# Getting Ready: Library Management System

Understand the library management system problem and learn the questions to simplify this problem.

**We'll cover the following**

* [Problem definition](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Problem-definition)
* [Expectations from the interviewee](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Expectations-from-the-interviewee)
  + [Efficient searching](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Efficient-searching)
  + [Versatility](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Versatility)
  + [Book reservation](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Book-reservation)
  + [Book renewal](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Book-renewal)
  + [Fine management](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Fine-management)
* [Design approach](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Design-approach)
* [Design pattern](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Design-pattern)

## Problem definition

A **library management system** **(LMS)** aims to automate all library activities. It is a software that helps manage all the primary functions of library management. With the help of a library management system, we can organize, handle, and maintain the record of numerous books and the members in a comprehensive and systematic way.

A librarian can use this software to track the number of books in the library. They can also use it to retain several records including, the new books, borrowed books with due dates, the member who borrowed books, returned books, fine on the late returned books, etc. In short, the library management system stores and updates the complete library database.

LMS also supports maintaining the physical library. The user can keep track of the position of the book in the library and can search for whether or not the specific book is currently available in the library. Therefore, LMS helps organize and retrieve library data in an efficient manner. .

An example of the library management system

## Expectations from the interviewee

There are multiple components of the LMS, each with its own specific requirements and constraints. Let’s look at some of the main expectations that the interviewer will want to hear you discuss in more detail during the interview.

### Efficient searching

Searching for books is one of the most crucial functions of LMS. The user must be able to search for any book. Different users may want to search for a book through different methods. Therefore, the interviewer can ask questions like these:

* Would the user be able to search for a book using attributes other than the book name?
* How will the user be able to search for a book by its author name, publication date, etc.?
* How will the user search a specific category of books like magazines, journals, newspapers, etc.?

### Versatility

Before designing the system, it is mandatory to specify the actors of the system. Hence, the interviewer can ask about the actors of the system as follows:

* Can the software only be used by a librarian or by all library members?

### Book reservation

Another significant feature of LMS is the reservation of the book.

* What is the mechanism of book reservation?
* Can a member reserve a book again if it is already reserved?
* How does the status of the book change when a member returns a book?

### Book renewal

Similar to the book reservation, the interviewer can ask about the book renewal functionality with a question like this:

* What is the mechanism of book renewal if a member wants to hold a book for a longer period of time?

### Fine management

There is another question that the interviewer may be interested to ask:

* How is the calculation and deduction of fines handled if the book is returned late?

## Design approach

We are going to design this library management system using the bottom-up design approach. For this purpose, we will follow the steps below:

* Identify and design the smallest components first.
* Use these small components to design bigger components.
* Repeat the steps above until we design the whole system.

## Design pattern

It is always a good practice to discuss the design patterns that an LMS falls under, during the interview. Stating the design patterns will give the interviewer a positive impression and shows that the interviewee is well-versed in the advanced concepts of object-oriented design.

The following design patterns can be used to design the library management system:

* Factory design pattern
* Delegation design pattern
* Observer design pattern

Let’s explore the requirements of the library management system in the next lesson.

Back

Code of Elevator System

**Requirements for the Library Management System**

Learn about all requirements of the library management system.

**We'll cover the following**

* [Requirement collection](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Requirement-collection)

In this lesson, we will list the requirements of our library management system. This is a very crucial step as requirements define the scope of a problem, so getting them right from the interviewer and understanding them well will make the system designing process smooth and easy.

We’ll use the notational convention to identify each requirement with a unique label “Rn”, where “R” is short for Requirement and “n” is a natural number.

**Requirement collection**

For LMS (Library Management System), the requirements have been defined below:

**R1:** The system should be able to store the information about books and members of the library. Moreover, the complete log of the book borrowing process should also be stored.

**R2:** Every book is supposed to have a unique identification number and other details including a rack number to help locate the book physically.

**R3:** Every book should have an associated ISBN, title, author name, subject, and publication date.

**R4:** There can be multiple copies of the book. Each copy will be recognized as a book item.

**R5:** There can be two types of users, i.e., the librarian and the members.

**R6:** Every user must have a library card with a unique card number.

**R7:** One member can issue a maximum of 10 books at a time.

**R8:** The member can issue a book for a maximum of 15 days.

**R9:** Each book item can only be reserved by a single member.

**R10:** The system should be able to keep a record of who issued or reserved a particular book and on which date.

**R11:** The system should allow the user to renew the reserved book.

**R12:** The system should send a notification if the book is not returned within the due date.

**R13:** If the book is currently not available, then the member should be able to reserve it for whenever it is available.

**R14:** The system should allow the user to search a book by its title, author name, subject, or publication date.

We’ve identified our requirements for the problem, and in the next lesson, we will define different use cases of our library management system.

# Use Case Diagram for the Library Management System

Learn how to define use cases and create the corresponding use case diagram for the library management system.

**We'll cover the following**

* [System](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#System)
* [Actors](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Actors)
  + [Primary actors](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Primary-actors)
  + [Secondary actors](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Secondary-actors)
* [Use cases](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Use-cases)
  + [Librarian](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Librarian)
  + [Member](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Member)
  + [System](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#System)
* [Relationships](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Relationships)
  + [Generalization](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Generalization)
  + [Associations](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Associations)
  + [Include](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Include)
  + [Extend](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Extend)
* [Use case diagram](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Use-case-diagram)

Let’s build the use case diagram for the library management system and understand the relationship between its different components.

First, we will define the different elements of our library, followed by the complete use case diagram of the system.

## System

Our system is a "library."

## Actors

Next, we will define the main actors of our library management system.

### Primary actors

* **Member:**This actor is the client of the library. It can search, reserve, renew, or return a book and modify its library membership.
* **Librarian:**It acts as an admin in a library management system. It can add or remove a book as well as modify the status of members or books. It can also issue and return books.

### Secondary actors

* **System:** It can send alerts related to reservations and late returns of books.

## Use cases

In this section, we will define the use cases for the library. We have listed the use cases according to their respective interactions with a particular actor.

**Note:**You will see some use cases occurring multiple times because they are shared among different actors in the system.

### Librarian

* **Add book:**To add a new book to the library
* **Remove book:**To remove an existing book from the library
* **Edit book:**To modify a book
* **Register new account:**To register a new library member
* **Cancel membership:**To cancel the library membership of a member
* **Register/Update account:**To create or update an account
* **Login/Logout:**To log in or log out account
* **Issue book:** To issue a book to a member
* **Remove reservation:** To remove reservation of books
* **Renew book:** To renew the issuance of the book
* **Reserve book:**To reserve a book that is currently not available
* **View account:**To view the account and access all account details

### Member

* **Search catalog:**To search for a book in the library
* **Cancel membership:**To cancel the library membership of a member
* **Register/Update account:**To create or update an account
* **Login/Logout:**To log in or log out of the account
* **Checkout book:**To complete the issue book process
* **Remove reservation:** To remove the reservation of books
* **Renew book:** To renew the issuance of the book
* **Reserve book:**To reserve a book that is currently not available
* **View account:**To view the account and access all account details
* **Return book:**To return a book to the library

### System

* **Overdue notification:** To send an alert if the book is not returned on time
* **Reservation available notification:** To send an alert when the book is available for reservation
* **Reservation canceled notification:**To send an alert when a book reservation is canceled

There are some use cases that are not directly related to any actor. They are elaborated below.

* **Add book item:**To add an item of a book in the catalog
* **Edit book item:**To edit the details of a book item in a catalog
* **Remove book item:**To remove a book item from the catalog
* **Update catalog:**To update (add, edit, or remove) a book item or book from the catalog
* **Issue library card:**To issue a library card to new members that will be for identification.
* **By subject name:** To search for a book in the catalog by its subject name
* **By book title:**To search for a book in the catalog by its title
* **By author name:**To search for a book in the catalog by its author name
* **By publication date:**To search for a book in the catalog by its publication date
* **Pay fine:**To pay a fine if the book is returned after the due date

## Relationships

This section describes the relationships between and among actors and their use cases.

### Generalization

We can search for a book with the title, subject name, author name, or publication date. This shows that the “Search catalog” use case has a generalization relationship with “By subject name,” “By book title,” “By author name,” and “By publication date” use cases.

### Associations

The table below shows the association relationship between actors and their use cases.

|  |
| --- |
|  |

A screenshot of a computer

Description automatically generated

### Include

* To add a new book, we add its copies (book items), so the "Add Book" use case has an include relationship with the "Add bookitem"use case.
* To edit a book, we need to edit its items, so the "Edit Book" use case has an include relationship with the "Edit book item"use case.
* To remove a book from the library, we need to remove its items, so the "Remove Book" use case has an include relationship with the "Remove book item"use case.
* To update a catalog, we need to update all the book items. This will include adding, editing, or removing a book item since “Edit book item,” “Add book item,” and “Remove book item” have an include relationship with the “Update catalog” use case.
* To issue a book, we need to go through a checkout process, so the “issue book” use case has an include relationship with the “Checkout book” use case.
* Whenever we go through the checkout process, our book reservation will be removed as it had been issued. So the "Checkout book"use case has an include relationship with the "Remove reservation"use case.

### Extend

* When a new member is registered, a library card is issued. So the "Register new member" use case has an extend relationship with the "Issue library card" use case.
* Whenever a member returns a book, the librarian will check if the submission is late or not and will ask the member to pay for a fine if it exists, so the "Return book" use case has an extend relationship with "Pay fine" use case.

## Use case diagram

Here is the use case diagram of the library management system:

A diagram of a library system

Description automatically generated

In the next lesson, we will discuss the class diagram with a detailed explanation of all classes and their relationship with each other.

# Class Diagram for the Library Management System

Understand how to create a class diagram for a library management system by using the bottom-up approach.

**We'll cover the following**

* [Components of a library management system](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Components-of-a-library-management-system)
  + [Book and book item](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Book-and-book-item)
  + [Rack](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Rack)
  + [Person and author](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Person-and-author)
  + [User, librarian, and library member](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#User,-librarian,-and-library-member)
  + [Library card](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Library-card)
  + [Book reservation](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Book-reservation)
  + [Book lending](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Book-lending)
  + [Notification](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Notification)
  + [Search and catalog](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Search-and-catalog)
  + [Library](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Library)
  + [Enumerations](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Enumerations)
  + [Custom data type](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Custom-data-type)
* [Relationship between the classes](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Relationship-between-the-classes)
  + [Association](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Association)
    - [One-way association](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#One-way-association)
    - [Two-way association](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Two-way-association)
  + [Composition](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Composition)
  + [Aggregation](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Aggregation)
  + [Inheritance](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Inheritance)
* [Class diagram of the library management system](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Class-diagram-of-the-library-management-system)
* [Design pattern](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Design-pattern)
* [Additional requirements](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Additional-requirements)

Here, we’ll create the class diagram for our system on the basis of the requirements that we gathered previously. In the **class diagram**, we will first design/create the classes, abstract classes, and interfaces for the system, and then we’ll identify the relationship between classes in accordance with all requirements of the library management system.

## Components of a library management system

In this section, we will define the classes for LMS. Since we are following the bottom-up approach for designing a class diagram, we’ll first create the classes of small components. After that, we will integrate those components and create the class diagram for the whole library management system.

### Book and book item

Book is an abstract class in which the complete information of the book is stored like ISBN, title, subject, etc. Since the book can be written by many authors, there is a complex member named authors in the class representing the list of book authors. The Book class consists of the enum BookFormat for encountering the different book types. It will be helpful in displaying the catalog and assists in searching as well.

The BookItem class extends the Book class, which represents a single and unique copy of the book. Each book item has its price, rack information, etc. related to it. A particular book is placed in a particular position in the library. So, the Rack type object will take care of where the book is placed. This class is using an enum BookStatus to keep track of the current status of the book. All the book borrowing and purchasing information is associated with the BookItem class. The UML representation of Book and BookItem is shown in the class diagram below:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a library card

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

### Search and catalog

Search is one of the most important functionalities of the system. Search is the interface that allows the user to search for any book and return the list of books upon searching by any of the following methods:

* Search a book by its **title**.
* Search a book by its **author name.**
* Search a book by its**subject.**
* Search a book by its **publication date.**

Catalog is a class where the search functionality is implemented. In each catalog, the books are sorted according to one of the given search techniques, i.e., on the basis of the book’s title, author, subject, or publication date.

The following UML diagram shows this relationship:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a bookkeeping system

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A diagram of a library management system

Description automatically generated

## Design pattern

We can apply the Factory design pattern to create objects and mandate that they go via a single factory. For example, we can create a BookFactory class to create a book object in an arranged manner.

Similarly, we can use the Delegation design pattern to delegate a task from one class to another class. For example, librarian functionalities like adding book items, deleting book items, or modifying book items are actually implemented in the BookItem class. The Librarian class uses the BookItem class and has access to its data and methods.

Moreover, we can use the Observer design pattern to notify library members. For example, if a member searches for a book that is unavailable at that time, then the observer interface system will notify the member when that book is available for reservation.

## Additional requirements

The interviewer can introduce some additional requirements in LMS, or they can ask some follow-up questions. Let’s see an example of additional requirements:

**Barcode Reader:** Each member should have a unique barcode on their library card and each book should also have a distinct barcode associated with it, and the system should be able to scan the barcode of every book and member. To fulfill this requirement, we have the class diagram shown below:

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

**Sequence Diagram for the Library Management System**

Create a sequence diagram for lending a book from the library and solve a challenge.

**We'll cover the following**

* [Issue a book](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Issue-a-book)
* [Sequence challenge: Return a book](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Sequence-challenge:-Return-a-book)

Sequence diagrams are a great way to understand the interactions between different entities and objects in the system. There can be different sequence diagrams that we can create for our library management system. For the sake of this lesson, we will create sequence diagrams for the following two interactions:

* **Lend a book:** The member requests the librarian to lend them a book.
* **Sequence challenge:** The member returns a book.

**Issue a book**

The sequence diagram for issuing a book should have the following actors and objects that will interact with each other:

* **Actors:** Member and Librarian
* **Object:**Book

Here are the steps in the sequence to issue a book:

1. The member requests to issue a book.
2. The librarian verifies the lending quota of the member.
3. If the lending quota is equal to the maximum quota:
   1. The librarian informs the member that the maximum quota is reached. No more books can be issued.
4. Else if the lending quota is less than the maximum quota:
   1. The librarian gets the book status.
   2. If the book is available:
      1. The librarian issues the book to the member.
   3. If the book is reserved:
      1. The librarian cancels the member’s request to issue the book.

Based on the order above, the sequence diagram below demonstrates the issuance of a book in the library management system.

A screenshot of a computer

Description automatically generated

## Sequence challenge: Return a book

You will complete a sequence diagram for the return of a book to the library. The sequence diagram below demonstrates a skeleton for the return of a book to the library.

A diagram of a diagram

Description automatically generated

Notice that the arrows in the above diagram are numbered from 1 to 13. The message boxes shown below are the messages to be exchanged between the actor(s) and object(s). Can you rearrange the below messages in the correct sequence of order they should appear in the sequence diagram’s skeleton above?

**Note:** If you get stuck, just click the “Show Solution” button to check out the correct answer.

Alternatively, you can also click the "Show complete diagram" button below to view the complete sequence diagram of the return book sequence.

A screenshot of a computer

Description automatically generated

# Activity Diagram for the Library Management System

Create some activity diagrams for the library management system problem.

**We'll cover the following**

* [Check out a book from the library](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Check-out-a-book-from-the-library)
  + [States](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#States)
  + [Actions](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Actions)
* [Return a book to the library](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Return-a-book-to-the-library)
  + [States](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#States)
  + [Actions](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Actions)
* [Activity challenge: Renew a book from the library](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Activity-challenge:-Renew-a-book-from-the-library)

Activities diagrams are a great way to visualize the flow of messages from one activity to the other in the system. There can be different activity diagrams that we can create for our LMS. For the sake of this lesson, we will create activity diagrams for the following three activities:

* Check out a book from the library.
* Return a book to the library.
* **Activity challenge:**Renew a book from the library.

## Check out a book from the library

The following are the states and actions involved in this activity diagram.

### States

**Initial state:**The member selects a book and initiates checkout.

**Final state:**There are two final states present in this activity diagram, shown below:

* The member completes the checkout process successfully, and the book will be allocated to the member.
* An error occurred during the checkout process due to book unavailability or issue book limit exceeds.

### Actions

The member selects a book and enters the ID. The system will perform a few checks like book availability, the max limit of the member, and book reservations. If all checks are clear, then the book will be issued. Otherwise, the system will show an error message.

Based on the order shown above, the activity diagram of a checkout book from the library is given below.

A diagram of a book

Description automatically generated

### States

**Initial state:**The member returns a book back to the library.

**Final state:**There are two final states present in this activity diagram, shown below:

* The member completes the return process and pays a fine, if any.
* The system allocates a book to someone who reserved that book.

### Actions

The member enters the book ID. The system will check if the book is returned within the due date, and the member will pay a fine, if any. Then, the book will be allocated to someone who has reserved the book.

Based on the order above, the activity diagram below demonstrates returning a book to the library.

A diagram of a bookkeeping process

Description automatically generated

## Activity challenge: Renew a book from the library

You will create an activity diagram of a member renewing a book from the library.

The skeleton of the activity diagram given below demonstrates a customer looking for book renewal from the library.

A diagram with arrows and points

Description automatically generated

A screenshot of a diagram

Description automatically generated

# Code of Library Management System

Let's write the code for the designed classes in different languages.

**We'll cover the following**

* [Library management](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Library-management)
  + [Enumerations](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Enumerations)
  + [Address and person](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Address-and-person)
  + [User](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#User)
  + [Book reservation, book lending and fine](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Book-reservation,-book-lending-and-fine)
  + [Book and rack](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Book-and-rack)
  + [Notification](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Notification)
  + [Search and catalog](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Search-and-catalog)
  + [Library](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Library)
* [Wrapping up](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Wrapping-up)

We’ve gone over the different aspects of the library management system and observed the attributes attached to the problem using various UML diagrams. Let us now explore the more practical side of things, where we will work on implementing the library management system using multiple languages. This is usually the last step in an object-oriented design interview process.

We have chosen the following languages to write the skeleton code of the different classes present in the library management system:

* Java
* C#
* Python
* C++
* JavaScript

## Library management

In this section, we will provide the skeleton code of the classes designed in the class diagram lesson.

**Note:** For simplicity, we are not defining getter and setter functions. The reader can assume that all class attributes are private and accessed through their respective public getter methods and modified only through their public method functions.

### Enumerations

First, we will define all the enumerations required in the library management system. According to the class diagram, there are four enumerations used in the system: BookFormat, BookStatus, ReservationStatus, and AccountStatus. The code to implement these enumerations is as follows:`

**Note:** JavaScript does not support enumerations, so we will be using the Object.freeze() method as an alternative that freezes an object and prevents further modifications.

// definition of enumerations used in library management system

enum BookFormat {

HARDCOVER,

PAPERBACK,

AUDIOBOOK,

EBOOK,

NEWSPAPER,

MAGAZINE,

JOURNAL

}

enum BookStatus {

AVAILABLE,

RESERVED,

LOANED,

LOST

}

enum ReservationStatus {

WAITING,

PENDING,

CANCELED,

NONE

}

enum AccountStatus {

ACTIVE,

CLOSED,

CANCELED,

BLACKLISTED,

NONE

}

### Address and person

This section contains the code for Address and Person classes where the Person class is composed of an Address class. The implementation of these classes can be found below:

Java

public class Address {

private String streetAddress;

private String city;

private String state;

private int zipCode;

private String country;

}

public class Person {

private String name;

private Address address;

private String email;

private String phone;

}

### User

The User is an abstract class that represents the various people or actors that can interact with the system. Since there are two types of users, the librarian and the library member, the user can either be a Librarian or a Member. The implementation of the mentioned classes is shown below:

// User is an abstract class

public abstract class User {

private String id;

private String password;

private AccountStatus status;

private Person person;

private LibraryCard card;

public abstract boolean resetPassword();

}

public class Librarian extends User {

public boolean addBookItem(BookItem bookItem);

public boolean blockMember(Member member);

public boolean unBlockMember(Member member);

public boolean resetPassword() {

// definition

}

}

public class Member extends User {

private Date dateOfMembership;

private int totalBooksCheckedOut;

public boolean reserveBookItem(BookItem bookItem);

private void incrementTotalBooksCheckedOut();

public boolean checkoutBookItem(BookItem bookItem);

private void checkForFine(String bookItemId);

public void returnBookItem(BookItem bookItem);

public boolean renewBookItem(BookItem bookItem);

public boolean resetPassword() {

// definition

}

}

### Book reservation, book lending and fine

This component shows the implementation of BookReservation, BookLending, and Fine classes. These classes will be responsible for managing reservations against books, managing reservations, and calculating fine on books. The code is shown below:

public class BookReservation {

private String itemId;

private Date creationDate;

private ReservationStatus status;

private String memberId;

public static BookReservation fetchReservationDetails(String bookItemId);

}

public class BookLending {

private String itemId;

private Date creationDate;

private Date dueDate;

private Date returnDate;

private String memberId;

public static boolean lendBook(String bookItemId, String memberId);

public static BookLending fetchLendingDetails(String bookItemId);

}

public class Fine {

private Date creationDate;

private String bookItemId;

private String memberId;

public static void collectFine(String memberId, long days);

}

### Book and rack

The Book is an abstract class and BookItem represents each copy of the book. For example, if there are two copies of the same book then there would only be one Book object and two BookItem objects. The code to implement these classes is as follows:

Java

C#

// User is an abstract class

public abstract class Book {

private String isbn;

private String title;

private String subject;

private String publisher;

private String language;

private int numberOfPages;

private BookFormat bookFormat;

private List<Author> authors;

}

public class BookItem extends Book {

private String id;

private boolean isReferenceOnly;

private Date borrowed;

private Date dueDate;

private double price;

private BookStatus status;

private Date dateOfPurchase;

private Date publicationDate;

private Rack placedAt;

public boolean checkout(String memberId);

}

public class Rack {

private int number;

private String locationIdentifier;

}

### Notification

The Notification class is another abstract class responsible for sending notifications to the users, with the PostalNotification and EmailNotification classes as its child classes. The implementation of this class can be found below:

Java

// User is an abstract class

public abstract class Notification {

Private String notificationId;

Private Date creationDate;

Private String content;

public abstract boolean sendNotification();

}

public class PostalNotification extends Notification {

private Address address;

public boolean sendNotification(){

// definition

}

}

public class EmailNotification extends Notification {

private String email;

public boolean sendNotification(){

// definition

}

}

### Search and catalog

The Search is an interface used in the efficient searching of library books by various methods, and the Catalog class is used to implement the search interface to help in book searching. The code to perform this functionality is presented below:

Java

C#

public interface Search {

// Interface method (does not have a body)

public List<Book> searchByTitle(String title);

public List<Book> searchByAuthor(String author);

public List<Book> searchBySubject(String subject);

public List<Book> searchByPublicationDate(Date publishDate);

}

public class Catalog implements Search {

private HashMap<String, List<Book>> bookTitles;

private HashMap<String, List<Book>> bookAuthors;

private HashMap<String, List<Book>> bookSubjects;

private HashMap<String, List<Book>> bookPublicationDates;

public List<Book> searchByTitle(String query) {

// definition

}

public List<Book> searchByAuthor(String query) {

// definition

}

public List<Book> searchBySubject(String query) {

// definition

}

public List<Book> searchByPublicationDate(String query) {

// definition

}

}

### Library

The final class of LMS is the Library class which will be a Singleton class, meaning the entire system will have only one instance of this class. The implementation of this class can be found below:

public class Library {

private String name;

private Address address;

public Address getAddress();

// The Library is a singleton class that ensures it will have only one active instance at a time

private static Library library = null;

// Created a static method to access the singleton instance of Library class

public static Library getInstance() {

if (library == null) {

library = new Library();

}

return library;

}

}

**Wrapping up**

We've explored the complete design of a library management system in this chapter. We've looked at how a basic library management system can be visualized using various UML diagrams and designed using object-oriented principles and design patterns.

Back