# Library Management System

## 1. Introduction

**This project demonstrates a simple Library Management System using the Object-Oriented Programming (OOP) principles in Python. It allows the library to manage books and members, as well as operations such as adding, searching, borrowing, returning, and deleting books. The system uses classes, objects, methods, and dictionaries to store and manipulate data efficiently.**

## 2. System Overview

**The system includes:  
- Books — each identified by an ISBN, title, author, and genre.  
- Members — each with an ID, name, and email address.  
- Library — manages all books and members, and handles borrowing/returning.**

## 3. Data Structures Used

|  |  |  |
| --- | --- | --- |
| **Structure** | **Description** | **Example** |
| **Dictionary** | **Used to store book and member details in key-value pairs** | **books = {isbn: {"title": ..., "author": ...}}** |
| **List** | **Used to store borrowed books for each member** | **member.borrowed\_books = []** |
| **Class** | **Used to represent Books, Members, and the Library itself** | **class Book, class Member, class Library** |

## 4. Python Code Implementation

**class Book:  
 def \_\_init\_\_(self, isbn, title, author, genre, total\_copies):  
 self.isbn = isbn  
 self.title = title  
 self.author = author  
 self.genre = genre  
 self.total\_copies = total\_copies  
 self.available\_copies = total\_copies  
  
class Member:  
 def \_\_init\_\_(self, member\_id, name, email):  
 self.member\_id = member\_id  
 self.name = name  
 self.email = email  
 self.borrowed\_books = []  
  
class Library:  
 def \_\_init\_\_(self):  
 self.books = {}  
 self.members = {}  
  
 def add\_book(self, isbn, title, author, genre, total\_copies=1):  
 if isbn in self.books:  
 return f"Book with ISBN {isbn} already exists."  
 self.books[isbn] = Book(isbn, title, author, genre, total\_copies)  
 return f"Book '{title}' added successfully."  
  
 def add\_member(self, member\_id, name, email):  
 if member\_id in self.members:  
 return f"Member ID {member\_id} already exists."  
 self.members[member\_id] = Member(member\_id, name, email)  
 return f"Member '{name}' added successfully."  
  
 def search\_book(self, keyword):  
 result = [book.title for book in self.books.values() if keyword.lower() in book.title.lower()]  
 return result if result else "No books found."  
  
 def borrow\_book(self, member\_id, isbn):  
 if member\_id not in self.members:  
 return "Member not found."  
 if isbn not in self.books:  
 return "Book not found."  
  
 member = self.members[member\_id]  
 book = self.books[isbn]  
  
 if book.available\_copies <= 0:  
 return f"'{book.title}' is not available."  
 if isbn in member.borrowed\_books:  
 return f"'{book.title}' is already borrowed by {member.name}."  
  
 book.available\_copies -= 1  
 member.borrowed\_books.append(isbn)  
 return f"'{book.title}' borrowed successfully by {member.name}."  
  
 def return\_book(self, member\_id, isbn):  
 if member\_id not in self.members:  
 return "Member not found."  
 if isbn not in self.books:  
 return "Book not found."  
  
 member = self.members[member\_id]  
 book = self.books[isbn]  
  
 if isbn not in member.borrowed\_books:  
 return f"{member.name} did not borrow '{book.title}'."  
  
 member.borrowed\_books.remove(isbn)  
 book.available\_copies += 1  
 return f"'{book.title}' returned successfully."  
  
 def delete\_book(self, isbn):  
 if isbn not in self.books:  
 return "Book not found."  
 del self.books[isbn]  
 return f"Book with ISBN {isbn} deleted successfully."**

demo.py

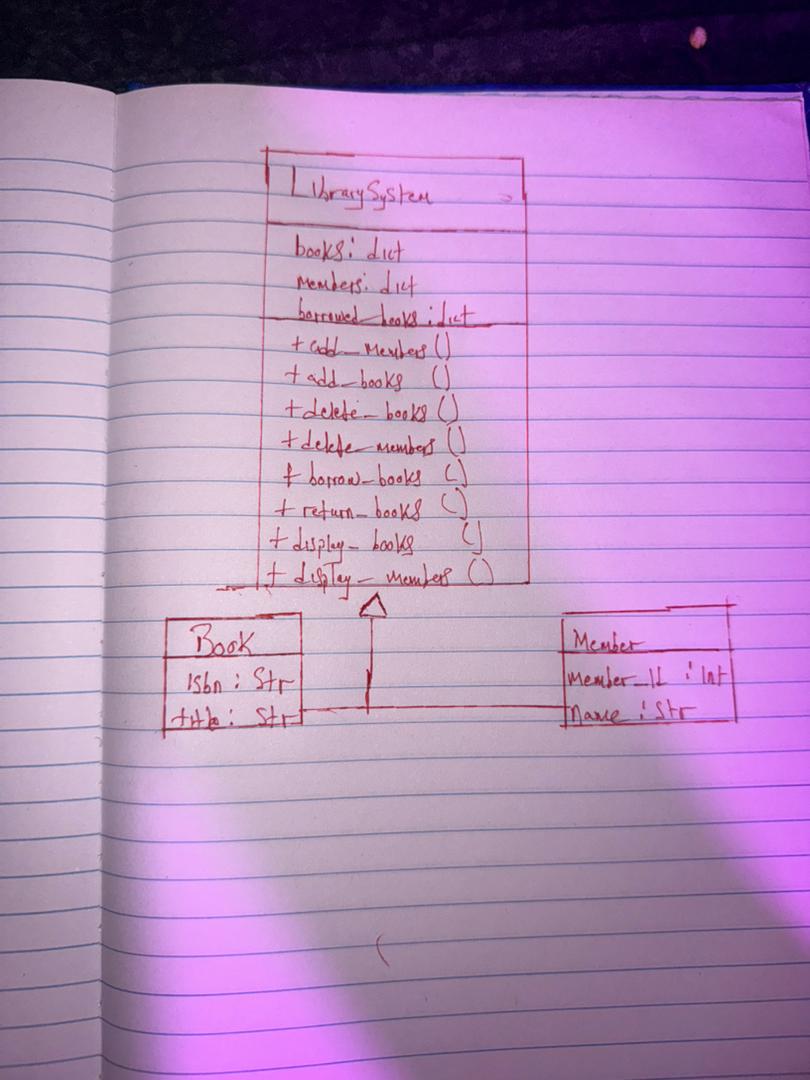
**from operations import Library  
  
library = Library()  
  
print(library.add\_book("001", "Python for Everyone", "John Doe", "Fiction", 3))  
print(library.add\_book("002", "World History", "Jane Ray", "History", 2))  
print(library.add\_book("003", "Narrative Writing", "Paul Jones", "Narrative", 4))  
print(library.add\_book("004", "The Great Novel", "Sam Kamara", "Novel", 1))  
  
print(library.add\_member(10, "Emmanuel Kallon", "emmanuel@example.com"))  
print(library.add\_member(41, "Samuel Siaka", "samuel@example.com"))  
print(library.add\_member(25, "Abu", "abu@example.com"))  
  
print("\nSearching for 'Python':", library.search\_book("Python"))  
  
print(library.borrow\_book(10, "001"))  
print(library.return\_book(10, "001"))  
  
print(library.delete\_book("004"))**

test\_library.py

**from operations import Library  
  
lib = Library()  
  
lib.add\_book("101", "Python Basics", "John Doe", "Education", 2)  
lib.add\_book("102", "Future World", "Jane Ray", "Sci-Fi", 1)  
lib.add\_member(10, "Emmanuel Kallon", "emmanuel@example.com")  
  
assert lib.borrow\_book(10, "101")  
assert lib.return\_book(10, "101")  
assert lib.delete\_book("102")  
  
print(" All tests is complete!")**

## 5. UML Class Diagram Description

**The UML diagram contains three main classes:  
  
- Book: Attributes: isbn, title, author, genre, total\_copies, available\_copies  
- Member: Attributes: member\_id, name, email, borrowed\_books  
- Library: Attributes: books (dictionary), members (dictionary)  
 Methods: add\_book(), add\_member(), search\_book(), borrow\_book(), return\_book(), delete\_book()**

****

## 6. Conclusion

**This system models a small library with book and member management features. It uses object-oriented programming to organize code efficiently, improve readability, and make maintenance easier. It can later be expanded with a database or a graphical interface for real-world use.**