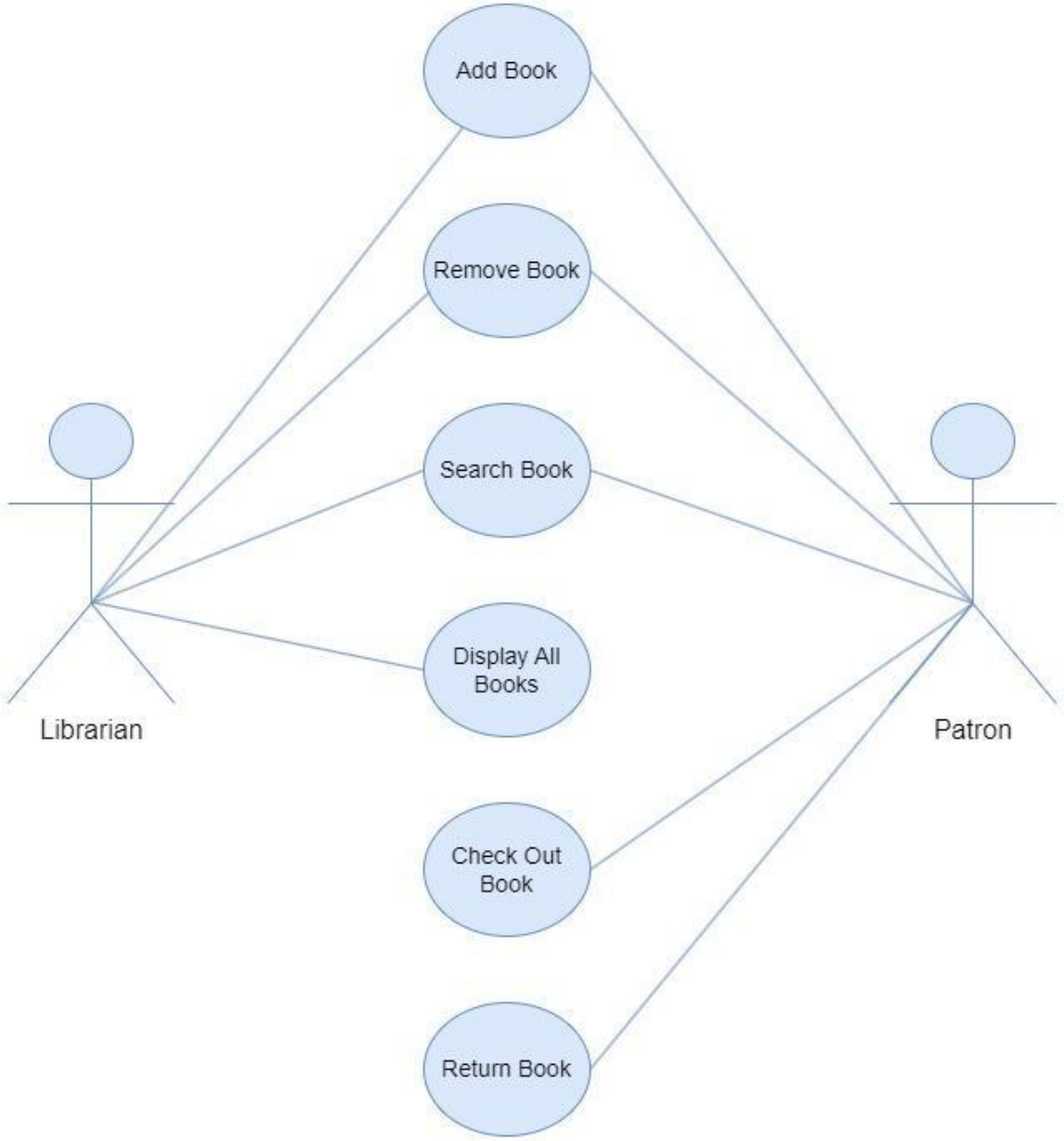
Class Diagram

- A class diagram is like a map that shows the different parts and relationships of a system.
- It helps developers plan and design the structure of a system.



Use Case Diagram

- A use case diagram is like a flowchart that shows how different people (called actors) can use a system to do different things.
- It helps developers understand what the system needs to do and how it will be used.



Low-Level Design of Library Management System Classes and Methods Involved

- Low-level design for a library management system in C++ would involve creating classes and functions to handle the various operations and data required for the system.

Book Class: This class would store information about a single book, such as its title, author, ISBN number, and availability status. It would also include functions to check out and return a book.

Library Class: This class would store a collection of books and handle operations such as adding and removing books, searching for books, and displaying the list of books.

Patron Class: This class would store information about library patrons, such as their names and ID numbers. It would also include functions to check out and return books.

Checkout Class: This class would handle the checking out and returning of books. It would store information about which patron has checked out which book and when it is due to be returned.

main() function: This function would be responsible for creating objects of the Library, Book, Patron, and Checkout classes and calling the appropriate functions to carry out the library management tasks.

Here is an example of how the Book class could be implemented in C++:

```
class Book {
private:
    string title;
    string author;
    string ISBN;
    bool availability; // true if available, false if checked out

public:
    // constructor to initialize book's information
    Book(string t, string a, string i) {
        title = t;
```

```
    author = a;
    ISBN = i;
    availability = true;
}

// function to check out a book
void checkOut() {
    if (availability) {
        availability = false;
        cout << "Book has been checked out." << endl;
    } else {
        cout << "Book is not available." << endl;
    }
}

// function to return a book
void returnBook() {
    if (!availability) {
        availability = true;
        cout << "Book has been returned." << endl;
    } else {
        cout << "Book has not been checked out." << endl;
    }
}

// function to get book's title
```

```
    string getTitle() {
        return title;
    }

    // function to get book's author
    string getAuthor() {
        return author;
    }

    // function to get book's ISBN
    string getISBN() {
        return ISBN;
    }

    // function to check if book is available
    bool isAvailable() {
        return availability;
    }
};
```