Skill Shikshya Pvt. Ltd.
Siddhartha Margha, Kathmandu

Personal Library Management System

Assignment - 2

Kripa Hayanju

Python and Django Course
Prashant Karna
11th March 2025

Table of Contents

Introduction

A Personal Library Management System is a software application designed to help individuals efficiently organize and manage their book collections. With the increasing number of books people own, it becomes essential to have a system that allows easy tracking and retrieval of book information.

This project, developed using Python, provides an interactive interface where users can add, view, search, update, and delete books from their personal library list. It simplifies the process of managing books, making it an ideal solution for students, researchers, and book enthusiasts. The system is designed to be user-friendly, ensuring that even those with minimal technical knowledge can use it effectively.

By implementing this system, users can maintain a structured and well-organized book catalog, improving accessibility and efficiency. This report outlines the project’s objectives, features, implementation details or approach, and the output of our program.

Literature Review

The Personal Library Management System is based on fundamental concepts of digital book management and programming techniques commonly used in software development. This project leverages Python, a versatile and widely used programming language, to create an efficient and user-friendly solution for managing personal book collections.

1. **Existing Library Management Systems**

Traditional library management systems are designed for large institutions like schools, universities, and public libraries. These systems often use relational databases, cloud storage, and barcode scanning to manage vast collections. However, such systems are complex and may not be suitable for individual users who need a lightweight, simple, and efficient book management tool.

2. **Python for Personal Book Management**

Python is an ideal choice for developing a Personal Library Management System due to its:

- Ease of Use – Simple syntax makes it easy to write and understand.
- File Handling Capabilities – Built-in support for reading/writing JSON files ensures easy data storage.
- Scalability – Can be expanded to include a database or a graphical interface.

3. **CRUD Operations in Library Management**

The core functionality of the system is built around CRUD **(Create, Read, Update, Delete**) operations, which are essential in data management:

- Create: Users can add books with details like title, author, genre, and ISBN.
- Read: Users can view a list of all stored books in a structured JSON format.
- Update: Users can modify book details, ensuring accurate record-keeping.
- Delete: Users can remove books they no longer need.

4. **Data Storage with JSON**

Unlike large-scale systems that rely on SQL databases, this project uses JSON (JavaScript Object Notation) for data storage. JSON is lightweight, human-readable, and easy to manipulate in Python, making it a practical choice for a personal book management system.

Methodology

The Personal Library Management System is designed to perform key operations—Add, View, Search, Update, and Delete—on a collection of books. These operations are implemented as functions within the Library class(library.py), which interacts with the user and stores book information in a JSON file(books.json).

1. **Add a Book**

The *add_book* method allows the user to add a new book to the library with the information of different fields.

- User Input: The system prompts the user to enter details for the book, including Title, Author, Genre, and ISBN.
- Book Creation: A dictionary(book) is created with the entered data, and an ID is automatically generated based on the next available ID number. The *get_next_id* method ensures that the ID is unique and sequential.
- Storage: The new book is appended to the list(books), and the updated list is saved to a JSON file using the *save_books* method.

2. **View Books**

The *view_books* method allows the user to view all books stored in the library.

- Check for Books: The method first checks if there are any books in the books list.
- Display: The list of books is converted into a JSON string format for easy reading and printing. If no books are found, “No books available” is printed.

3. **Search for a Book based on Title**

The search_title method allows the user to search for a book by its title.

- User Input: The user is prompted to enter the title of the book they wish to search for.
- Search: The system compares the entered title to the “Title” field of each book in the books list (case-insensitive).
- Results: If matching books are found, they are added to an empty list(found_books) and printed. If no match is found, “The book is not in the list” is printed.

4. **Update a field of the book**

The update_field method allows the user to update the details of a book.

- Search for Book: Similar to the search method, the user provides a book title, and the system finds all matching books.
- Select Book: A list of matching books is displayed each with an index number, and the user selects the index number of the book they want to update.
- Choose Field to Update: The user is prompted to choose a field to update.

- Update: The new value for the selected field is entered and saved in the books list.

#Note: The ID field of the book cannot be updated!

5. **Delete a Book**

The delete_book method allows the user to delete a book from the library.

- Search for Book: The user enters a title, and the system finds all matching books.
- Select Book: A list of matching books is displayed each with an index number, and the user selects the index number of the book they wish to delete.
- Delete: The selected book is removed from the books list using del keyword.

Key Learnings

Throughout the development of the Personal Library Management System, I have gained valuable skills and knowledge. Here are some in points:

1. **max() function**

The max() function finds the maximum value in the list. It helped in automatically generating unique IDs for the books.

2. **os.path.exists() function**

The 'import os' statement gives access to the os module. The os.path.exists() function checks whether a specific file or directory exists in the given path. This is important because it ensures you don't attempt to open or read a file that might not exist, preventing errors in your program.

3. **JSONDecodeError**

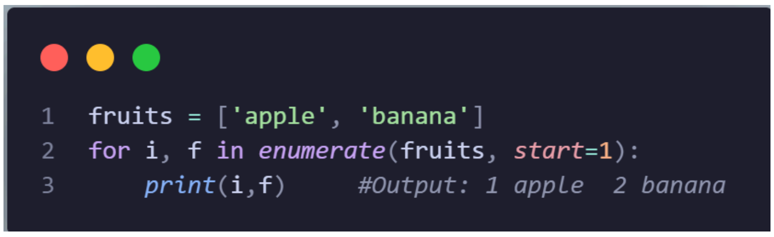
A JSONDecodeError is necessary when reading a JSON file. This error occurs when the file's content is not valid JSON, and can be resolved by using exception handling and returning an empty list if the file is corrupted or empty.

4. **.strip() method**

The .strip() method removes any leading and trailing whitespace from a string. This was useful for cleaning up user input and ensuring the data is properly formatted when searching or updating book details.

5. **enumerate() function**

The enumerate() function iterates over a list/tuple with both the index and the value of each element. This was especially useful when presenting options to the user for selecting or updating books by their index.

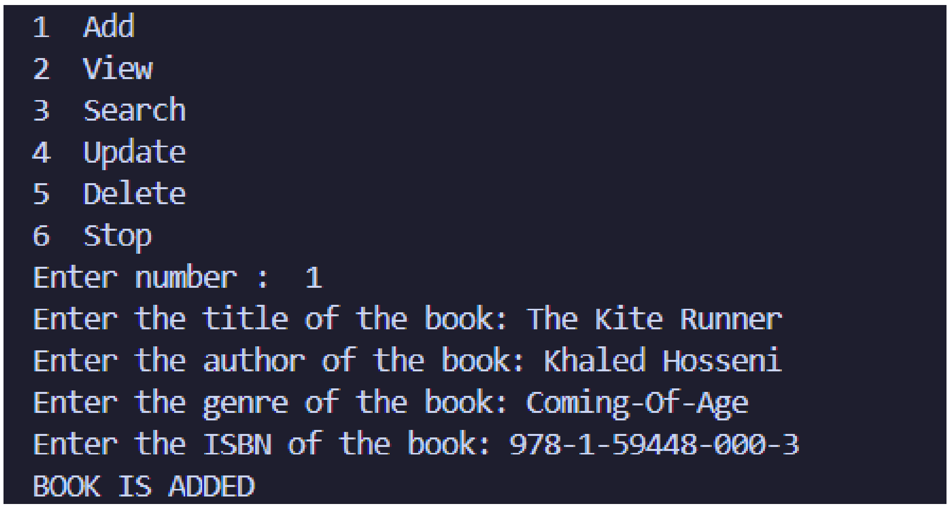


```
fruits = ['apple', 'banana']
for i, f in enumerate(fruits, start=1):
    print(i,f)      #Output: 1 apple  2 banana
```

Result

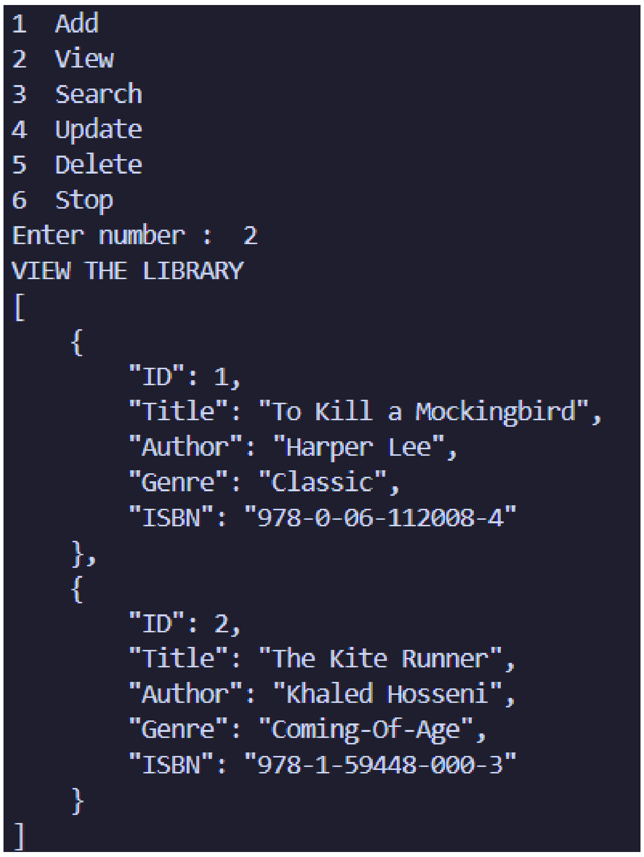
The Personal Library Management System successfully meets its objective of managing a collection of books, allowing the user to add, view, search, update, and delete books interactively. The system includes basic error handling for issues such as empty files, invalid input, and failed file operations, ensuring a smooth user experience.

1. **Add a Book**



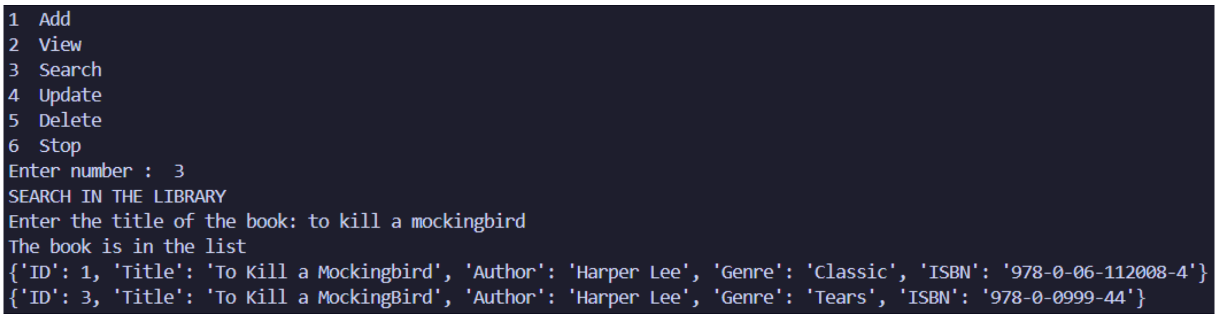
```
1  Add
2  View
3  Search
4  Update
5  Delete
6  Stop
Enter number :  1
Enter the title of the book: The Kite Runner
Enter the author of the book: Khaled Hosseni
Enter the genre of the book: Coming-Of-Age
Enter the ISBN of the book: 978-1-59448-000-3
BOOK IS ADDED
```

2. **View Books**



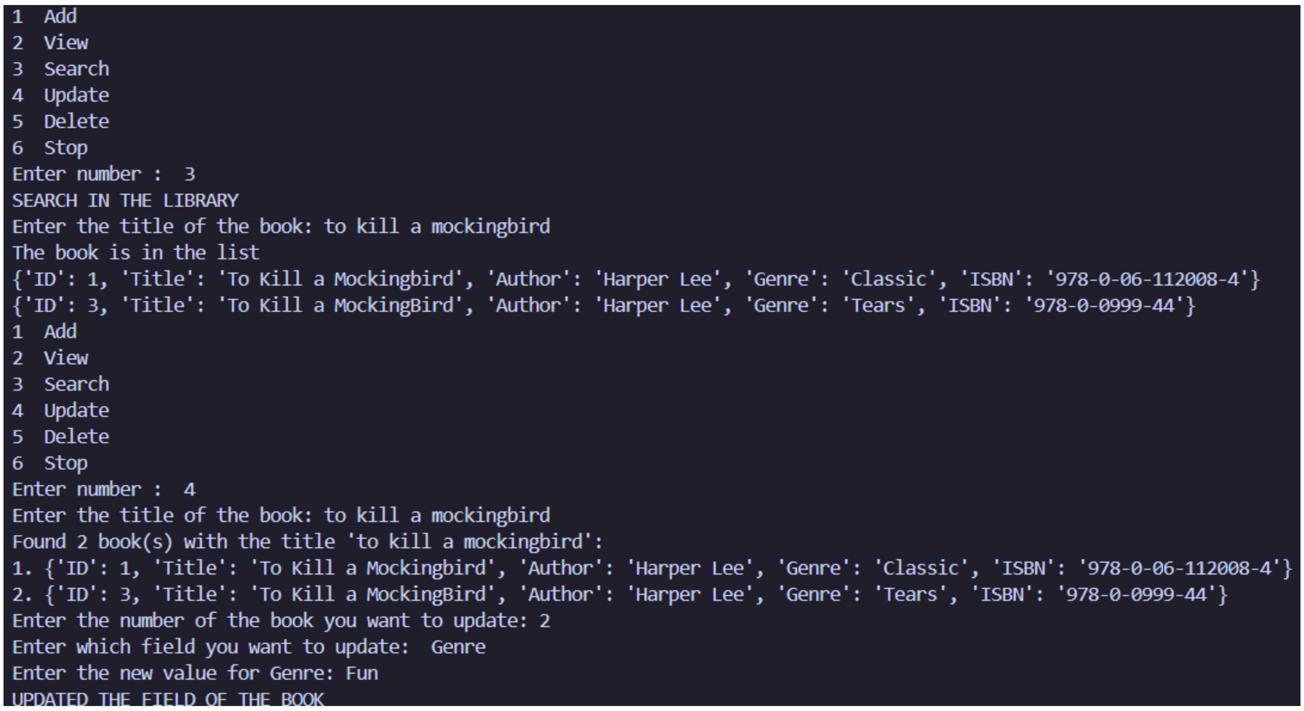
```
1  Add
2  View
3  Search
4  Update
5  Delete
6  Stop
Enter number :  2
VIEW THE LIBRARY
[
    {
        "ID": 1,
        "Title": "To Kill a Mockingbird",
        "Author": "Harper Lee",
        "Genre": "Classic",
        "ISBN": "978-0-06-112008-4"
    },
    {
        "ID": 2,
        "Title": "The Kite Runner",
        "Author": "Khaled Hosseni",
        "Genre": "Coming-Of-Age",
        "ISBN": "978-1-59448-000-3"
    }
]
```

3. Search for a Book based on Title



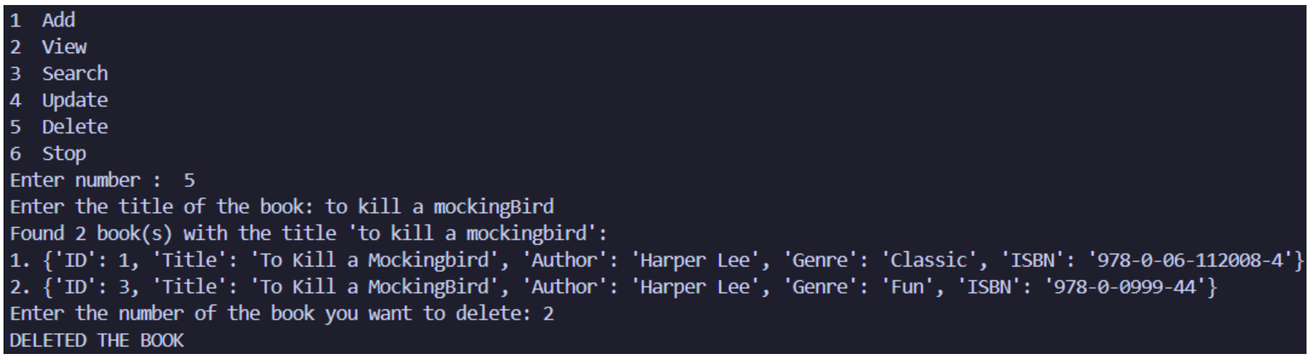
```
1  Add
2  View
3  Search
4  Update
5  Delete
6  Stop
Enter number :  3
SEARCH IN THE LIBRARY
Enter the title of the book: to kill a mockingbird
The book is in the list
{'ID': 1, 'Title': 'To Kill a Mockingbird', 'Author': 'Harper Lee', 'Genre': 'Classic', 'ISBN': '978-0-06-112008-4'}
{'ID': 3, 'Title': 'To Kill a MockingBird', 'Author': 'Harper Lee', 'Genre': 'Tears', 'ISBN': '978-0-0999-44'}
```

4. Update a field of the book



```
1  Add
2  View
3  Search
4  Update
5  Delete
6  Stop
Enter number :  3
SEARCH IN THE LIBRARY
Enter the title of the book: to kill a mockingbird
The book is in the list
{'ID': 1, 'Title': 'To Kill a Mockingbird', 'Author': 'Harper Lee', 'Genre': 'Classic', 'ISBN': '978-0-06-112008-4'}
{'ID': 3, 'Title': 'To Kill a MockingBird', 'Author': 'Harper Lee', 'Genre': 'Tears', 'ISBN': '978-0-0999-44'}
1  Add
2  View
3  Search
4  Update
5  Delete
6  Stop
Enter number :  4
Enter the title of the book: to kill a mockingbird
Found 2 book(s) with the title 'to kill a mockingbird':
1. {'ID': 1, 'Title': 'To Kill a Mockingbird', 'Author': 'Harper Lee', 'Genre': 'Classic', 'ISBN': '978-0-06-112008-4'}
2. {'ID': 3, 'Title': 'To Kill a MockingBird', 'Author': 'Harper Lee', 'Genre': 'Tears', 'ISBN': '978-0-0999-44'}
Enter the number of the book you want to update: 2
Enter which field you want to update:  Genre
Enter the new value for Genre: Fun
UPDATED THE FIELD OF THE BOOK
```

5. Delete a Book



```
1  Add
2  View
3  Search
4  Update
5  Delete
6  Stop
Enter number :  5
Enter the title of the book: to kill a mockingBird
Found 2 book(s) with the title 'to kill a mockingbird':
1. {'ID': 1, 'Title': 'To Kill a Mockingbird', 'Author': 'Harper Lee', 'Genre': 'Classic', 'ISBN': '978-0-06-112008-4'}
2. {'ID': 3, 'Title': 'To Kill a MockingBird', 'Author': 'Harper Lee', 'Genre': 'Fun', 'ISBN': '978-0-0999-44'}
Enter the number of the book you want to delete: 2
DELETED THE BOOK
```

Conclusion

The Personal Library Management System project successfully achieves its goal of creating a simple yet efficient system for managing a personal library. By implementing features like adding, viewing, searching, updating, and deleting books, the system allows for comprehensive management of book collections in a user-friendly way.

- Data Persistence: The use of JSON for storing and retrieving data ensures that the library remains intact across sessions, enhancing the overall usability of the system.
- User Interaction: The command-line interface (CLI) allows users to easily interact with the system, providing an intuitive and efficient way to manage their library.
- Error Handling: Effective error handling was implemented to ensure the system operates smoothly, even in the event of user mistakes or file-related issues.
- Learning and Skill Development: Through this project, I gained valuable knowledge of Python functions and libraries, such as enumerate(), max(), os.path.exists(), JSONDecodeError, and .strip(). These concepts significantly contributed to the robustness and functionality of the system.

While the system is functional and meets the project's objectives, there is potential for future enhancements, such as adding a graphical user interface (GUI), integrating more advanced search features (e.g., by author or genre), and supporting data export/import in other formats (e.g., CSV).

In conclusion, this project not only provided practical experience in Python programming but also resulted in a functional and expandable tool for managing a personal library.