**TABLE OF CONTENT**

Introduction…………………………………………………...Page1

Objectives………………………………………………….…..Page1

File structure………………………………………………....Page1

Module documentation…………………….……………..Page2-Page7

System design summary………………………………...Page8

UML diagram……………………………………………….Page.9

Conclusion………………………………………………..Page10

References……………………………………………….Page11

**LIBRARY MANAGEMENT SYSTEM PROJECT DOCUMENTATION**

**1. INTRODUCTION**

This project is a simple **Library Management System** developed using Python programming.  
It demonstrates the use of **functions**, **data structures**, and **modular programming** to manage books and members in a small-scale library environment.

The system allows users to add, update, delete, borrow, and return books while maintaining member records and borrowing limits.

**2. OBJECTIVES**

The main objectives of this project are:

1. To design a program that automates basic library operations.
2. To apply programming concepts such as dictionaries, lists, and functions in Python.
3. To demonstrate code reusability through separate modules.
4. To implement simple testing and demonstration scripts.

**3. FILE STRUCTURE**

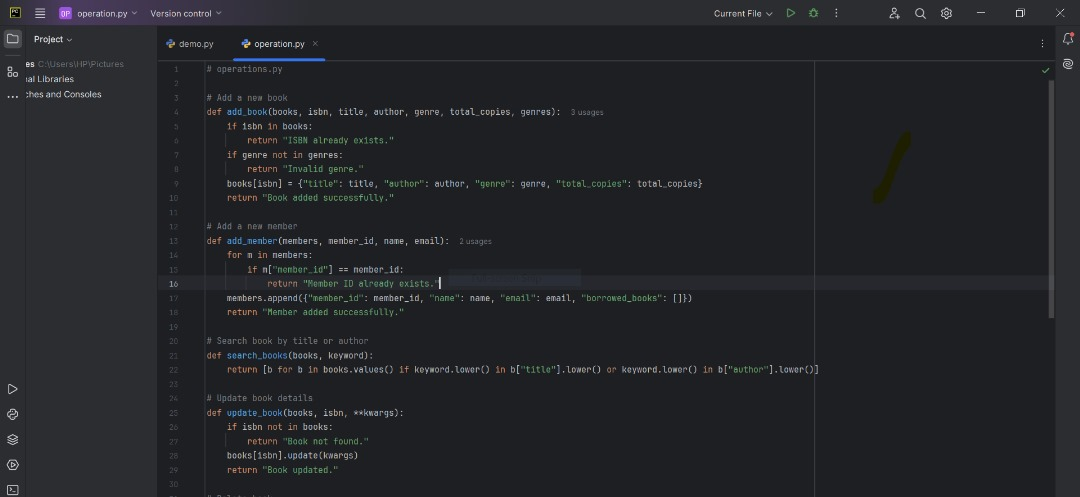
The project consists of three Python files:

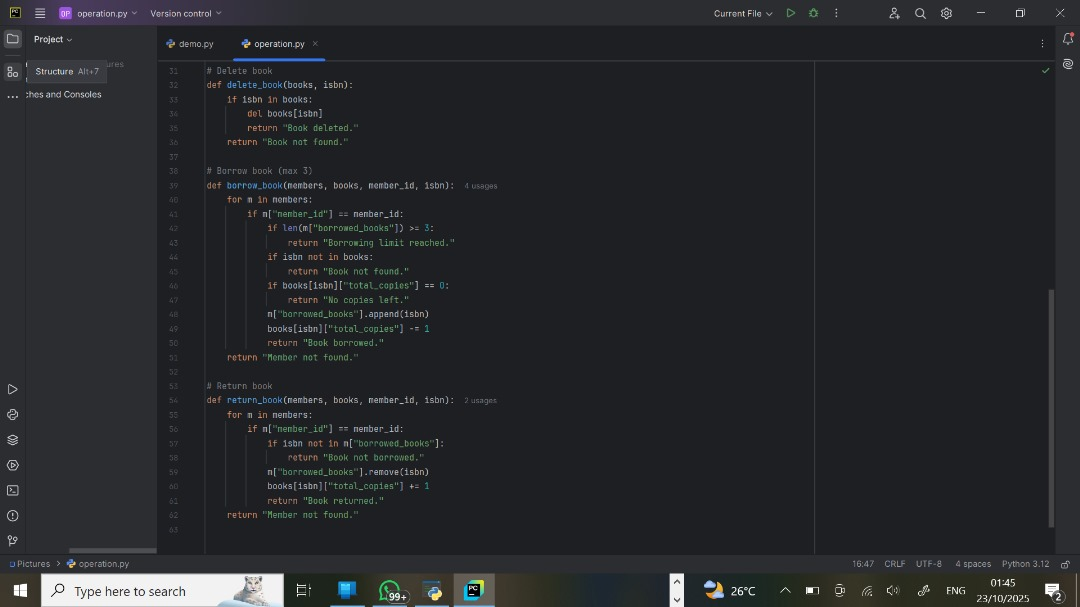
|  |  |
| --- | --- |
| **File Name** | **Purpose** |
| operation.py | Contains all main functions that perform book and member operations. |
| test.py | Tests each function to ensure they behave as expected. |
| demo.py | Demonstrates how the system can be used in real life. |

**4. MODULE DOCUMENTATION**

**4.1 OPERATION.PY**

This is the **core logic module** of the project.  
It defines functions for adding books, managing members, borrowing, returning, and searching for books.



****

**Purpose**

To manage all library operations logically and efficiently.

**Main Functions**

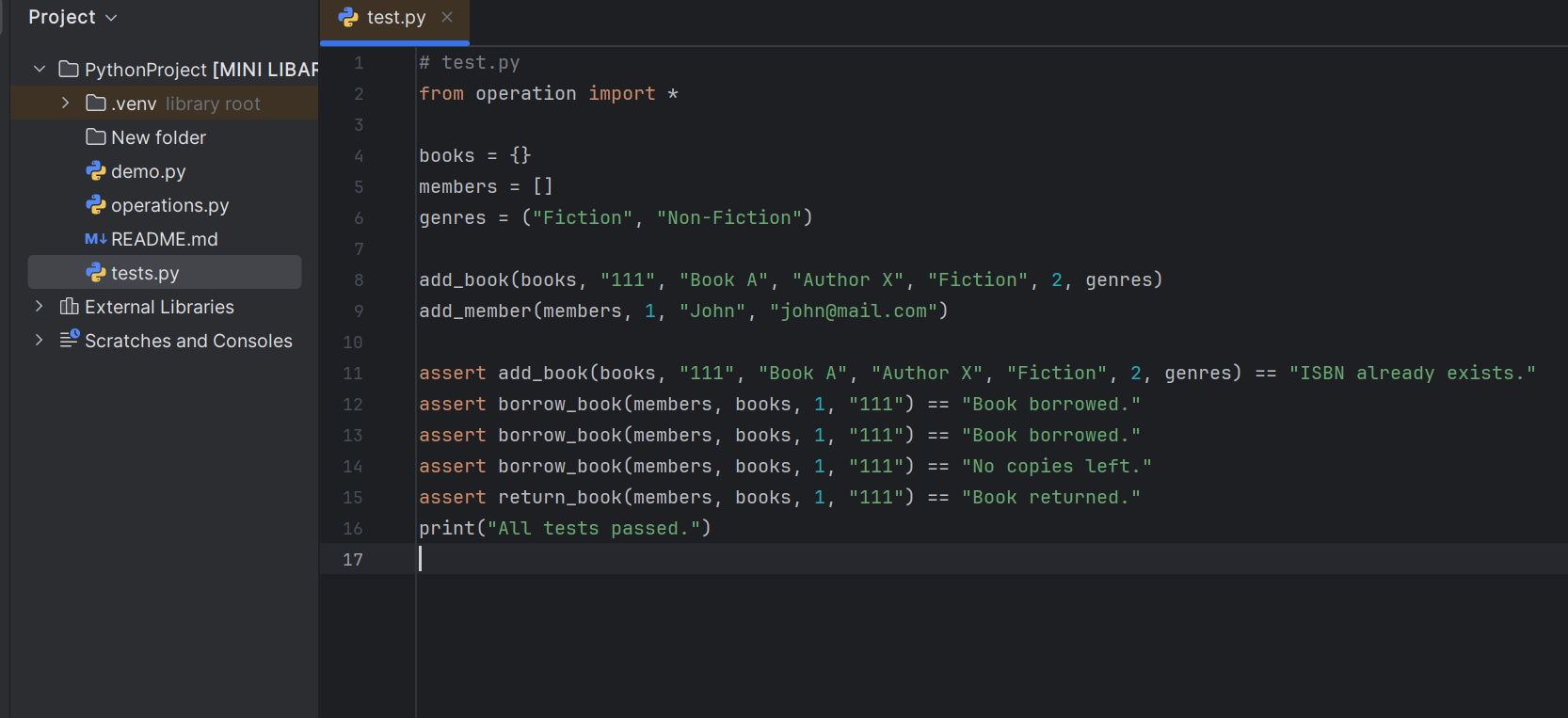
* add\_book() – Adds a new book and validates ISBN and genre.
* add\_member() – Registers new members and prevents duplicates.
* search\_books() – Finds books by title or author.
* update\_book() – Updates existing book details.
* delete\_book() – Deletes a book from the library.
* borrow\_book() – Allows members to borrow up to three books.
* return\_book() – Returns borrowed books and updates copies.

**Rationale**

This file implements the main functionality using a **procedural programming approach**, keeping data and operations simple yet structured.

**4.2 TEST.PY**

This module tests all the functions defined in operation.py to ensure they are working properly.

. 

**Purpose**

To confirm that each operation (adding, borrowing, returning) works as expected without errors

**Process**

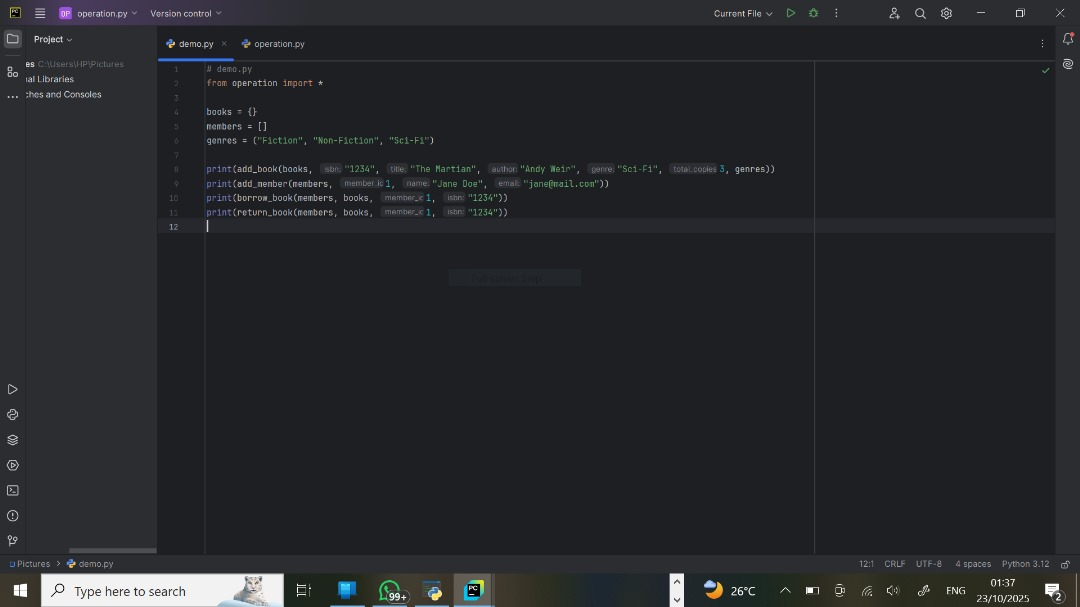
* Imports functions from operation.py
* Creates temporary dictionaries for books and members
* Uses assert statements to check expected results
* Prints *“All tests passed”* if successful

**Rationale**

This file ensures **accuracy, reliability, and error handling** in the core functions.  
It also helps detect problems early before using the system in real scenarios.

**4.3 DEMO.PY**

This module demonstrates how the system can be used in a practical way.  
It simulates adding a book, registering a member, borrowing, and returning the book.

****

**Purpose**

To give a real example of how the functions operate together in sequence.

**Operations Demonstrated**

1. Adding a new book
2. Adding a member
3. Borrowing a book
4. Returning a borrowed book

**Sample Output**

Book added successfully.

Member added successfully.

Book borrowed.

Book returned.

**Rationale**

This script helps visualize how the library management system works when executed. It’s a simple demonstration that complements the logical and test modules.

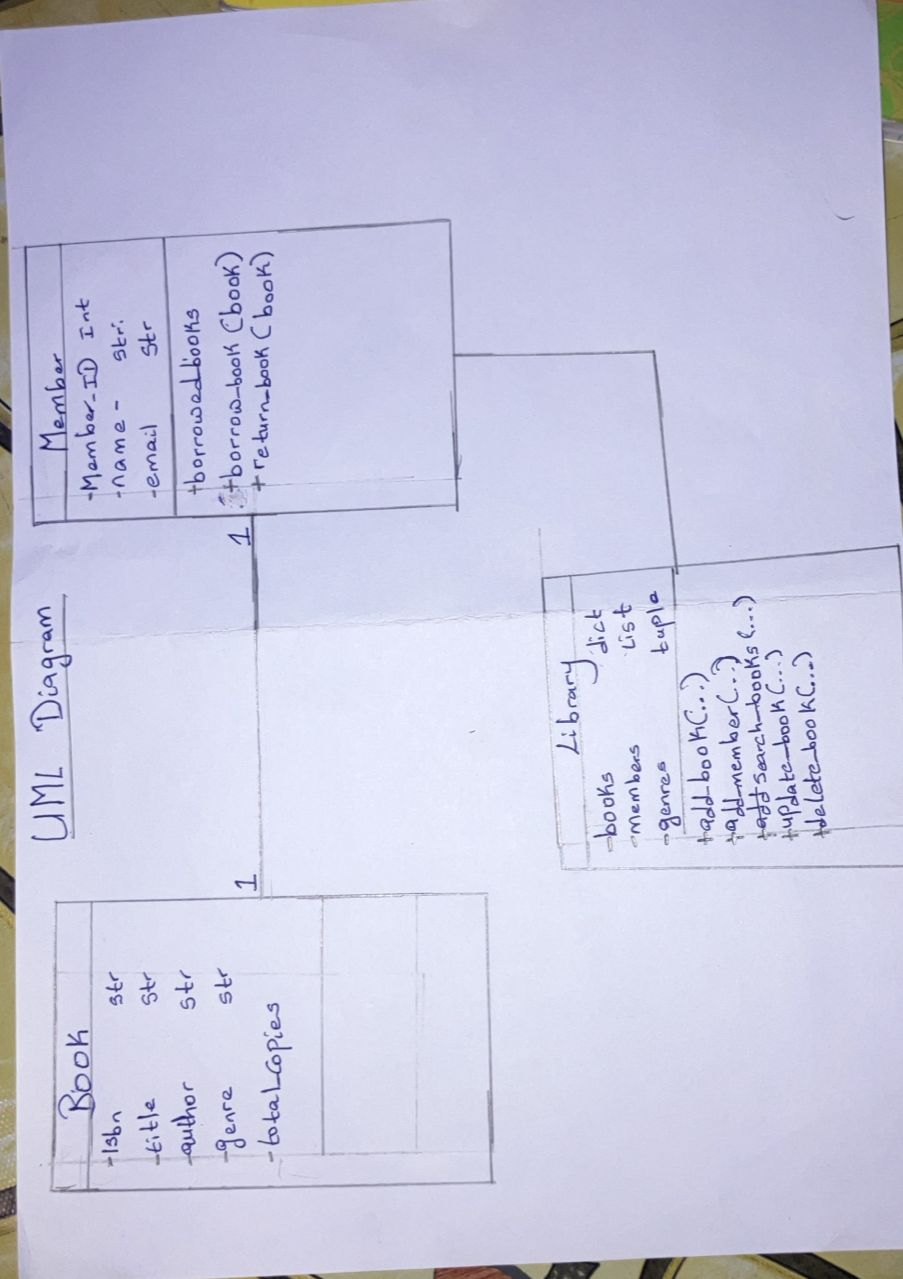
**5. SYSTEM DESIGN SUMMARY**

The entire project is structured using **three layers**:

|  |  |  |
| --- | --- | --- |
| **Layer** | **File** | **Function** |
| Logic Layer | operation.py | Handles the core operations and data management. |
| Testing Layer | test.py | Verifies correctness and stability of all functions. |
| Demonstration Layer | demo.py | Shows how the system works interactively. |

This structure promotes **code organization**, **testing reliability**, and **easy maintenance**.

**6.UML DIAGRAM**

**7. CONCLUSION**

The Library Management System provides an efficient and modular approach to handling library activities.  
It demonstrates the practical use of Python’s data structures and functions to perform everyday management tasks.

Through **testing** and **demonstration scripts**, this project highlights good software development practices modularity, validation, and functional decomposition.

**8. REFERENCES**

Python Official Documentation: https://docs.python.org/3/

W3Schools Python Tutorial: https://www.w3schools.com/python/

GeeksforGeeks – Python Dictionaries & Functions: https://www.geeksforgeeks.org/