Jason O’Neill

Professor Yang

CMPSC 132

March 3rd, 2024

Library Management System

**Description:**

The Library Management System that I have developed is a tool that is designed to organize the operations that a library’s systems use on a daily basis. This management system specifically organizes things like books, patrons or members of the library, and transactions such as checking out books. Built using Python, the system is structured for the usage of administrators and librarians. Each of these have their own logins with aspects that they are able to do. The usernames for these two roles are “Librarian” and “Administrator” their passwords being, “libpass” and “adminpass.” Once logged into the system, the administrator or librarian will be able to execute functions within the system. These consist of things like adding and removing books or patrons, generating reports, and saving books to CSV. All of the following is able to simply be done because of the simple UI interface. The idea behind this project was to make a simple and easy-to-use system that anyone working at a library would be able to use. For example, when in the menu, to add a book you simply need to choose option 1, then follow the instructions that you are given. Overall this user-friendly library management system is a program that could be utilized effectively by libraries to help make the experience of working at a library simpler.

**Code Structure**

My foundation of this system is based on four core classes, each responsible for handling different library functions. The first component is the Book class which is used to represent individual books in the library and other attributes for just the books, the second is the Patron class which is used for the patrons/staff of the library, this mainly takes care of the contact information of patrons. The Transactions class is what administers all transactions such as managing when books are checked out to patrons and returned by them, these records include transaction ID, associated book and patron, and any fines that may be applied. Finally, the Library class, this is meant to act as the central management entity that coordinates the system’s overall operations. This class maintains the lists of books, patrons, and transactions which makes it the most important class out of the four. Many features also go into this, some of these are role-based access control, an interactive menu system that offers user-friendly navigation, and data persistence that allows saving the current state of the library books to a CSV file. Upon execution, the system tells the user to log in, depending on if they log in as an admin or a librarian will give them access to levels that are based on these roles. The user can navigate the menu, selecting actions that are permitted by their roles. In the backend of this, the system manages data integrity and consistency, updating the records in response to the user’s actions and making the experience accurate of tracking library assets and activities. Overall, the creation of this Library management system uses multiple different classes, attributes that go with these classes, and functions that are able to display and make the interface of said Library Management System.

**Flow Chart:**



**Comments on the Flow Chart:**

**Program Initialization:**

The code will start by initializing the library system, and setting up necessary resources, objects, and variables.

If there's existing data in CSV format, the system will load it to restore the previous state, populating books, patrons, and transactions.

**Login Process:**

Users are prompted to log in. The code will handle authentication, checking usernames and passwords against stored credentials.

Successful login will grant access to different menus based on the user's role—either an Admin Menu or a Librarian Menu.

**Admin and Librarian Menus:**

Admins have a broad range of commands they can execute, which include administrative tasks like adding/removing librarians, generating reports, and saving data to CSV.

Librarians have a more focused command list related to day-to-day library operations, such as managing books and patrons and overseeing checkout transactions.

The code will present a list of actions as menu options. Upon selection, it will execute the corresponding function.

**Performing Actions:**

When an action is chosen, the code will carry out the task. For example, adding a book will create a new Book object and append it to the books list.

Removing a book will search the books list for the given ISBN and remove the matching entry.

**Role-Based Permissions:**

The code will reference the roles\_permissions dictionary to verify if the logged-in user has the authority to perform the chosen action.

If a user lacks permission, the code will deny access to the function and provide feedback.

**Action Completion:**

After an action is executed, the code will either loop back to the menu for further tasks or provide an option to sign out.

Signing out will likely clear any session data or cached user credentials, ensuring security.

**End of Program:**

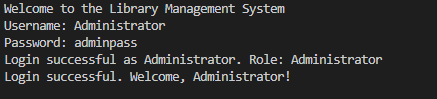
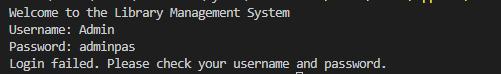
The code will have a clear exit point. If the user selects to end the program or signs out, the code will terminate the session and close the application.

**How to use:**

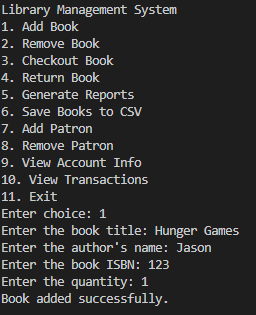
1. Run Code
2. Log in as a Librarian or Administrator by putting in login info
3. Once the menu pops up, you can choose an action to execute by putting the number that corresponds to the action.
4. If you feel like it, you can save the CSV file data for future use.
5. When back at the main menu you can either choose another action or choose to exit by putting the corresponding number.

**Verifying the Sanity of my code:**

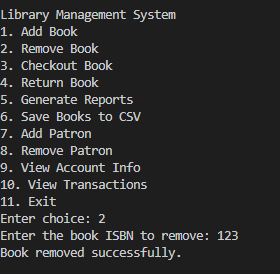
All of the following screenshots will be verifying the fact that the code is working the way it is intended.  
  
  
**Login (Correctly and Incorrectly)**



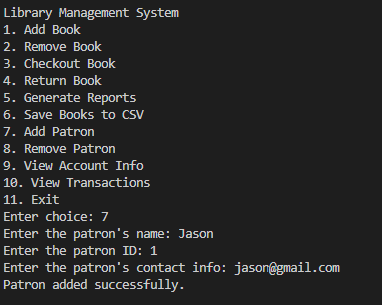
(Correct on bottom, Incorrect on top)

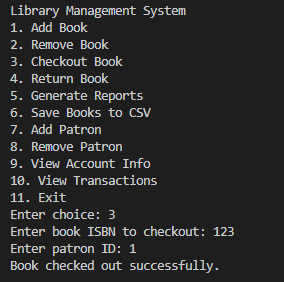
**Adding a Book  
**

**Removing a Book**

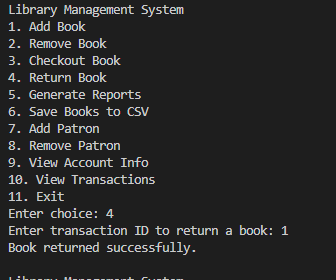
****

**Add Patron**

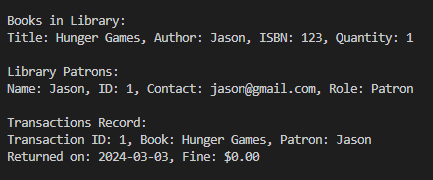
****

**Checkout Book  
**

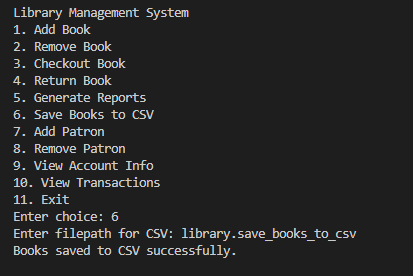
**Return Book**

****

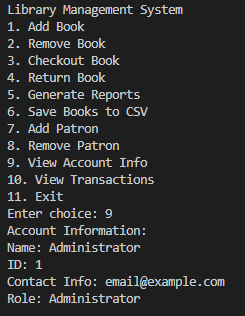
**Generate Reports**

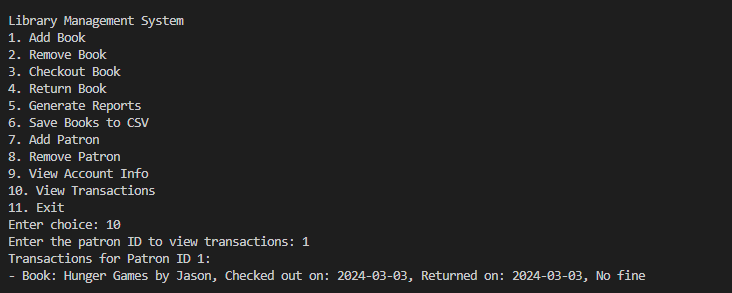
****

**Save to CSV**

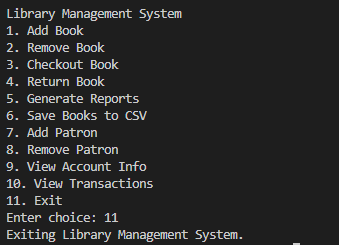
****

**View Account Info**

****

**View Transaction  
**

**EXIT**

****

**Challenges and what I learned**

During this project, I ran into plenty of challenges. When it came to coding the generating reports, this was definitely what took the longest overall time to figure out. For days I had been getting error after error with this but had eventually solved the problem and overcame what I was stressing over. Another huge problem was displaying the syntax for the due date. This took many of attempts to figure out and I had to resolve them by following YouTube guides that were able to push me in the right direction. Most of the other stuff I was already familiar with but that does not mean that there was no trouble at all. When coding the login I was not able to get it right on the first few tries but simply looked back at the old code I had and was able to fix the problem. Overall from this project I learned plenty of different aspects that I thought I had already known but it was clear that I did not. The code from this Library System has taught me plenty of things like how to perform error handling and the collaboration of classes, functions, and CSV files.

**Room For Improvement**

While this Library Management System does what it is supposed to, there is definitely things that could be improved. The user authentication system, for instance, could benefit from enhanced security measures like password encryption, and the implementation of multi-factor authentication to better protect user data. The user interface, currently command-line based, could be evolved into a graphical user interface to improve user experience and accessibility. Lastly, the system's handling of data could be upgraded from CSV files to a more robust database solution, like SQLite or PostgreSQL, to allow for more efficient data retrieval and manipulation, especially as the library's inventory grows.