REPORT OF MY LIBRARY MANAGEMENT PROJECT

**Description of the project:**

This project given by my professor in the CS 132 class is a library management system project in Python. It allows users to perform tasks such as adding, updating, and removing books and patrons, managing transactions (checking-in and checking-out), searching for books, displaying books and patrons, and saving data to JSON files.

**Structure of the code:**

1. **Book:** Represents a book with attributes such as title, author, ISBN, and quantity.

Example:

library = Library()

book = Book("Title", "Author", "ISBN", 5)

library.add\_book(book)

1. **Patron:** Represents a library patron with attributes like name, ID, and contact information.

Example:

library = Library()

patron = Patron("Name", "ID", "Contact Info")

library.add\_patron(patron)

1. **Transaction:** Represents a transaction involving a book being checked out or checked in by a patron.

Example (Check-Out):

library = Library()

book = library.search\_books("Title")[0] # Assuming you found the desired book

patron = library.patrons[0] # Assuming you selected a patron

transaction = Transaction(book, patron)

library.handle\_transaction(transaction)

Example (Return):

library = Library()

book = library.search\_books("Title")[0] # Assuming you found the desired book

patron = library.patrons[0] # Assuming you selected a patron

transaction = Transaction(book, patron)

transaction.return\_book()

1. **Library:** Represents the library itself, containing methods for managing books, patrons, and transactions, as well as for data loading and saving.

Example (searching for books)

library = Library()

found\_books = library.search\_books("Title")

for book in found\_books:

book.display\_details()

Example (Saving Data):

library = Library()

library.save\_data("library\_data.json")

Example (Loading Data):

library = Library()

library.load\_data("library\_data.json")

* Initialization:

- Created instances of the Library, Book, Patron, and Transaction classes to initialize the Library Management System.

* Book Class:

- Defined the `Book` class with the following attributes and methods such as:

- Attributes: `title`, `author`, `isbn`, `quantity`.

- Methods: `display\_details`, `update\_quantity`.

* Patron Class:

- Defined the `Patron` class with the following attributes and methods:

- Attributes: `name`, `id`, `contact\_info`.

- Methods: `display\_details`, `return\_book` `.

* Transaction Class:

- Defined the `Transaction` class with the following attributes and methods:

- Attributes: `book`, `patron`, `due\_date`.

- Methods: `check\_out\_book`,’check\_in\_book’.

* Library Class:

- Created the `Library` class to implement the management of the library system:

- Attributes: `books`, `patrons`, `transactions`.

- Methods: `search\_books`, `add\_book`, `remove\_book`, `add\_patron`, `remove\_patron`, `save\_data`, `load\_data`.

* Flow of the Program:

- Operations such as adding books and patrons etc., within the program flow.

* Data Persistence:

- Implemented methods to save and load data to/from a file using JSON.

* User Interface :

- Develop a simple user interface (CLI or GUI) to interact with the Library Management System.

- Allow users to perform operations like adding/removing books, managing patrons etc.

Here's a diagram showing the class structure:

+----------------------------+

| Start of Application |

+----------------------------+

|

|

v

+--------------------------------+

| Display Main Menu |

+--------------------------------+

|

+-------------+-------------+

| |

v v

+-----------------------------+ +-----------------------------+

| Perform Selected Task | | Exit Program |

+-----------------------------+ +-----------------------------+

|

+-----------+------------+

| |

v v

+----------------+ +-------------------+

| Add Book | | Remove Book |

+----------------+ +-------------------+

| |

v v

+----------------+ +-------------------+

| Add Patron | | Remove Patron |

+----------------+ +-------------------+

| |

v v

+----------------+ +-------------------+

| Search Books | | Search Patrons |

+----------------+ +-------------------+

| |

v v

+----------------+ +-------------------+

| Display Books | | Display Patrons |

+----------------+ +-------------------+

| |

v v

+----------------+ +-------------------+

| Check Out Book| | Check In Book |

+----------------+ +-------------------+

| |

v v

+----------------+ +-------------------+

| Save Data | | Exit Program |

+----------------+ +-------------------+

| |

v |

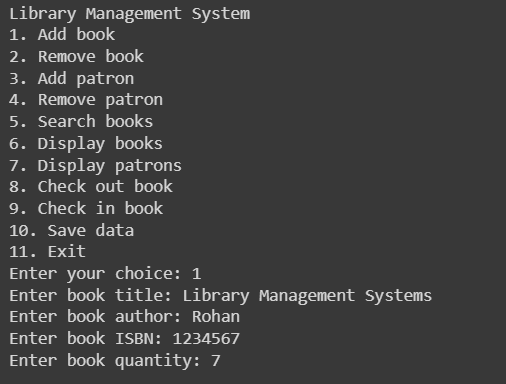
+-------------------------------+

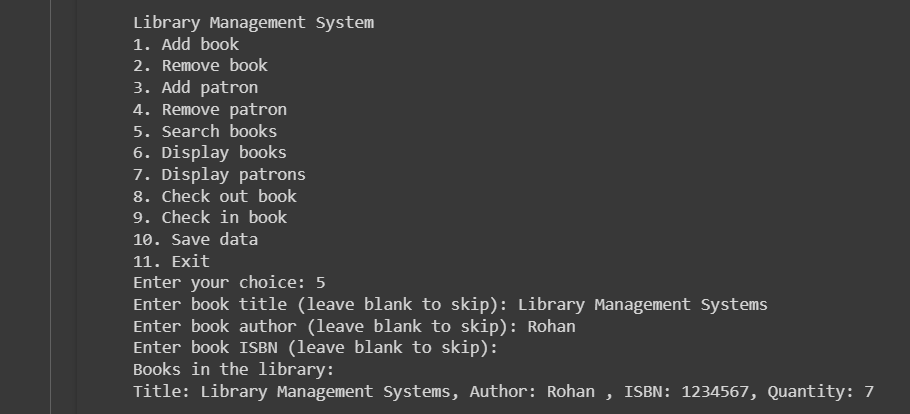
| End of Application |

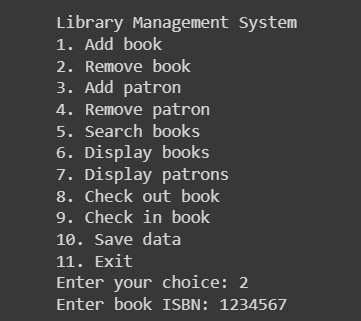
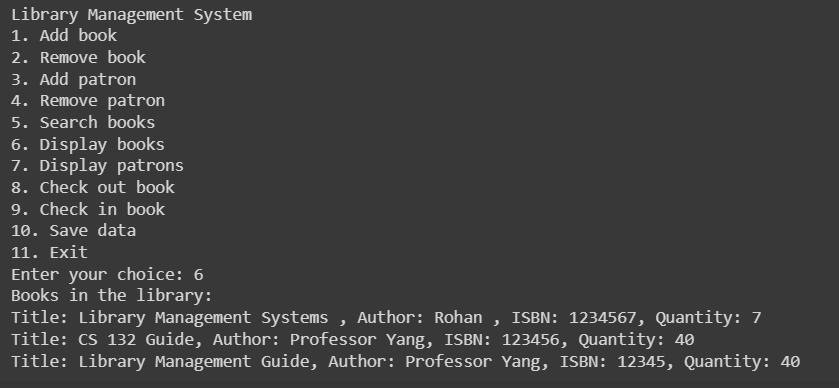
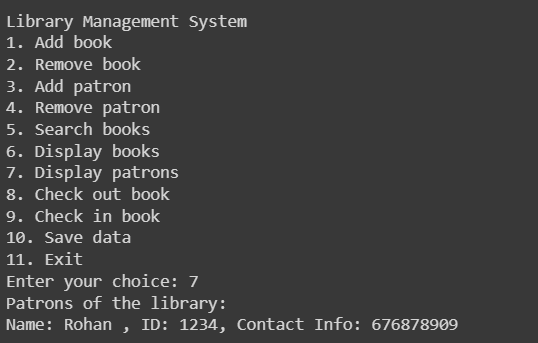
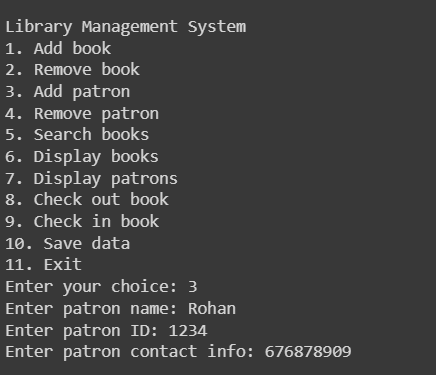
+-------------------------------+

Below are some of my sample screenshots running the code:

Adding books



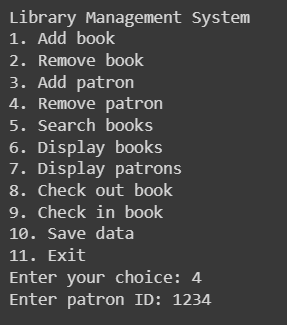
Searching for books

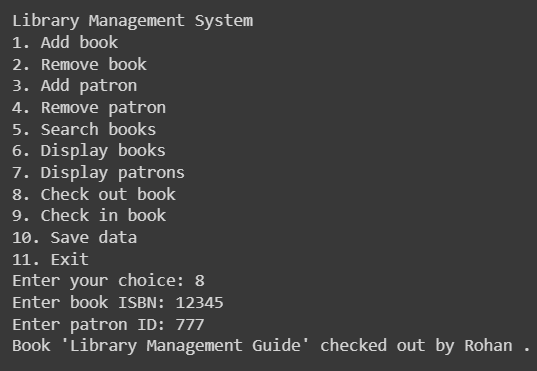
Removing Books 

1 Displaying Books

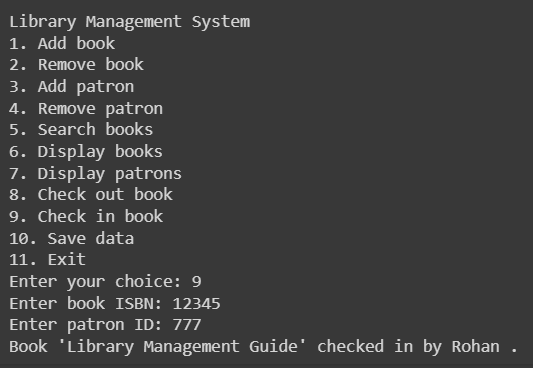
2 Adding Patrons

1 Displaying Patrons



3 Removing Patrons

2 Checking Out books



4 Checking in book

Findings through this project:

* Design and structure: Implementing a clean structure and design using classes allowed for a clean and organized structure.
* Data Persistence: Using JSON files for data persistence was effective for storing book and patron information. It ensured that data could be saved between sessions and library's operations.
* Testing : Performing tests and verifications throughout the project helped me ensure the correctness and reliability of the code and the areas which I could improve.

Challenges I faced:

* Keeping the code organized. I kept running into syntax errors while the code got longer so I had to write down the code in different files to keep it organized until I put them together.
* I had to refer to the lecture notes and my personal notes as well because there were certain implementations like the JSON files and the transaction class that took me a while to figure out.
* Ensuring the code came out with no errors took a lot of thinking and time so that I could get the most accurate output.

Areas for improvement:

* Constructing the code bit by bit is what I should’ve done from the beginning. I realized this too late and doing this would’ve saved me time and errors in the code.
* Maybe adding more features such as calculations, prices and fines would prove to be a task to further implement.
* Enhancing the user interface for better usability and user experience.

Overall, the project serves as a foundational platform for managing library operations. Although designing to facilitate the management of books, patrons, and transactions within a library was a task which pushed me to learn new things, adding more features like calculations and fines and mathematical calculations relating to a library management would make it more complete as a system.