Library Management System

A Python Console Project

Submitted by:

Team of Three Members

Under Toolkit for Research Project

Contents

1 Introduction

The Library Management System is a simple Python-based console application designed to manage a collection of books. It allows users to add, view, search, and delete book records, ensuring that data is stored persistently using a JSON file.

This project demonstrates the use of Python's core concepts like functions, file handling, and modular programming.

2 Objective

- To provide a basic and easy-to-use system for library data management.
- To apply Python programming concepts such as file handling, data structures, and modular coding.
- To help students understand how to split a project among multiple team members using GitHub.

3 Team Division

Member	**Files Assigned**
Member 1	main.py, book_operations.py
Member 2	manage_books.py, helpers.py
Member 3	storage.py, utils.py

4 System Design

The system is menu-driven and allows users to:

1. Add a new book with title, author, and year.
2. View all existing books.
3. Search for a book by title or author.
4. Delete a book using its unique ID.

5. Save all records into a JSON file.

5 Code Implementation

Below are key code snippets from the project.

5.1 Main Menu (main.py)

```python
from book_operations import add_book, view_books
from manage_books import search_book, delete_book
from storage import save_data, load_data

def main():
    books = load_data()

    while True:
        print("\n=== Library Management System ===")
        print("1. Add Book")
        print("2. View Books")
        print("3. Search Book")
        print("4. Delete Book")
        print("5. Save & Exit")

        choice = input("Enter your choice: ")

        if choice == ’1’:
            add_book(books)
        elif choice == ’2’:
            view_books(books)
        elif choice == ’3’:
            search_book(books)
        elif choice == ’4’:
            delete_book(books)
        elif choice == ’5’:
            save_data(books)
            print("Data saved! Exiting...")
            break
        else:
            print("Invalid choice, try again!")
```

Listing 1: main.py

5.2 Book Operations (book_operations.py)

```
from utils import generate_id

def add_book(books):
    title = input("Enter book title: ")
    author = input("Enter author name: ")
    year = input("Enter publication year: ")
    book_id = generate_id()

    book = {"id": book_id, "title": title, "author": author, "year": year}
    books.append(book)
    print("Book added successfully!")

def view_books(books):
    if not books:
        print("No books found in the library.")
        return
    for b in books:
        print(f"ID: {b[’id’]} | {b[’title’]} by {b[’author’]} ({b[’year’]})")
```

Listing 2: book$_{o}$*perations.py*

6 Data Visualization

The following charts show example data analysis from the library system.

6.1 Bar Graph — Number of Books by Year

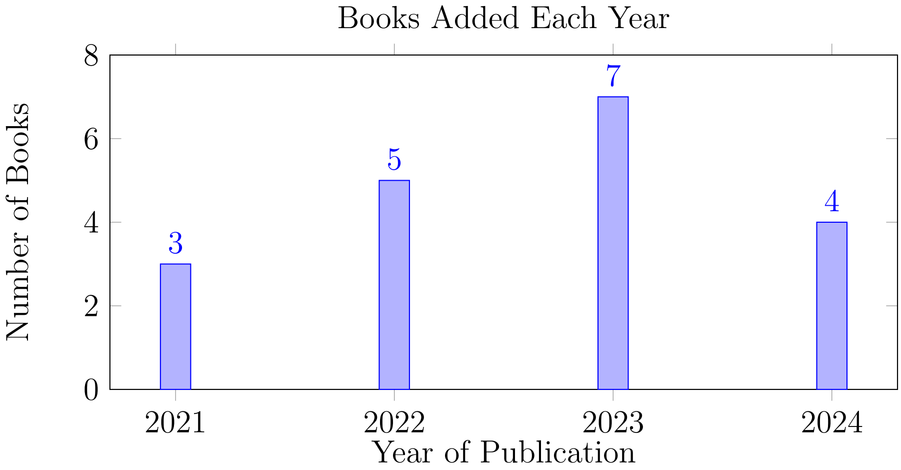

Figure 1: Bar graph showing number of books added per year.

6.2 Pie Chart — Distribution of Books by Author

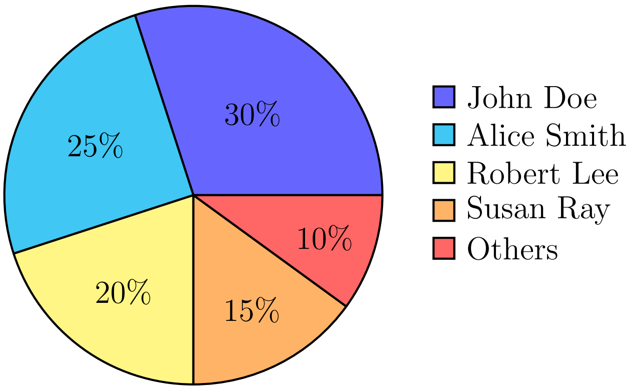

Figure 2: Pie chart showing distribution of books by author.

7 Results

The program successfully performs:

- Adding and viewing books.
- Searching and deleting by ID.
- Storing data persistently using JSON.

8 Conclusion

This project demonstrates modular Python programming and team collaboration using Git and GitHub. It provided practical experience in handling data, implementing file storage, and presenting visual results.

9 Future Enhancements

- Add GUI using Tkinter.
- Include admin and user login system.
- Add book borrowing and return feature.