Problem Description: Library Management System OOA

A local library wants to build an automated **Library Management System** to manage its books, members, borrowing and returning of books, and overdue book fines. Currently, everything is managed manually, which leads to inefficiencies like misplaced books, delayed return records, and difficulty in tracking overdue fines.

The system must handle the following features:

1. User management
    1. Registration
    2. Authentication
    3. Authorization
    4. Password change
    5. Password uniqueness(old password can't be used again)
    6. Password complexity constraints
    7. Account verification
    8. Search history
    9. Profile management
    10. Profile picture
2. Implement the trash features
3. **Manage Books**: The library has a collection of books, and new books can be added, updated, or removed. Each book has details like title, author, ISBN, publication date, and availability status.
4. **Manage Members**: Library members are allowed to borrow books. Each member has personal details like name, ID, contact info, and a borrowing limit (e.g., 5 books at a time).
5. **Borrow and Return Books**: Members can borrow available books and return them within the borrowing period (e.g., 14 days). If they return a book late, an overdue fine must be calculated and applied.
6. **Track Book Availability**: The system must track which books are borrowed and which are available.
7. **Generate Reports**: The system should provide reports on borrowed books, available books, overdue books, and fines.
8. **Store the user search history(stack most recent searches)**

Commented [ae1]: Third

Commented [ae2]: daf

Commented [ae3]: ad

Object-Oriented Analysis (OOA) for the Library Management System

Now that we understand the problem, we will apply the steps of **Object-Oriented Analysis (OOA)** to create a solution.

---

1. Identify the Objects

Based on the problem description, we can identify the following objects in the system:

Key Objects:

- **Book**: Represents a book in the library's collection.
- **Member**: Represents a person who can borrow books.
- **Librarian**: The library staff who manages book records and member details.
- **BorrowRecord**: Represents the borrowing of a book by a member.
- **Fine**: Represents the fine applied to a member for overdue books.

---

2. Identify the Attributes

We define the attributes or properties of each object. These attributes represent the data that each object will hold.

Attributes:

- **Book**: Title, Author, ISBN, Publication Date, Status (Available/Borrowed).
- **Member**: Name, ID, Contact Info, Borrowing Limit, Borrowed Books.
- **Librarian**: Name, ID, Role (Manager, Assistant), Contact Info.
- **BorrowRecord**: Member, Book, Borrow Date, Return Date, Status (Borrowed/Returned).
- **Fine**: Amount, Due Date, Paid Status, Associated BorrowRecord.

---

3. Identify the Relationships

We now identify how the objects relate to each other. The relationships describe how these objects interact with one another to achieve the system's functionality.

Relationships:

- **Member-BorrowRecord**: A **Member** can create multiple **BorrowRecords** when borrowing books.
- **Book-BorrowRecord**: A **BorrowRecord** involves a **Book** that the member has borrowed.
- **BorrowRecord-Fine**: A **BorrowRecord** may generate a **Fine** if the book is returned late.
- **Librarian-Book**: A **Librarian** is responsible for managing **Books** (adding, updating, removing).

---

4. Identify the Behaviors (Methods)

Next, we define the behaviors or methods for each object. Methods represent the actions that the object can perform.

Behaviors (Methods):

- **Book**:
  - `checkAvailability()`: Check if the book is available for borrowing.
  - `markAsBorrowed()`: Mark the book as borrowed.
  - `markAsReturned()`: Mark the book as returned.
- **Member**:
  - `borrowBook()`: Borrow a book if within the borrowing limit.
  - `returnBook()`: Return a borrowed book.
  - `viewBorrowedBooks()`: View the list of books currently borrowed.
- **Librarian**:
  - `addBook()`: Add a new book to the collection.
  - `removeBook()`: Remove a book from the collection.
  - `updateBookDetails()`: Update details (e.g., status, title) of an existing book.
- **BorrowRecord**:
  - `createRecord()`: Create a new record when a book is borrowed.
  - `updateRecord()`: Update the record when a book is returned.
- **Fine**:
  - `calculateFine()`: Calculate the fine based on the number of overdue days.
  - `markAsPaid()`: Mark the fine as paid.

---

5. Model the Objects and Relationships

Next, we create a visual representation of the objects, their attributes, methods, and relationships using a **class diagram**.

scss
Copy code

```
+---------------+              +---------------+              +------------
---+
|     Member    |              |      Book     |              |   Librarian
|
+---------------+              +---------------+              +------------
---+
| Name          |              | Title         |              | Name
|
| ID            |              | Author        |              | ID
|
| Contact Info  |              | ISBN          |              | Role
|
| BorrowingLimit|              | Status        |              | Contact
Info   |
| BorrowedBooks |              +---------------+              +------------
---+
+---------------+              | checkAvailability()          | addBook()
|
| borrowBook()  |              | markAsBorrowed()              |
removeBook()   |
| returnBook()  |              | markAsReturned()              |
updateBook()   |
| viewBorrowedBooks()          +---------------+              +------------
---+
+---------------+                      |                            |
                                        |                            |
                                        v                            |
+---------------+                    |
| BorrowRecord  |<-------------------+
+---------------+
| Borrow Date   |
| Return Date   |
| Status        |
+---------------+
| createRecord()|
| updateRecord()|
+---------------+
       |
       v
+---------------+
|     Fine      |
+---------------+
| Amount        |
| Due Date      |
| Paid Status   |
+---------------+
| calculateFine()|
| markAsPaid()   |
+---------------+
```

---

6. Define the System's Responsibilities

Here, we assign specific tasks or responsibilities to each object and define how they interact with other objects.

- **Member**: Responsible for borrowing and returning books, and viewing their borrowed books.
- **Book**: Responsible for tracking its availability status (borrowed or available).
- **Librarian**: Responsible for managing the library’s collection (adding, removing, updating books).
- **BorrowRecord**: Responsible for recording and updating information about book borrowing and returning.
- **Fine**: Responsible for calculating overdue fines and tracking their payment status.
