**Use Case for: LIBRARY MANAGEMENT SYSTEM**

A library management system is a software tool that helps libraries organize and manage their resources efficiently. It simplifies tasks such as cataloging, circulation, and inventory management, making it easier for librarians to handle day-to-day operations. Here's a breakdown structure of a library management system:

1. **User Interface**:

User Registration: The system allows users, such as students, faculty, and staff, to register and obtain a unique identifier, such as a library card or username, to access library services.

Login/Authentication: Allows library staff to securely log in to the system using their credentials.

Dashboard: Provides an overview of the system and access to different modules and features.

Search and Browse: Enables users to search and browse the library catalog by title, author, keywords, or other criteria.

Borrowing and Returning: Facilitates the borrowing and returning of library materials by users, including checking availability and managing due dates.

User Management: Allows librarians to create and manage user accounts, update user information, and handle user-related operations.

2) **Cataloging and Inventory Management**:

This module is responsible for creating and maintaining a catalog of library resources, including books, journals, multimedia, and other materials. Each item is assigned a unique identifier and relevant information, such as title, author, subject, and publication details.

Item Entry: Provides a feature to enter new books or other materials into the library catalog, including information such as title, author, publisher, ISBN, and subject.

Classification and Categorization: Enables librarians to assign appropriate categories, subjects, and keywords to each item for easier searching and browsing.

Barcode Generation: Generates unique barcodes for each item to streamline circulation and inventory management processes.

Inventory Tracking: Tracks the location, availability, and status of library materials, including information on borrowed items, reserved items, and their due dates.

3) **Circulation Management:**

The circulation module manages the borrowing and returning of library items. It tracks the availability of items, handles due dates, and generates reminders or penalties for late returns.

Check Out/In: Allows librarians to check out items to borrowers and record due dates, while also facilitating the return process and managing late fees, if applicable.

Reservation and Holds: Enables users to reserve items that are currently unavailable and notifies them when the items become available for borrowing.

Renewals: Provides the option for borrowers to renew their borrowed items for an extended loan period, if allowed by library policies.

Overdue Management: Sends notifications and generates reports for overdue items, managing fines, and handling any necessary follow-up actions.

4) **Search and Discovery:**

Users can search the library's collection using keywords, author names, or subject categories.

Search functionality: The system should provide a powerful search function that allows users to search for resources based on various criteria, such as title, author, subject, keywords, or a combination of these.

Browsing and recommendations: The system should include browsing features that allow users to explore the library's collection by categories, genres, or other predefined criteria. Additionally, personalized recommendations can be generated based on the user's borrowing history, reading preferences, or similar users' behavior, suggesting relevant resources they may be interested in.

User-friendly interface: The search and discovery interface should be intuitive and user-friendly, enabling users to easily navigate and interact with the library management system. It gives organized presentation of search results with detailed resource information.

4) **Reporting and Analytics:**

The system generates various reports and statistics to help librarians analyze library usage, track overdue items, and make informed decisions for resource allocation.

Transaction Reports: Generates reports on circulation activities, including checkouts, returns, and renewals, to help analyze library usage and trends.

Financial Reports: Generates reports on fines collected, payments, and other financial aspects to assist in managing library finances.

Benefits:

**PROBLEM STATEMENT: LIBRARY MANAGEMENT SYSTEM**

Suppose you have a class called Library that contains a list of books. Each book has a title, author, and number of pages. Implement the Library class with the following methods:

add\_book(title, author, num\_pages): adds a new book to the library with the given title, author, and number of pages.

remove\_book(title): removes a book from the library with the given title.

get\_books\_by\_author(author): returns a list of books by the given author.

get\_total\_pages(): returns the total number of pages in the library.

Using the Library class, perform the following operations:

Add at least 5 books to the library.

Use map to create a new list of books that contains only the titles of the books.

Use filter to create a new list of books that contain more than 300 pages.

Use reduce to calculate the total number of pages in the library.

Use the get\_books\_by\_author method to get a list of books by a specific author

**PROGRAM FOR LIBRARY MANAGEMENT SYSTEM**

print(" LIBRARY MANAGEMENT SYSTEM ")

from functools import reduce

print("")

book\_list=[ ]

class Library:

def add\_book(self):

b=[ ]

title=input("Enter book title name :")

author=input("Enter book author name :")

num\_page=int(input("Enter number of pages in book :"))

b.append(title)

b.append(author)

b.append(num\_page)

book\_list.append(b)

print("--------------------------------------------")

def get\_books\_by\_author(self):

author\_name=input("Enter name of author :")

p=0

for i in book\_list:

if author\_name in i:

print(f"{i[0]} book is written by {author\_name}")

p=1

if p==0:

print("book with this author name is not present")

print("-------------------------------------------")

def total\_pages(self):

num\_list=[]

for j in book\_list:

num\_list.append(j[2])

def addition(x,y):

return x+y

total=reduce(addition,num\_list)

print("total number of pages in library : ", total)

print("------------------------------------------")

def table(self):

print("book\_names author\_names pages")

print("\_\_\_\_\_\_\_\_\_\_\_\_")

for i in book\_list:

print(f"{i[0]} | {i[1]} | {i[2]}")

print("\_\_\_\_\_\_\_\_\_")

lib=Library()

while True:

# Displaying the option menu

print("Enter 1 to add new book")

if len(book\_list) >= 1:

print("Enter 2 to display the book name")

if len(book\_list)>= 1:

print("Enter 3 to display the total number of pages in library")

if len(book\_list)>= 1:

print("Enter 4 to display the table of books")

inp = input("Enter your choice :- ")

if int(inp) == 1:

lib.add\_book()

elif int(inp)==2 and len(book\_list) >= 1:

lib.get\_books\_by\_author()

elif int(inp)==3 and len(book\_list)>= 1:

lib.total\_pages()

elif int(inp)==4 and len(book\_list)>= 1:

lib.table()

else:

print("Invalid Choice, Exiting from the program")

break

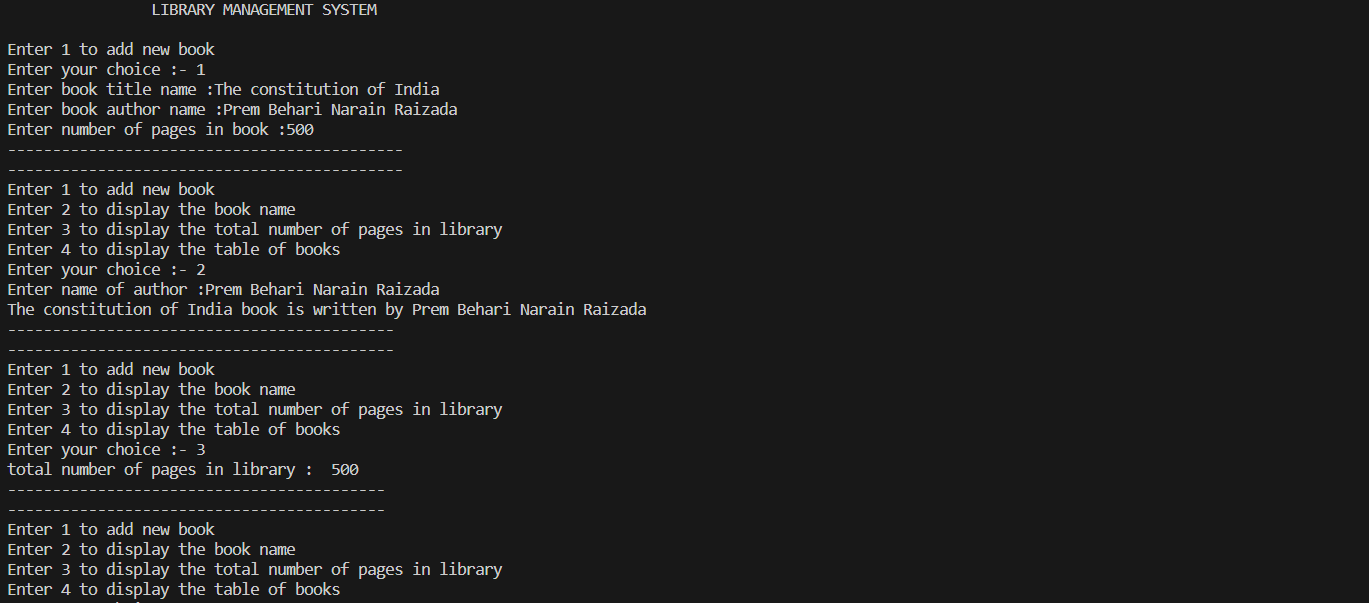
**Explanation of the code:**

This program is a basic implementation of a Library Management System. It allows users to perform various operations related to managing a library. Here's a brief explanation of the program:

1. The program starts by printing the title and a separator line.
2. The **‘book\_list’** variable is initialized as an empty list to store book information.
3. The **‘Library’** class is defined, which contains methods to perform different operations.
4. The **‘add\_book’** method allows the user to add a new book by providing the title, author name, and number of pages. The book details are stored as a list and appended to the **‘book\_list’**.
5. The **‘get\_books\_by\_author’** method prompts the user to enter an author's name and then searches the **‘book\_list’** for books written by that author. It prints the book titles along with the author's name.
6. The **‘total\_pages’** method calculates and prints the total number of pages in the library by summing up the number of pages for each book in the **‘book\_list’**.
7. The **‘table’** method displays a table of all the books in the book\_list, showing the **‘book title’**, author name, and number of pages.
8. An instance of the **‘Library’** class,**’lib’**, is created.
9. The program enters a while loop to continuously display the option menu and perform operations based on user input.
10. If the **‘book\_list’** is not empty, the user can choose options 2, 3, and 4 to display books by author, total number of pages, and the table of books, respectively.
11. The user can choose option 1 to add a new book by calling the **‘add\_book’** method.
12. If the user enters an invalid choice or selects an option without any books in the library, the program terminates.

Overall, this program provides basic functionality for adding books, searching books by author, calculating total pages, and displaying book information in a table format.

**Output of LIBRARY MANAGEMENT code:**

****