Digital Library Management System
Project Report

Submitted By: Kunal
Roll Number: 25BCE11146
Course: INTRODUCTION TO PROBLEM SOLVING USING PYTHON
Instructor: Dr.Lokesh Malviya, Dr.Lakshmi D.

Submission Date: 24-11-25

1 Introduction

A **Digital Library Management System** is a software application that allows users to manage books and users efficiently. It enables the librarian to add, delete, and manage books while allowing users to borrow and return books digitally, eliminating the need for manual record-keeping.

This system improves accessibility, reduces errors, and provides real-time tracking of book availability.

2 Problem Statement

Manual library management is time-consuming and prone to errors. Tracking issued books, handling overdue returns, and maintaining accurate records are difficult without automation.

Problem: To develop a digital system that automates book and user management, making library operations faster, more accurate, and convenient.

3 Functional Requirements

1. Add new books with details (ID, title, author).
2. Add new users with details (ID, name).
3. View all books and their availability.
4. Search books by title.
5. Borrow and return books.
6. Delete books that are not currently issued.
7. Persist data using JSON files.

4 Non-functional Requirements

1. **Usability:** Simple text-based interface.
2. **Reliability:** Data stored in JSON to prevent loss.
3. **Performance:** Fast search and retrieval of book records.
4. **Maintainability:** Modular functions for easy code updates.
5. **Portability:** Runs on any system with Python installed.

5 System Architecture

Architecture Type: Client-Side Python Application with File Storage

Description:

- The system follows a **modular design**, separating functionality into functions like addbook(), borrowbook(), returnbook(), and searchbook().
- Data storage is handled using JSON files (books.json and users.json).
- The menu-driven interface allows the user to interact with the system.

Architecture Diagram:

User Input → Menu System → Library Functions → JSON File Storage

6 Design Diagrams

Use Case Diagram

Actors: Librarian, User
Use Cases: Add Book, Add User, Borrow Book, Return Book, Search Book, Delete Book

Workflow Diagram

Start → Menu Display → User Choice → Execute Function → Save Data → Return to Menu

Sequence Diagram

User → Menu → Function → JSON File → Response → Menu

Class/Component Diagram

Components: Book, User, LibraryManager (functions handle operations)

7 Design Decisions & Rationale

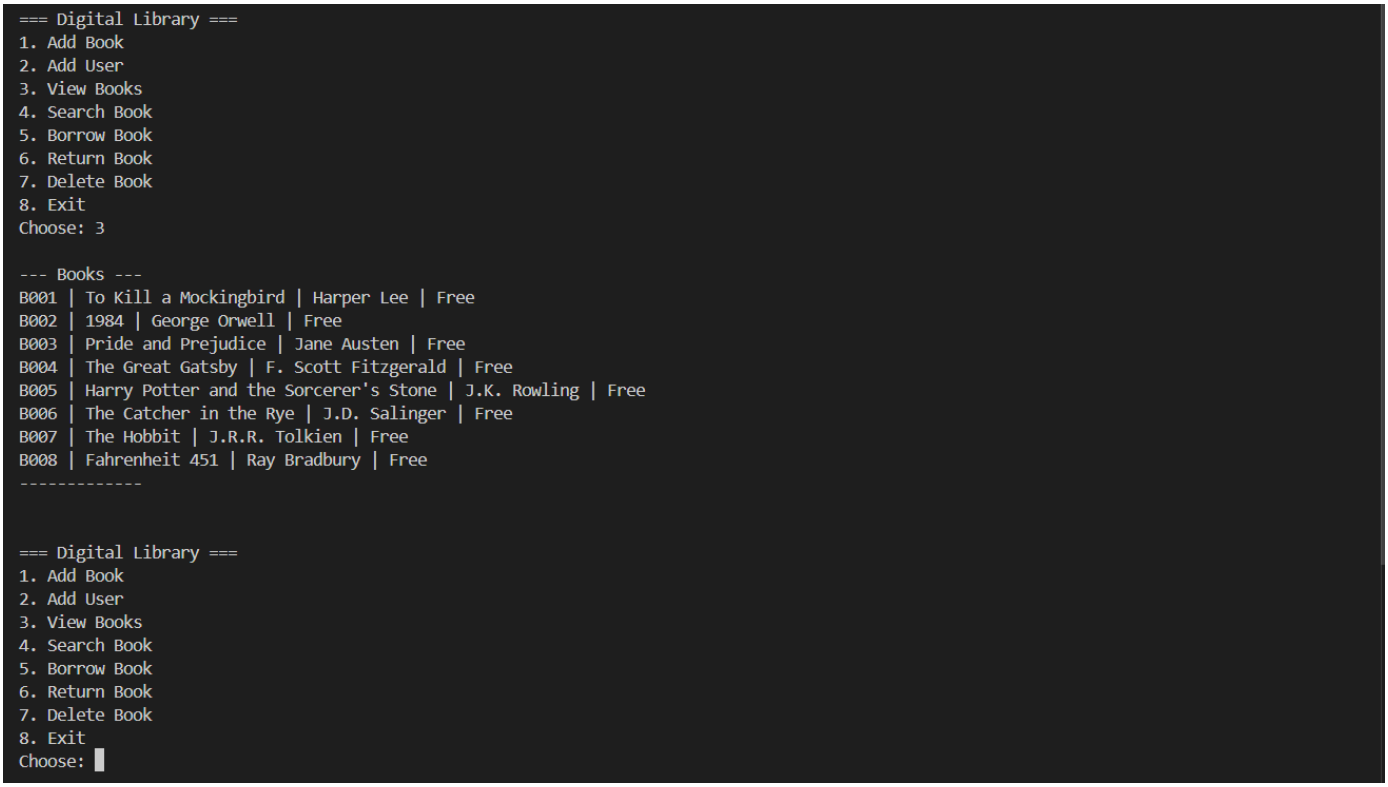
- **JSON Storage:** Chosen for simplicity and easy data manipulation.
- **Dictionary Structure:** Allows fast lookups by book ID or user ID.
- **Menu-driven CLI:** Simple, cross-platform interface without GUI complexity.
- **Function Modularity:** Each functionality (borrow, return, add) is isolated for maintainability.

8 Implementation Details

- Programming Language: Python 3.x
- Data Storage: books.json and users.json
- Modules: json, os
- Core Functions:
  - addbook()/adduser()
  - showbooks()/searchbook()
  - borrowbook()/returnbook()
  - deletebook()
  - saveall()/loadall()

9 Screenshots / Results

(Running the code)



(Selecting View Books Option)

10 Testing Approach

1. **Unit Testing:** Each function tested individually with valid and invalid inputs.
2. **Integration Testing:** Checked interactions between functions and JSON file updates.
3. **Boundary Testing:** Checked behavior for duplicate IDs, non-existent book/user, and issued book deletion.
4. **User Testing:** Verified menu choices and error handling for invalid inputs.

11 Challenges Faced

- Handling duplicate book/user IDs.
- Ensuring that borrowed books cannot be deleted.
- Persisting data correctly across program runs.
- Designing a clear, easy-to-use menu system.

12 Learnings & Key Takeaways

- Learned how to implement a file-based database using JSON.
- Understood modular programming in Python.
- Practiced handling user input, error checking, and validation.
- Learned to design a small-scale system with CRUD operations.

13 Future Enhancements

- Add GUI using Tkinter or PyQt.
- Implement due dates and overdue fines.
- Enable multiple book borrowing per user.
- Track borrowing history and generate reports.
- Migrate storage to SQL database for scalability.

14 References

1. Vityarthi's Python Essential Course– https://vityarthi.com/course/python-essentials-2