Ans1. **Main Actors:**

* User
* Librarian
* Administrator

**Main Objects:**

* Book
* User Account
* Notification
* Fine
* Library item

**Functionalities:**

* **User:**

Attributes: ID, name, email, Books: list <books>

Methods: getter & setter methods, PayFine(), Borrow\_a\_book (), Return\_a\_book(), Search-a-book(), librarian: Attributes:- (ed.

* **Librarian:**

Attributes: ID, Name, Email

Methods: getter & setter methods, cancel\_regristration(), update, add, remove\_a\_book(), Issue\_a\_book(), update\_memberDetails(), Issue\_Membership(), charge\_Fine

* **Administrator:**

Attributes: ID, Name, Email

Methods: getter & setter methods, Issue\_fine(); Send\_Notification(); update, add, remove, Issue\_a\_book()

* **Book:**

Attributes: String Name, Boolean IsBorrowed, Double Price, String Author\_Name, String ISBN, Publication\_date

Methods: getter and setter method

* **Notifications:**

Attributes: String message, Notification Id(String)

Methods: getter & setter methods, Send-Notification().

* **Fine:**

Attributes: Amount, Due Date

Methods: getter & setter methods

* **Library\_Item:**

Attributes: Item\_Type, Item\_Id, Item-Title, IsBorrowed(Boolean)

Methods: getter & setter methods

Ans3. Book is inherited from Library item. It helps in organizing & code reusability.

* We can have common attributes in library Item (base class) like: title, author, ISBN.
* Can have common methods like :- returnItem(); checkAvailability(); This organization makes code readable & maintainable.

Ans4.

* Can have abstract class for Notification, which can have abstract methods like write\_Message(), send\_Message(), can be implemented by class: Librarian & Administrator.
* Can have abstract class for Reservation Handling as well, which will contain method to update the availability of reserved books, ensuring fairness in the reservation Queue.

Ans5. **Challenges:**

* **Handling Book Reservation:** It is very difficult to maintain fairness in the Reservation Queue

Maintain a reservation queue to keep track of users who have reserved the book. It must follow first come, first serve principle. When a user requests a book reservation, add them to reservation queue & update user about their position in queue through notifications. Track availability of status of book, if reserved book is returned, notify the 1st user in the queue and mark book as reserved. Set a time limit within which a reserved book must be borrowed, otherwise it moves to next user. Set a limit on active reservations a user can have,

* **Late returns and Fine**: When user borrows book, record issue date & calculate due date.

Store the due date. Implement function that calculate fine for late returns based on difference b/w due date & actual return date. Notify the user about fines and due dates via email.

* **User Notifications:** Implement notification system that can send notifications. Design notification queue to handle notifications synchronously. Define different types of notifications. based on events such as due date approaching, reserved Hem availability etc.
* **Inventory management:** Make inventory class, that will maintain list of library Items, to add remove or search for Items. Assign ISBN (unique identifier) to each item to track & manage it easily. store detailed info about each Item. Maintain availability status for each item. (\*) Enable user to search item based on various criteria like: title, author, etc.