Library Management System

Detailed Project Report

Piyush Raj Singh

November 3, 2025

Abstract

This report presents a detailed overview of the Library Management System developed as a collaborative Git project. The report highlights the problem statement, proposed solution, code explanation, visualization of data, conclusion, and potential future improvements. The system is designed to streamline library operations such as book entry, search, view, and deletion in an efficient, modular, and scalable way.

Contents

1 Problem Statement

Libraries often struggle with maintaining a digital and structured catalog for their books. Manual record-keeping is prone to errors, inefficient retrieval, and data loss. The goal of this project is to develop a simple yet scalable digital system to:

- Maintain records of books efficiently.
- Enable users to add, search, and remove books easily.
- Create a foundation for larger future systems (e.g., web-based library platforms).

2 Solution Overview

The Library Management System is a console-based Python program that performs CRUD operations on a list of books. Each book is represented as a dictionary containing ID, title, author, and year. The system supports:

1. Adding new books with a unique ID.
2. Viewing all books stored in memory.
3. Searching books by title or author.
4. Deleting books by their ID.

All code is modularized into separate Python files and version-controlled using Git, allowing smooth collaboration and updates.

3 Code Explanation

The following functions are part of the core implementation pushed to GitHub as part of the project's backend:

```
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
    print("\n--- Library Books ---")
    for b in books:
        print(f"ID: {b['id']} | {b['title']} by {b['author']} ({b
            ['year']})")

def search_book(books):
    keyword = input("Enter title or author to search: ").lower()
    results = [b for b in books if keyword in b["title"].lower()
       or keyword in b["author"].lower()]
    if results:
        print("\n--- Search Results ---")
        for b in results:
            print(f"{b['title']} by {b['author']} ({b['year']})")
    else:
        print("    No books found matching your search.")

def delete_book(books):
    book_id = input("Enter book ID to delete: ")
    for b in books:
        if b["id"] == book_id:
            books.remove(b)
            print("       Book deleted successfully!")
            return
    print("    No book found with that ID.")
```

Explanation

- **add_book:** Accepts book details from the user, generates a unique ID, and stores it in a list.
- **view_books:** Displays all stored books in a formatted list.
- **search_book:** Finds matches based on user-entered title or author keywords.
- **delete_book:** Removes a book record using its ID.

These functions represent the essential logic that can later be integrated into a GUI or web-based interface.

4 Visualization

A visualization is included to show an example of how data can be represented, such as the number of books added each year.

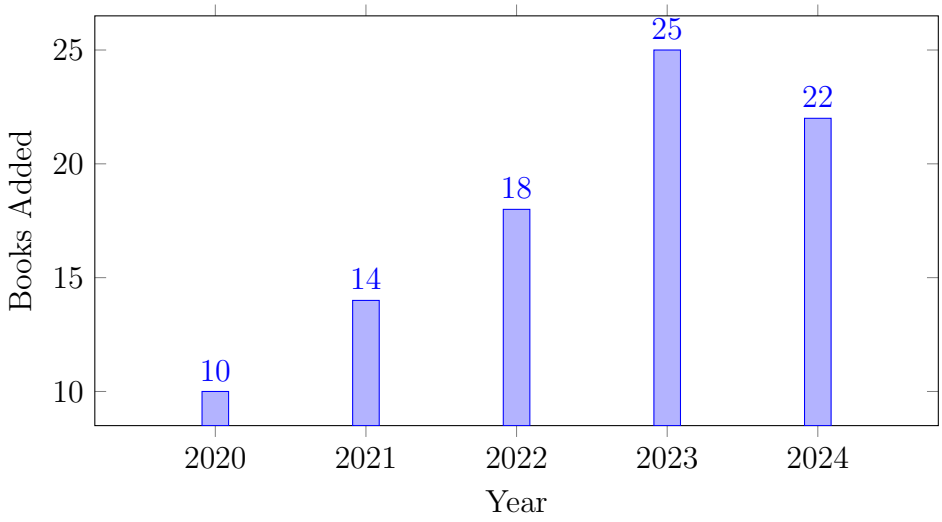

Figure 1: Sample visualization: Books added per year (example data)

5 Conclusion

The Library Management System effectively addresses the problem of manual book record handling by providing a simple yet functional digital solution. The modular code allows future integration with databases and front-end systems, and Git version control ensures collaborative scalability.

6 Future Improvements

To enhance this project further, the following steps can be taken:

- Implement a persistent storage system using JSON, CSV, or SQL.
- Add a user authentication system for librarians.
- Create a graphical interface using Tkinter or a web framework like Flask.
- Integrate automated testing and CI/CD workflows.
- Visualize data dynamically using libraries such as Matplotlib or Plotly.