A. Problem Description

The **Library Management System (LMS)** is designed to simplify the daily operations of a library by automating book management, member registration and transaction tracking. In most of the libraries, managing records are still handled manually. This method often creates inefficiencies, errors, and loss of valuable time. To handle these challenges, the library management system provides a digital platform that manages all library operations efficiently and accurately. Librarians and members use the system. Librarians can add, remove and issue books, while members can search for and borrow available books. The main system flow includes adding books to the database, registering members, issuing, and returning books, and recording each transaction with date and time. The helps reduce manual errors, saves time and ensures accurate record-keeping for efficient library management.

B. Use-Case Summaries

- **UC-1: Borrowing books:** This use case allows members to borrow books. Library Members select books from the available books in the library. Librarian searches the book in the system. System verifies the availability of the book. Librarian issues the selected books to registered members. System records the issue date, due date and update the book's status to "Issued." The transaction is saved in the system for tracking and return purposes.
- **UC-2: Returning Books:** This use case allows members to return books. The member returns the book to the librarian. The librarian searches for the transaction record in the system using the member's ID or book ID. The system verifies the issue details and calculates any late fees (if applicable). The librarian confirms the return. The system updates the book's status to "Available" and records the return date.

C. Classes List

Class	**Purpose**	**Key Fields**	**Key Methods**
User (Abstract)	Base Class for every person	name:String, address:String, contact:int, email:String	getters/setters checkEmail(email:String):boolean
Member extends User	Library member who can search, borrow and return books	memberID:Strinng borrowedbooks:Book[] borrowedCount:int	borrowBook(), returnBook(), displayBorrowed(), displayDetails(), getMembetID(), getBorrowedBooks(), getBorrwoingHistory()
Librarian extends User	Manages operations such as adding/removing/issuing books etc.	librarianID:String	issueBook(),returnBook(), viewIssued(), addNewMember(), updateBookDetails(),getLibrarianID()
Book	Stores information about individual books.	bookID:String, title:String, ISBN:int, publisher:String, isAvailable:Boolean, author:String	Getters/setters getBook() – returns a book.
Library	Main class that manages books,	books: List<Book> members:List<Member>	searchBook(), searchMember(), addBook(), removeBook()

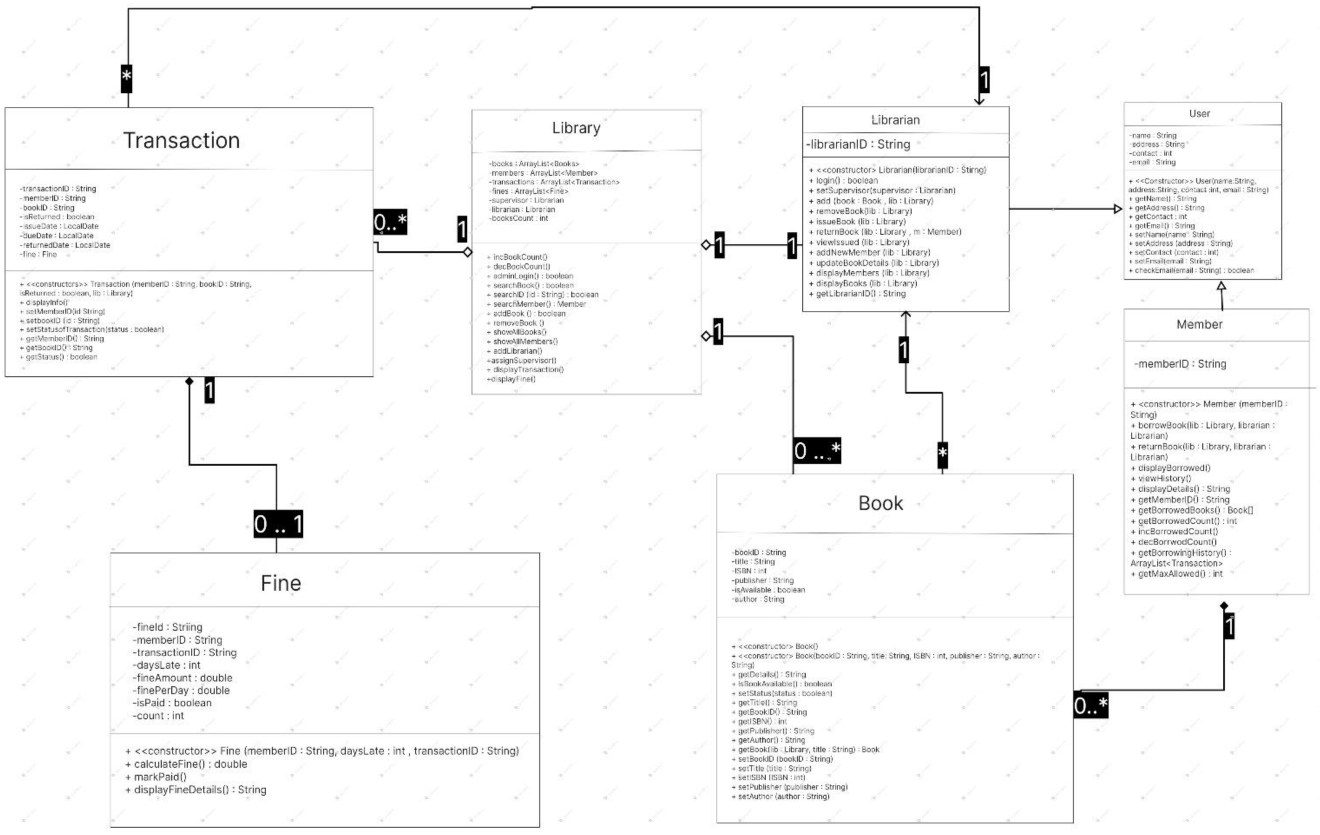
	members, librarians, transactions etc.	transactions:List<Transaction> booksCount:int	showAllBooks(), showAllMembers()
Transaction	Records each book issue and return details	transactionID:String, memberID:String, isbn:int, isReturned:Boolean, issueDate:LocalDate, dueDate:LocalDate, returnDate:LocalDate fine:Fine	Getters/Setters displayInfo()
Fine	Calculates and manages loans for late book returns	fineId:String, memberId:String, transactionID:String, daysLate:int fineAmount:double, isPaid:boolean	calculateFine(), markPaid(), displayFineDetails()

D. Relationships

A	**B**	**Type**	**Cardinality (A↔B)**	**Meaning**
User	**Member/Librarian**	Inheritance	-	Member and Librarian are specialized form of User Class
Library	**Book**	Aggregation	1 ↔ 0..*	Library has many books. Books can exist independently.
Library	**Member**	Aggregation	1 ↔ 0..*	Library has many registered members. Members can also exist independently as students.
Library	**Librarian**	Aggregation	1 ↔ 1	One Library has only one librarian. A Librarian can exist independently.
Library	**Transaction**	Composition	1 → *	Transactions are created and stored by library; they depend on it.
Member	**Book**	Aggregation	1 → 0.3	One member can have only 3 books at a time.
Transaction	**Fine**	Composition	1 ↔ 0..1	One transaction can have no or one fine. Fine can’t exist independently.
Transaction	**Member**	Association	* ↔ 1	Each transaction is linked to one member who borrowed a book.
Librarian	**Transaction**	Association	1 → *	A librarian generates transactions when issuing or returning books.

Librarian	**Book**	Association	1 → *	A librarian can add/remove/issue book to members.
Librarian	**Librarian**	Reflexive Association	1 → 1	Librarian has a supervisor who is also a Librarian.

E. UML Diagram



Transaction
-transactionID : String
-memberID : String
-bookID : String
-isReturned : boolean
-issueDate : LocalDate
-dueDate : LocalDate
-returnedDate : LocalDate
-fine : Fine
+ <<constructors>> Transaction (memberID : String, bookID : String, isReturned : boolean, lib : Library)
+ displayInfo()
+ setMemberID(id String)
+ setbookID (id : String)
+ setStatusofTransaction(status : boolean)
+ getMemberID() : String
+ getBookID() : String
+ getStatus() : boolean
Library
-books : ArrayList<Books>
-members : ArrayList<Member>
-transactions : ArrayList<Transaction>
-fines : ArrayList<Fine>
-supervisor : Librarian
-librarian : Librarian
-booksCount : int
+ incBookCount()
+ decBookCount()
+ adminLogin() : boolean
+ searchBook() : boolean
+ searchID (id : String) : boolean
+ searchMember() : Member
+ addBook () : boolean
+ removeBook ()
+ showAllBooks()
+ showAllMembers()
+ addLibrarian()
+assignSupervisor()
+ displayTransaction()
+displayFine()
Librarian
-librarianID : String
+ <<constructor> Librarian(librarianID : Stirng)
+ login() : boolean
+ setSupervisor(supervisor : Librarian)
+ add (book : Book , lib : Library)
+ removeBook(lib : Library)
+ issueBook (lib : Library)
+ returnBook (lib : Library , m : Member)
+ viewIssued (lib : Library)
+ addNewMember (lib : Library)
+ updateBookDetails (lib : Library)
+ displayMembers (lib : Library)
+ displayBooks (lib : Library)
+ getLibrarianID() : String
User
-name : String
-address : String
-contact : int
-email : String
+ <<Constructor>> User(name:String, address:String, contact :int, email : String)
+ getName() : String
+ getAddress() : String
+ getContact : int
+ getEmail() : String
+ setName(name : String)
+ setAddress (address : String)
+ setContact (contact : int)
+ setEmail(email : String)
+ checkEmail(email : String) : boolean
Member
-memberID : String
+ <constructor>> Member (memberID : Stirng)
+ borrowBook(lib : Library, librarian : Librarian)
+ returnBook(lib : Library, librarian : Librarian)
+ displayBorrowed()
+ viewHistory()
+ displayDetails() : String
+ getMemberID() : String
+ getBorrowedBooks() : Book[]
+ getBorrowedCount() : int
+ incBorrowedCount()
+ decBorrwodCount()
+ getBorrowingHistory() : ArrayList<Transaction>
+ getMaxAllowed() : int
Book
-bookID : String
-title : String
-ISBN : int
-publisher : String
-isAvailable : boolean
-author : String
+ <<constructor> Book()
+ <<constructor> Book(bookID : String, title: String, ISBN : int, publisher : String, author : String)
+ getDetails() : String
+ isBookAvailable() : boolean
+ setStatus(status : boolean)
+ getTitle() : String
+ getBookID() : String
+ getISBN() : int
+ getPublisher() : String
+ getAuthor() : String
+ getBook(lib : Library, title : String) : Book
+ setBookID (bookID : String)
+ setTitle (title : String)
+ setISBN (ISBN : int)
+ setPublisher (publisher : String)
+ setAuthor (author : String)
Fine
-fineId : Striing
-memberID : String
-transactionID : String
-daysLate : int
-fineAmount : double
-finePerDay : double
-isPaid : boolean
-count : int
+ <<constructor>> Fine (memberID : String, daysLate : int , transactionID : String)
+ calculateFine() : double
+ markPaid()
+ displayFineDetails() : String
*
1
0..*
1
1
1
1
0 ..*
1
*
1
0 .. 1
1
0..*