Simple Console Library Management System

SAANCH GOYAL 25BAI10805
Department of Computer Science
Engineering (AI & ML)
Vellore Institute of Technology
Bhopal, Madhya Pradesh
saanch.25bai10805@vitbhopal.ac.in

1. Introduction

This project details a basic **Library Management System (LMS)** implemented as a command-line interface (CLI) application in Python. Its core function is to track the availability of books in a small library. The system uses a simple text file, **books.txt**, to maintain a list of currently available book titles, demonstrating fundamental concepts of file persistence and basic inventory management.

2. Objective

The main objectives achieved by this project are:

- To create a **functional CLI system** for managing a list of available books.
- To demonstrate **data persistence** in Python using flat file I/O (reading from and writing to a text file).
- To implement core library operations: **adding**, **issuing** (borrowing), **returning**, and **displaying** book titles.
- To use **lists** as the primary data structure for in-memory book management.

3. Technology Used

1. **Programming Language:** Python 3.x

2. **Libraries/Modules:**

- **os module:** Used to check for the existence of the data file (books.txt) before attempting to load books, preventing errors on first run.

4. Data Structure

The available books are stored in memory as a simple **Python list of strings**, where each string represents the **title** of an available book.

File Persistence

- **File Name:** books.txt
- **Format:** Each line in the text file contains the title of **one available book**.

- **Mechanism:**
    - **Loading:** The load_books() function reads the file line by line and creates the list of strings (titles).
    - **Saving:** The save_books() function overwrites the entire file with the current list of available titles, one title per line.

5. Module Breakdown and Functionality

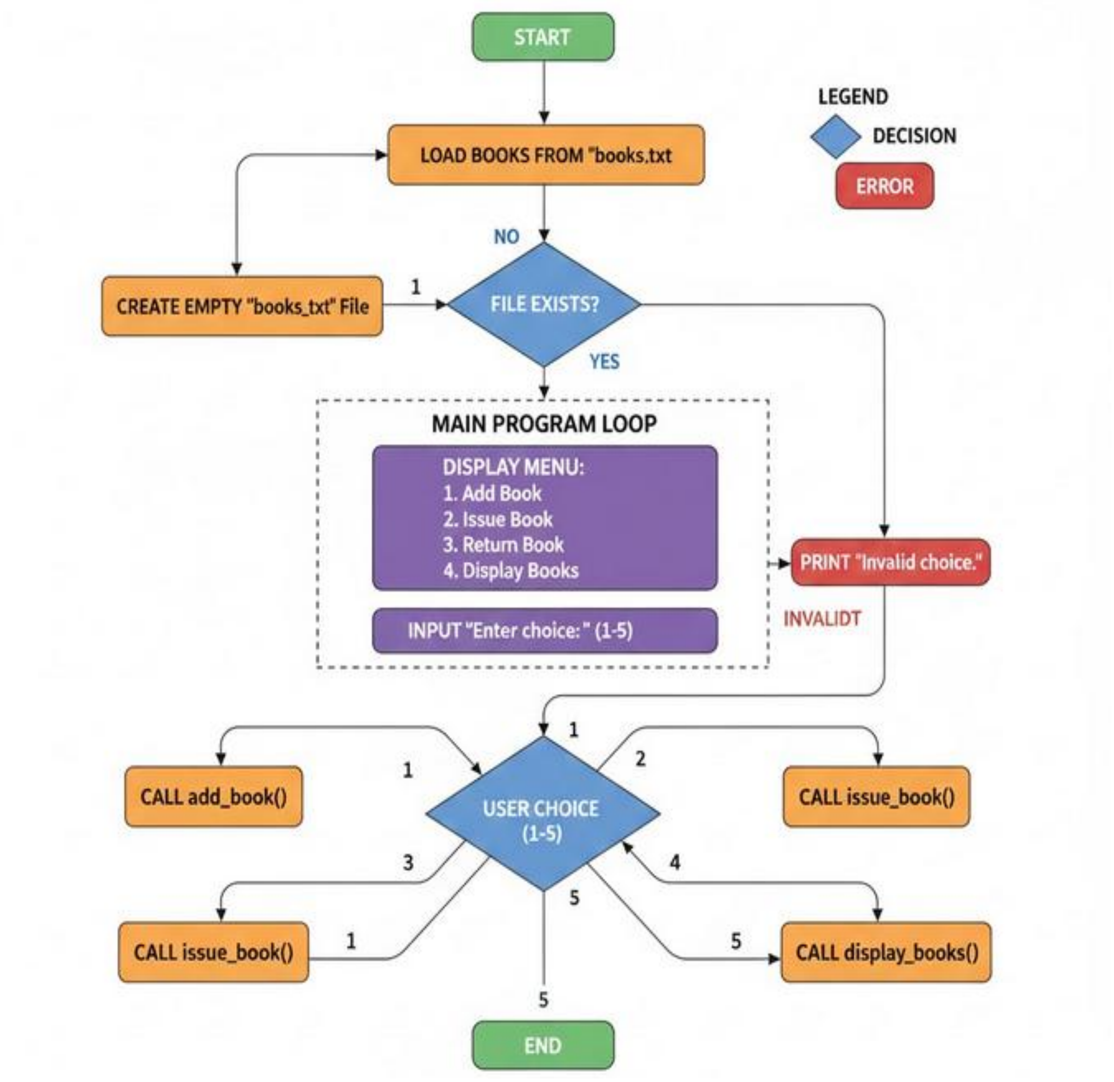
The program is organized into distinct functions, each handling a specific part of the system logic.

Function Name	**Purpose**	**Data Operation**
load_books()	Checks if books.txt exists; reads all book titles into a list.	Read
save_books(books)	Writes the current list of available book titles back to books.txt.	Write
add_book()	Prompts for a title, adds it to the list of available books, and saves the updated list.	Append, Save
issue_book()	Prompts for a title; **removes** the title from the list if found (marking it as issued/unavailable).	Search, Remove, Save
return_book()	Prompts for a title and **adds** it back to the list of available books (making it available).	Append, Save
display_books()	Prints all titles currently present in the list (available books).	Read, Display
menu()	The main loop that displays options and calls the corresponding functions based on user choice.	Control Flow

6. ALGORITHM

1. **Start.**
2. **Enter Loop:** a. Display the **Library Management System Menu** (options 1-5). b. Prompt the user to **Enter choice**. c. **Process Choice:** * If **1**: Call add_book(). * If **2**: Call issue_book(). * If **3**: Call return_book(). * If **4**: Call display_books(). * If **5**: Print "Goodbye!" and **Break Loop**. * If **Invalid**: Print "Invalid choice." d. **Continue Loop.**
3. **End.**

7. FLOWCHART



8. Core Function Logic (Pseudocode)

This function demonstrates the core logic for checking availability and updating the inventory.

```
1. FUNCTION issue_book():
2. INPUT book_title_to_issue
3. books = load_books() // Get current available
   list

4. IF book_title_to_issue IS IN books:
5. REMOVE book_title_to_issue FROM books
6. save_books(books)
7. PRINT "Book issued."
8. ELSE:
9. PRINT "Book not available." // Title was not
   found in the file
```

9. Potential Improvements and Enhancements

The current system is very basic. Key improvements for a real-world library system include:

- **Handling User Data:** Currently, the system only tracks *availability*, not *who* issued the book. An improvement would be to use a **more complex data structure** (like a dictionary of dictionaries or a CSV file) to track book attributes (Author, ISBN) and borrower information.
- **Robust Inventory:** If a user tries to issue a book that is already issued, the current system simply says "Book not available." A better system would allow for **multiple copies** and track the count of available copies.
- **Error Handling:** Implement try-except blocks for file I/O operations to handle potential permissions or corruption issues, making the application more robust.
- **Case Sensitivity:** The current search (if title in books:) is case-sensitive. It should be modified to be case-insensitive for better user experience (e.g., converting all inputs and stored titles to lowercase before comparison).

10. CONCLUSION

The Python Library Management System provides a clear, working example of file-based data persistence in Python. By using a simple text file and list operations, it successfully models the fundamental concepts of inventory tracking, making it an excellent introductory project in data management and console application development