**Library Management System :**

**Description:**

This C++ program is a simple implementation of a library management system (LMS) that allows two types of users: library administrators and students. This program is used to manage library collections and allows various operations such as adding, removing, updating, deleting, reserving, purchasing, and browsing books.

Below is a brief description of the main components and features of the program.

**1. Classes:** The program uses her three main classes: "LMS", "Books", "LibraryManager", and "Student".

- "LMS" is an abstract class that defines a common interface for library administrators and students. - The "Books" class handles book-related operations. B. Save book information, view available books, and edit book reservations. – The “LibraryManager” class represents library management functions such as adding, deleting, updating, deleting, reserving, and displaying books. It also handles authentication for library administrators. - The Student class provides student-specific functionality such as purchasing books, viewing reserved books, and viewing available books. There is also an easy registration system for students.

**2. Main features:**

- The program starts with a main menu where the user can select the roles of "Library Administrator" and "Student". - Library administrators can log in, perform various book-related actions, and return to the main menu. - Students can log in to purchase books, view reserved books, and see a list of available books. You can also return to the main menu. - The program ensures basic data validation and error handling.

**3. Login System:**

- Both library administrators and students have a simple login system that requires a username and password. - Library administrators can choose their favorite questions in case they forget their login information.

**4. Book management:**

- The program manages the list of available books, and library administrators can add, update, remove, reserve, and view these books. - Students can purchase books and view reserved books.

**5. Menu navigation:**

- The program provides a menu that guides the user through the available operations and allows the user to return to the main menu at any time.

**6. User authentication:**

– The program includes basic user authentication and error handling for incorrect credentials.

1. **Loop and output:**

- The main program loop allows the user to interact with the system until they decide to exit.

In summary, this C++ program is a basic library management system that provides a platform for library managers to manage their collections and for students to purchase books and check the reserved book list. This program is designed for simplicity and provides a foundation for further development and expansion of functionality.

**Main Goals:**

1. **To analyse the requirements for a particular software system:**

**Here is a brief overview of the library management system requirements analysis:**

1. **Identify Stakeholders:**

- Find out who the stakeholders are. This may include librarians, students, administrators and possibly external users. Each stakeholder may have different needs and expectations.

**2. Collect Requirements:**

- Communicate with stakeholders to gather their requirements. This can be done through interviews, surveys or workshops. Some key questions are:

- What are the main goals and objectives of the library management system? - What special tasks do users have to perform in the system? - What are the roles and responsibilities of different user groups (eg librarians, students, administrators)? - Does the system meet legal or compliance requirements (eg data protection rules)? - What is the expected amount of data and transactions that the system should process? - Are there performance and scalability requirements? - What are the security and access control requirements? - What are the reporting and analysis needs?

**3. Functional Requirements:**

- Document functional requirements that describe what the system must do. These may include:

- User registration and authentication

- Management of book lists (add, update, delete)

- Borrowing and returning books

- Reserve books

- Good calculation

- reporting (e.g. overdue books, event history)

- Search and filter functions

**4. Non-functional requirements:**

- Identify the non-functional requirements that describe the operation of the system. These may include:

- Performance: response times for various functions. - Scalability: Ability to handle increasing numbers of users and books. - Security: access control, data encryption and vulnerability protection. - Reliability: minimize system downtime. - Ease of use: user interface design and ease of use. - Data integrity: data integrity and integrity. - Backup and recovery: Mechanisms for data backup and recovery.

**5. Use cases and user stories:**

- Create use cases or user stories to describe how users interact with the system. They should describe specific scenarios and user journeys within the system.

1. **Data Model:**

- Develop a data model that represents the structure of the database. Define entities (eg books, users) and their relationships (eg a user can borrow multiple books).

**7. UML diagrams:**

- Create UML diagrams such as use case diagrams, class diagrams and sequence diagrams to visualize the system and its design and interaction.

**8. Preference and confirmation:**

- Prioritize requirements based on importance and feasibility. Validate requirements with stakeholders to ensure they accurately meet their needs.

**9. Requirements for documents:**

- Create comprehensive requirements documents that include detailed specifications, acceptance criteria and potential limitations.

**10. Review and Feedback:**

- Share requirements documentation with stakeholders for review and feedback. Make corrections if necessary.

**11. Maintain traceability:**

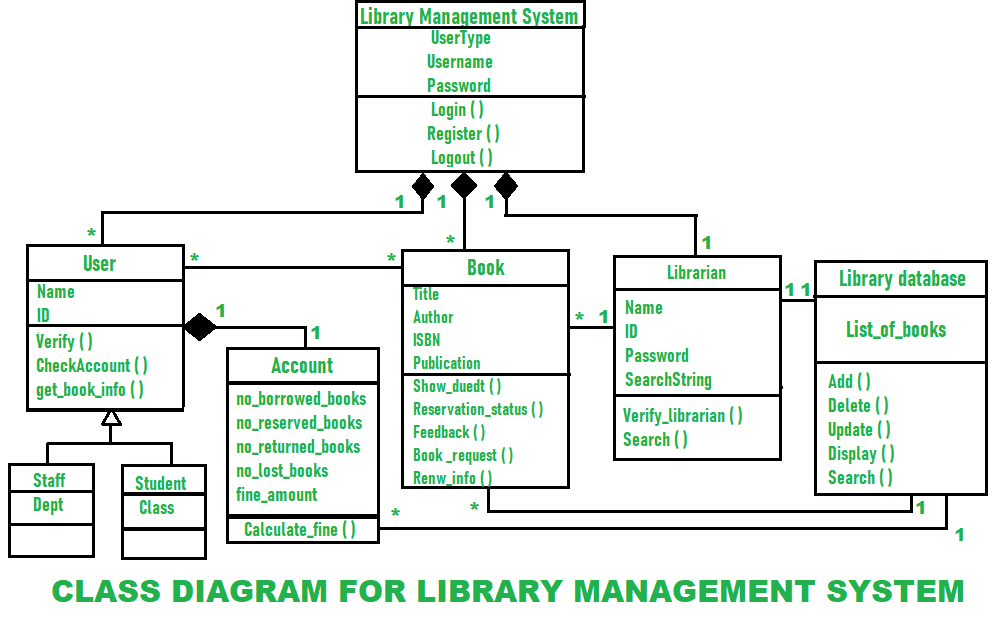
- Ensure traceability between requirements and final implementation to ensure all requirements are met.

**12. Repeat:**

- Requirements analysis may require several iterations to refine and clarify requirements as the project progresses.

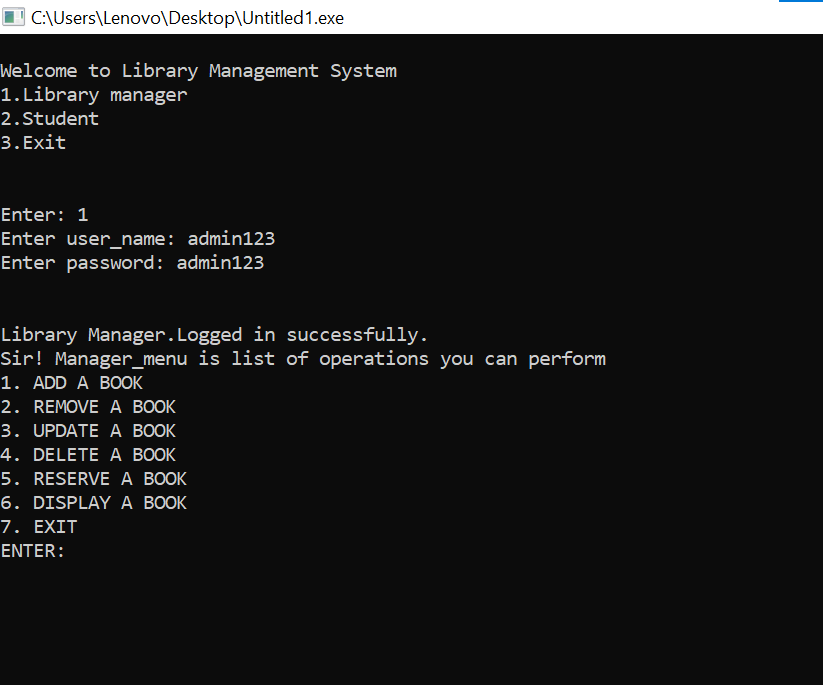
Once you have clear requirements, you can use them as a basis for system design, development, and testing. Regular communication and collaboration with stakeholders throughout the software development process is key to creating a system that meets their needs.

1. **To design and model a software system using UML:**

****

**Details :**

**Functional and Non Functional Requirements:**



**Functional requirements:**

**User registration and authentication:**

Users must be able to register an account with a unique username and password. Users must log in to access the system. Librarians and administrators should have special privileges, including the ability to manage user accounts. Booklist Management:

**Librarians must be able to:**

Add new books to the list with information (name, author, publication date, price). Update book information. Remove books from the list. Borrowing and returning books:

**Users should be able to:**

Borrow books, which should reduce the amount available. Returning books that should increase the amount available. You will receive reminders about the anniversary. Librarians can handle book loan and return requests.

**Reserve books:**

Users must be able to reserve books. If the ordered book becomes available, the system should notify the user. Users can reserve only a limited number of books at a time.

**Good calculation:**

The system should calculate fines for overdue books based on predefined rules. Fines must be collected and displayed to users.

**Reporting and Analysis:**

The system must generate reports on:

Late books. Event history. Most Cited Books User activity Users and librarians must access these reports.

**Search and filter functions:**

Users should be able to search for books by name, author or other criteria. The system should provide filter options to refine the search results. Non-functional requirements:

**Non Functional :**

The system must respond to the user's actions within a reasonable time. Event processing must be efficient. The system must handle simultaneous user interactions.

**Scalability:**

The system should accommodate an increasing number of books and users. Scalability should be achieved through efficient database design.

**Security:**

User data must be secure. Access control should limit user actions based on roles (eg, librarian, student). Data encryption should be used to protect sensitive data.

**Reliability:**

The availability of the system must be high. This should minimize downtime for maintenance and upgrades. Availability:

The user interface must be intuitive and user-friendly. Users must be able to perform tasks without extensive training.

**Data integrity:**

Data consistency must be maintained throughout the system. The information must be accurate and reliable.

**Backup and Restore:**

Data should be backed up regularly. Disaster recovery should be in place to restore the system in the event of data loss or system failure.

1. **Evidence of research and analysis to adopt an appropriate life-cycle model in the current situation:**

**Analysis of project requirements:**

Gather detailed requirements from stakeholders such as librarians, students, and administrators. Identify the need for flexibility to add, update, or remove functionality as your library's needs change. Limitations and context analysis:

Review budget, time, and resource constraints. Understand the regulatory environment regarding privacy and security. risk assessment:

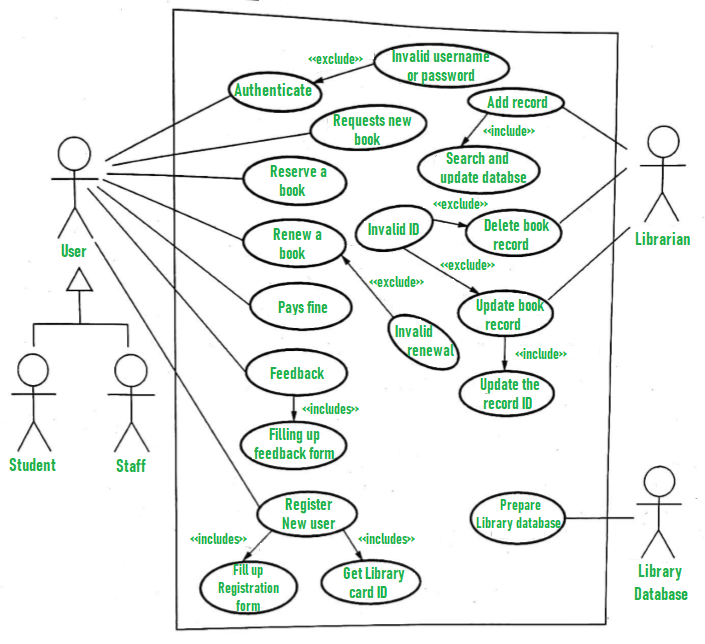
Identify potential risks associated with your project, such as: B. Technical challenges, budget constraints, or changes in requirements. Consider the risks associated with data security and integrity, especially when sensitive user information is involved. Communication with stakeholders:

Participate in regular communication with stakeholders to understand their expectations, priorities, and preferences regarding the software development process. Life cycle model research:

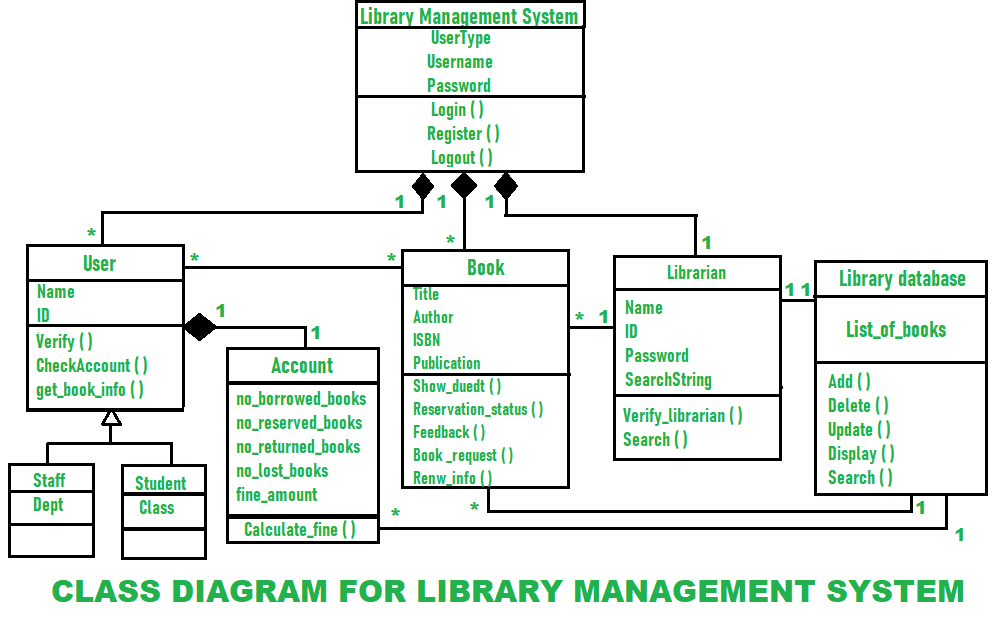
Consider different software development lifecycle models, including: B. Waterfall, Agile, Iterative, Spiral. Perform a comparative analysis to understand the strengths and weaknesses of each model from a library management system perspective.

1. **Initial System Architecture/Design Specification. The design/architecture specification should be enhanced using 4 UML diagrams to model several phases of the system:**

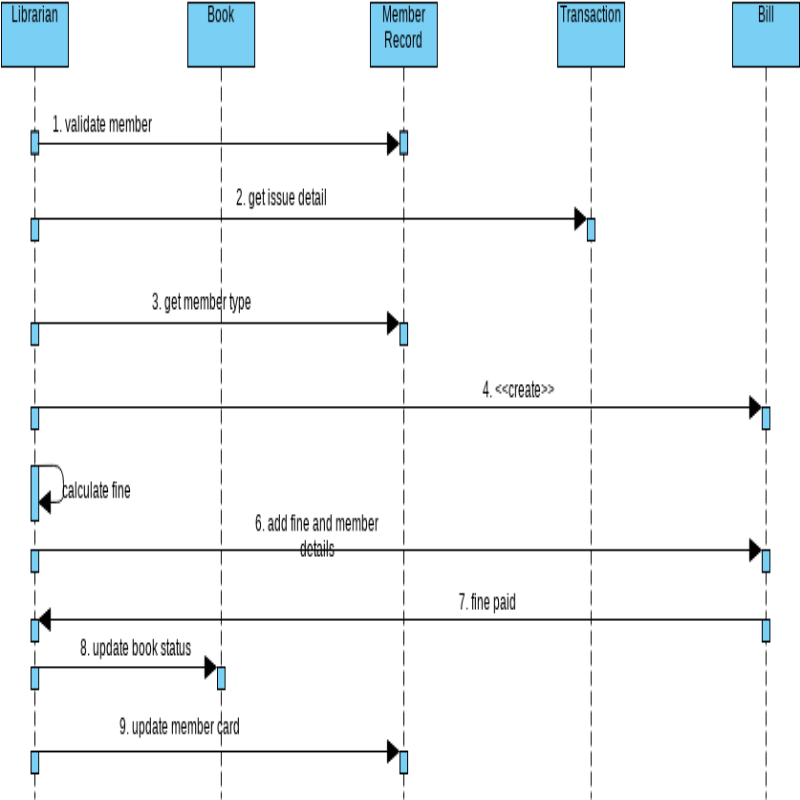
**Use case :**

****

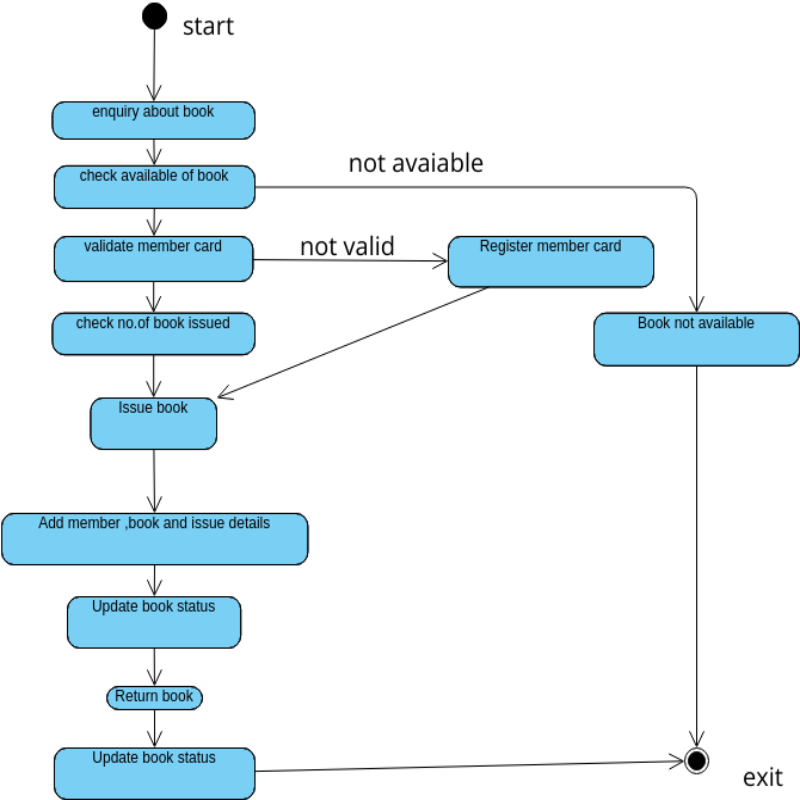
**Class Diagram:**

****

**Use case Diagram :**

****

**Machine Diagram :**

****