LIBRARY MANAGEMENT SYSTEM

Developing library management system in python

Sergiusz Pieszak
[redacted]

Contents

1. Introduction

In today's digital world, managing libraries efficiently is essential for smooth operations and easy access to books. I was tasked to create a library management system and had to choose between using a NoSQL or SQL database. I picked SQL because of its reliability and structure.

I created a Python program with a secure login page using SHA256 encryption. This ensured that only authorized personnel could access the system. The program I created can do all the important things such as adding, finding, updating, and removing books from the library. It also allows users to sort and filter books based on what they desire.

To ensure that everything was done correctly, I followed the secure software development process. This helped me maintain the system secure and functioning smoothly. Whenever I found any problems, I used debugging to fix them. This ensured that the program was running without any issues.

1.1 Aim and Objectives

The aim of this project is to develop a library management system using Python and SQLite. The objectives include:

1. Implementing user authentication for secure access to the system.

2. Designing database schema for storing book information.

3. Creating functions for adding, updating, and deleting books from the database.

4. Providing functionality to check book availability and print inventory

5. Demonstrating debugging techniques to ensure the reliability of the application.

2. Tools, Libraries, and Modules

2.1 Libraries:

After conducting extensive research, I carefully selected the following libraries for my program:

hashlib: Utilized for password hashing to bolster security, particularly for the login functionality. Python (no date, a)

os: Used for system commands, although not mandatory. Implemented to clear the terminal screen upon selecting a new option, thereby enhancing user experience. Python (no date, c)

sqlite3: Employed for interfacing with SQLite databases, a lightweight relational database management system integrated into Python. This facilitates data storage by creating a separate .db (Database File) file for future access. Python (no date, b)

I decided to stick to the sqlite3 over MySQL since sqlite3 is a lightweight, well-documented library that is ideal for such a project.

2.2 Functions:

The program utilizes functions for several reasons:

1. For Procedural Tasks: Functions are ideal for tasks that involve a series of steps or procedures without needing to maintain state or behaviour. They offer a straightforward way to execute such tasks efficiently.

2. Modularization: Functions facilitate breaking down code into smaller, reusable components, promoting modularity and enhancing code organization. This modular approach simplifies maintenance and makes the code more understandable.

3. Handling Simple Tasks: Functions are well-suited for handling simple tasks that do not require complex data structures or behaviour. They offer a concise and direct solution for implementing such tasks.

An example of a function in my program is the main() function. “In Python, the main function serves as the starting point of execution for any software program. The program begins executing only when the main function is defined, as Python executes code directly. If the program is imported as a module, the main function will not run automatically.” Great Learning (2022)

2.3 Variables:

The program leverages variables to manage, store, and manipulate data, which can originate from user input or supplementary sources. Variables serve various purposes within the program.

2.4 Classes:

I chose to utilize a class for the library database instead of a function due to the following advantages:

1. State and Behaviour: Classes enable the encapsulation of both data and functions, promoting a cohesive program structure.

2. Object-Oriented Design: Classes align with object-oriented programming principles, facilitating inheritance, encapsulation, and polymorphism for enhanced flexibility.

3. Complexity and Abstraction: Classes offer a structured approach for managing complexity and modelling real-world entities with attributes and behaviours.

3. Software Development Lifecycle

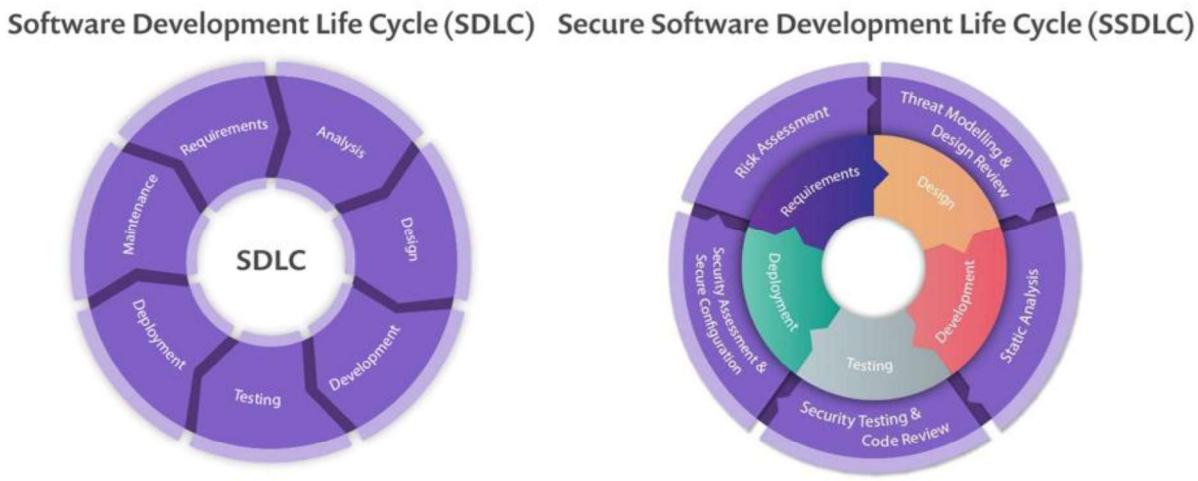

Figure 1 – Diagram comparing SDLC to SSDLC Synk (no date)

Every phase of the SDLC must prioritize the security of the overall application. Security considerations vary across each phase, underscoring the importance of maintaining security awareness throughout the entire development process. Let's explore a secure SDLC example for a team developing a membership renewal portal:

Phase 1: Requirements

During this initial phase, stakeholder input is gathered to define new features, ensuring security considerations are incorporated into functional requirements.

Phase 2: Design

Functional requirements are translated into a plan for the application, with a focus on security measures such as session token validation to prevent unauthorized access.

Phase 3: Development

Attention shifts to coding practices aligned with security guidelines, including secure coding techniques and rigorous code reviews to detect vulnerabilities. Utilizing open-source components is common, with thorough security checks using Software Composition Analysis tools.

Phase 4: Verification

Applications undergo comprehensive testing, including automated security testing, to validate adherence to design and requirements. Tools like CI/CD pipelines are employed to automate testing and deployment processes.

Phase 5: Maintenance and Evolution

Post-release, ongoing vigilance is required to address vulnerabilities discovered in production. This may involve patching code, rewrites, or responses to external penetration tests or bug bounty submissions.

Throughout the SDLC, maintaining a proactive approach to security ensures the development of a robust and resilient application.

4. Debugging

Debugging involves the task of identifying and resolving errors or bugs within software source code. When software behaves unexpectedly, programmers examine the code to understand the cause of errors. Debugging tools are utilized to execute the software under controlled conditions, scrutinize the code quickly, and address any issues encountered. Amazon Web Services (no date)

```python
def main(): #def main=
    os.system('cls||clear')
    #login need to go here.
    LoginSystem()
    #Selection for the system
    foo = int(input("Hello welcome to the USW Library!\nPleas
    if foo == 1:
        checker = str(input("You have selected ADD BOOK\nIs t
        if checker == 'n':
            main()
        if checker == 'y':
            print("Please input the following information:")
            temp1 = str(input("ISBN Number: "))
            temp2 = str(input("Book Name: "))
            temp3 = str(input("Author: "))
```

Figure 2

In *figure 2* an issue was present as I had LoginSystem() inside the main() function. Every time user wanted to continue in the library system, main() would restart and it would ask for the user to log in again. LoginSystem() had to be placed outside of the main() function and is used as a gatekeeper to prevent unauthorized access to main() function.

```
def main(): #def main=
    os.system('cls||clear') # clears the terminal. Makes it more clean
    #Selection for the system
    foo = int(input("Hello welcome to the USW Library!\nPlease select one of the following options:\n1. Add Book\n2. Update Quantity\n3. Chec

Exception has occurred: ValueError
invalid literal for int() with base 10: '4n'
  File "C:\Users\sergi\Documents\University Work\Programming\Project 1\Python Folder\SQLITE LIBRARY SYSTEM.py", line 136, in main
    foo = int(input("Hello welcome to the USW Library!\nPlease select one of the following options:\n1. Add Book\n2. Update Quantity\n3. Check Book Status\n4. Update Description\n5. Delete Book\n6. Print all inventory\n7. Save and Quit\n"))

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\sergi\Documents\University Work\Programming\Project 1\Python Folder\SQLITE LIBRARY SYSTEM.py", line 169, in main
    main()
  File "C:\Users\sergi\Documents\University Work\Programming\Project 1\Python Folder\SQLITE LIBRARY SYSTEM.py", line 157, in main
    main()
  File "C:\Users\sergi\Documents\University Work\Programming\Project 1\Python Folder\SQLITE LIBRARY SYSTEM.py", line 140, in main
    main()
  File "C:\Users\sergi\Documents\University Work\Programming\Project 1\Python Folder\SQLITE LIBRARY SYSTEM.py", line 227, in <module>
    main() # after final complete
    ^^^^^^
ValueError: invalid literal for int() with base 10: '4n'

    if foo == 1:
```

Figure 3

I encountered an issue with the program's inability to handle incorrect data types, as shown in *figure 3*, which became apparent during debugging with the Python debugger. The debugger effectively identified and explained the specific problem.

In summary, the testing and debugging phase of the application proceeded smoothly. I am pleased to affirm that the minimal number of issues encountered can be attributed to the well-structured flowchart (refer to the appendix). Integrating the software development lifecycle and the flowchart during program planning significantly minimized debugging challenges.

5. Screenshots

Screenshots of the working application, demonstrating different functionalities such as user login, book addition, and inventory display, are included below:

```
def LoginSystem(): #this function is the log-in gateway that controls access to the library system.
    valid_username = "user123" #this is the username for the system
    hashed_password = "9b8769a4a742959a2d0298c36fb70623f2dfacda8436237df08d8dfd5b37374c" #this is the hashed password - never store password in plain text

    def hash_password(password):
        return hashlib.sha256(password.encode()).hexdigest() #encodes the entered password for comparison of saved hash (see above)

    def login(): #this function manages the login process
        print("Welcome to the Login Page")
        username = input("Enter your username: ") #user input of username
        password = input("Enter your password: ") #user input of password
        if username == valid_username and hash_password(password) == hashed_password: #cleartext password gets hashed using hash_password(password)
            print("Login successful! Welcome, " + username + "!") #more personalised to increase user experience
            return 0 #the function has now completed it's task and returns 0, closing the function.
        else:
            print("Invalid username or password. Please try again.")
            LoginSystem() #resets the function, so we can have another go at crackin' the password.
    login() #loopback to prevent user bypass. Only way to exit the LoginSystem() function is to enter correct credentials
```

Figure 4

Figure 4 serves as the login system, implemented as a function for better encapsulation of data and its simplicity, thereby reducing the risk of critical bugs in the system.

```
>python3 "SQLITE LIBRARY SYSTEM.py"
Welcome to the Login Page
Enter your username: user123
Enter your password: password
Invalid username or password. Please try again.
Welcome to the Login Page
Enter your username: user123
Enter your password: pass123
```

Figure 5

Looking at *figure 5,* this represents the login process during program operation, evident by the incorrect password input. I deliberately opted not to disclose whether the username or password was incorrect to prevent granting additional knowledge to potential attackers.

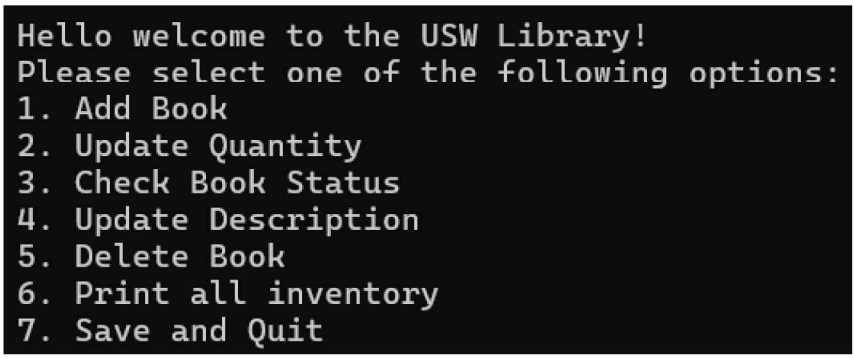

Figure 6

This dashboard (*figure 6*) is accessed within the main() function. Users navigate through this dashboard by inputting a corresponding number for each option.

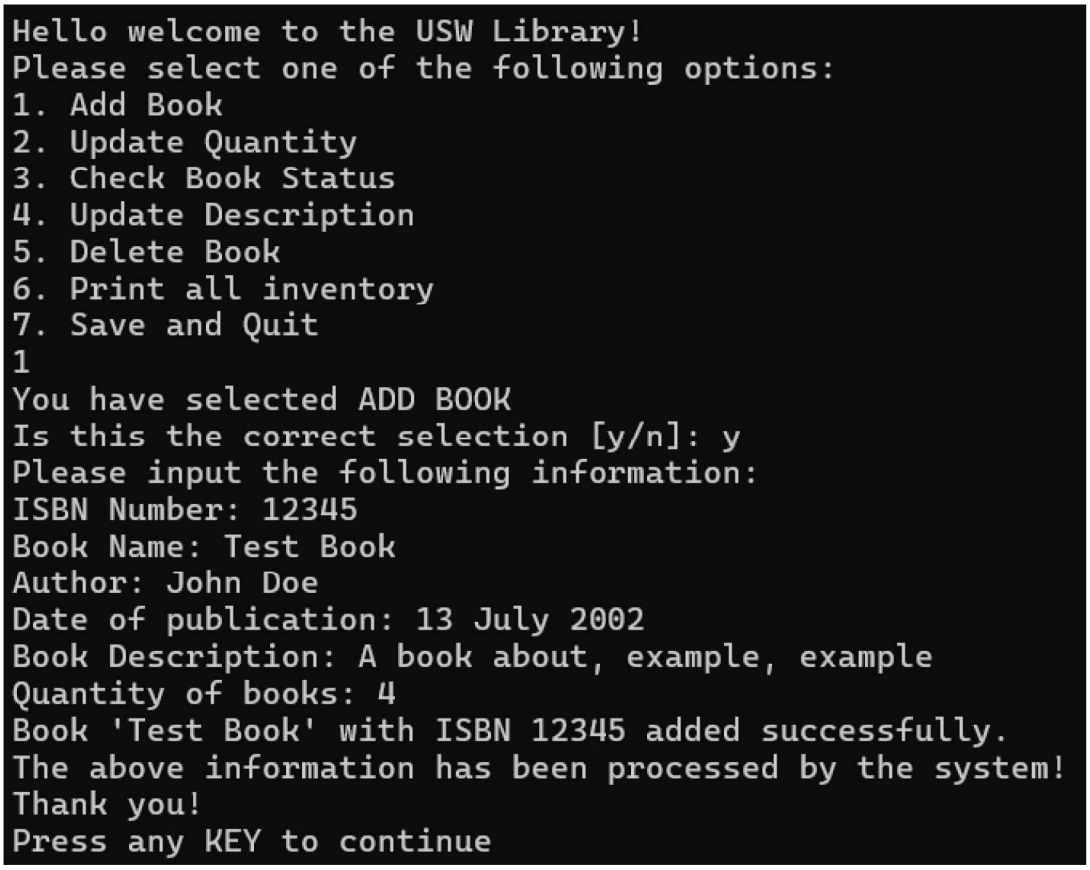

Figure 7

Figure 7 illustrates an example of the user's process in adding a book to the database.

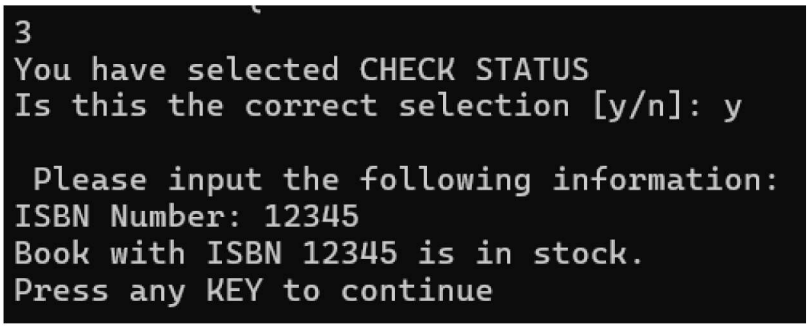

Figure 8

In another example, *figure 8* demonstrates the user checking whether a book is in stock or not.

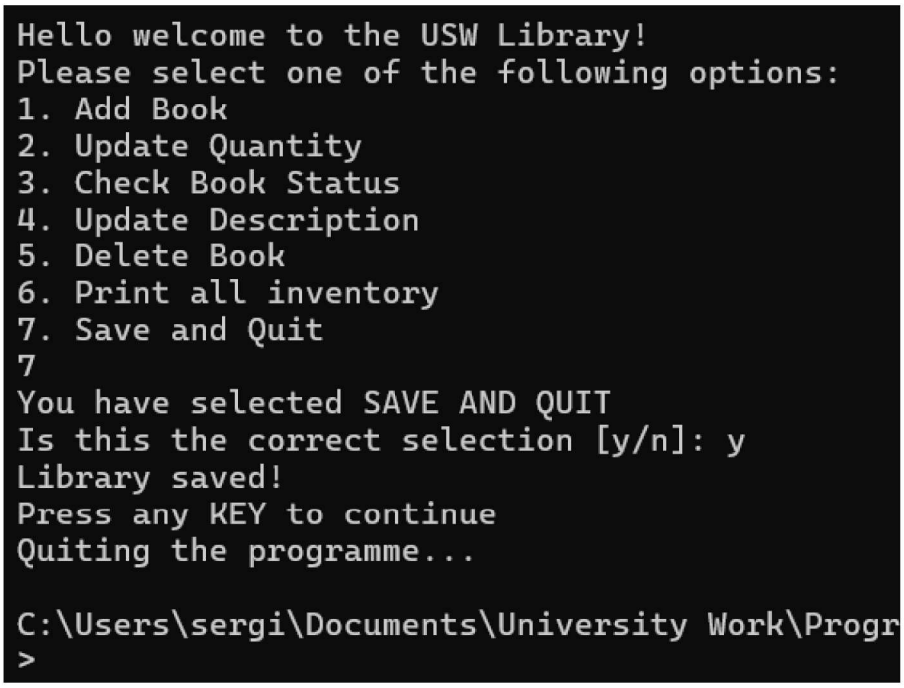

Figure 9

Figure 9 depicts the process of a user exiting the program. In this scenario, the user presses the number 7 and confirms their choice. The program performs a final check to ensure all data is saved correctly before exiting.

6. Recommendations and Lessons Learnt

Throughout the project development, several recommendations and lessons have emerged:

- Improved error handling: Enhancing error handling mechanisms would enhance user experience and application robustness.

- Integration of user feedback: Incorporating user feedback mechanisms would enable continuous improvement of the application based on user suggestions.

- Encrypting the database and decrypting using hashlib for every user request. While this would heighten security, it would also increase processing demands, thus necessitating a trade-off between resources and security.

- Implementing limits on incorrect password attempts would mitigate the risk of brute force attacks.

Conclusion

In conclusion, the development of a sophisticated library management system was an interesting project, it showed the importance of efficient library administration in today's digital era. Opting for a SQL database over NoSQL due to its reliability and structural integrity was the right decision, the program was meticulously crafted in Python, featuring a secure login mechanism employing SHA256 encryption.

Embracing the software development lifecycle ensured the system's security and smooth operation, with debugging serving as a crucial tool in addressing any encountered issues. I am glad that I took a long time at the planning stage as this tremendously limited the amount of issues with the program in the debugging and testing stage.

With a commitment to excellence in design, development, and refinement, the library management system emerges as a robust solution poised to revolutionize library operations and enhance accessibility to literary resources.

References:

Amazon Web Services (no date) *What is Debugging? Debugging Explained*. Available at: https://aws.amazon.com/what-is/debugging/ (Accessed on: 14 March 2024)

Great Learning (2022) *Python Main Function and Examples with Code*. Available at: https://www.mygreatlearning.com/blog/python-main/ (Accessed on: 13 March 2024)

Python (no date, a) *hashlib — Secure hashes and message digests* Available at: https://docs.python.org/3/library/hashlib.html (Accessed on: 13 March 2024)

Python (no date, b) *sqlite3 — DB-API 2.0 interface for SQLite databases* Available at: https://docs.python.org/3/library/sqlite3.html (Accessed on: 11 March 2024)

Python (no date, c) *os — Miscellaneous operating system interfaces* Available at: https://docs.python.org/3/library/os.html (Accessed on: 13 March 2024)

Synk (no date) *Secure Software Development Lifecycle (SSDLC)*. Available at: https://snyk.io/learn/secure-sdlc/ (Accessed on: 14 March 2024)

Appendix

Flowchart of the application

```
START

    -> [User starts program]

    -> [Display Log-In screen]

        -> [User inputs username and password]

            -> [IF incorrect password]

                -> [Request user to input username and password again]

            -> [If correct password]

    -> [Display main menu options]

        -> [User selects an option]

            -> [If option is 'Add Book']

                -> [Prompt user for confirmation]

                    -> [If confirmed]

                        -> [Prompt user for book details]

                               -> [Call 'add_book' function with input de-
tails]

                            -> [Display success message]

                            -> [Ask user to continue]

                                -> [User continues to main menu]

                            -> [User chooses to quit]

                                -> [End program]

                    -> [If not confirmed]

                        -> [Return to main menu]

            -> [If option is 'Update Quantity']

                -> [Prompt user for confirmation]

                    -> [If confirmed]

                        -> [Prompt user for ISBN and new quantity]

                            -> [Call 'update_quantity' function with input
details]

                            -> [Display success message]
```

```
                                -> [Ask user to continue]
                                    -> [User continues to main menu]
                                -> [User chooses to quit]
                                    -> [End program]
                        -> [If not confirmed]
                            -> [Return to main menu]
                -> [If option is 'Check Book Status']
                    -> [Prompt user for confirmation]
                        -> [If confirmed]
                            -> [Prompt user for ISBN]
                                -> [Call 'check_status' function with input
ISBN]
                                -> [Display book status]
                                -> [Ask user to continue]
                                    -> [User continues to main menu]
                                -> [User chooses to quit]
                                    -> [End program]
                        -> [If not confirmed]
                            -> [Return to main menu]
                -> [If option is 'Update Description']
                    -> [Prompt user for confirmation]
                        -> [If confirmed]
                            -> [Prompt user for ISBN and new book details]
                                -> [Call 'update_description' function with in-
put details]
                                -> [Display success message]
                                -> [Ask user to continue]
                                    -> [User continues to main menu]
                                -> [User chooses to quit]
                                    -> [End program]
                        -> [If not confirmed]
                            -> [Return to main menu]
```

```
            -> [If option is 'Delete Book']
                -> [Prompt user for confirmation]
                    -> [If confirmed]
                        -> [Prompt user for ISBN]
                            -> [Call 'delete_book' function with input
ISBN]
                            -> [Display success message]
                            -> [Ask user to continue]
                                -> [User continues to main menu]
                            -> [User chooses to quit]
                                -> [End program]
                    -> [If not confirmed]
                        -> [Return to main menu]
            -> [If option is 'Print All Books']
                -> [Call 'print_all_books' function]
                -> [Display all books]
                -> [Ask user to continue]
                    -> [User continues to main menu]
                -> [User chooses to quit]
                    -> [End program]
            -> [If option is 'Save and Quit']
                -> [Prompt user for confirmation]
                    -> [If confirmed]
                        -> [Close database connection]
                        -> [Display success message]
                        -> [End program]
                    -> [If not confirmed]
                        -> [Return to main menu]
    -> [End]
```