ShelfAware: A Library Management System for Efficient and Time Saving Process of Book Tracking and Library Transactions

Cabangon, Mike Justin A.
Diaz, Jannus Recz C.
Lanurias, Lance Lenard B.
Libo-on, Gian Carlo G.
Mauro, Ryan Lewis B.

Technological Institute of the Philippines
Quezon City

November 2025

Table of Contents

Introduction

A library is the brain of education. It promotes literacy and education by giving out knowledge and information for free by the means of books (Padilla, 2022). A library is the center of wisdom and knowledge of all subjects especially since it is a collection of books that a person can use to satisfy their brain with knowledge. A library should be accessible to everyone and it should be able to cater to the needs of the people since it is an important part of education.

Library Management is one of the most common problems in the education field. It is time consuming, confusing and most of the time inefficient especially when a student is in a rush but they cannot find the book that they need even when looking all around the vast collection of books in a library. Even when asking a librarian, it is extremely time consuming and takes a lot of effort in order to look for a book only to find out that it is not available in the library.

According to Iwayemi and Oyeniyi (2019), a library is a fast growing organism, it is constantly evolving and increasing which is why current methods of library management are not dynamic and up to date. In the current years, libraries, although open to the public, are not accessible due to the fact that looking for information here is time consuming and inefficient. In the same study, the researchers also stated that an efficient way of managing a library has been a necessity especially for large libraries such as public libraries and academic libraries.

In the study of Araya (2020), the transaction of a book in the institute library is prepared manually which takes a lot more time to finish a transaction. And since an institute library should be accessible to students, it is usually crowded and occupied which makes it more difficult to handle multiple transactions and is extremely inefficient in handling large collections of books. In the similar study, they also stated the need to introduce an automated system of library management in order to ensure efficient handling of the transactions and to make it more accessible for the students of the institution.

In this project, the proponents will create a C++ Program for the Library Management System This program will be tested with the use of dev C++ 5.11 version specifically and will create a database through a .txt file. This program will have both User Interface which allows for searching, borrowing, returning, and buying of books. On the other hand, the Admin Access is capable of adding, updating and removing books. Set penalties, and also search the library. They can also see the entirety of all the books in the library.

The Project

The Library Management System is a C++ console-based program that provides an easy and organized way to manage books and transactions in a library. When the program runs, it displays a simple text-based interface that presents three main options: Admin, User, and Exit. The admin section is protected by a password to ensure only authorized personnel can access it. Once logged in, the admin menu appears, displaying several functions including Add Book, Edit Book, Remove Book, Set Penalty, View Books, Search Books, Change Admin Password, and Back. Each option serves a specific purpose: Add Book lets the admin register new books by entering details such as title, category, ID, price, shelf number, and copies; Edit Book allows modification of existing records if updates are needed; Remove Book permanently deletes a book from the system; Set Penalty changes the daily overdue fine rate; View Books displays all books currently available in the library, including their availability status; Search Books helps locate a specific book by its ID, title, or category; and Change Admin Password enhances system security by allowing the admin to update the default password. On the other hand, the User interface provides access to non-admin users, where they can view available books, borrow or return them, and check penalties if they exceed the allowed borrowing days. Each transaction updates the stored data in text files, ensuring that all records are consistent and saved automatically. For summarization, the system simplifies library operations by combining functionality and ease of use, allowing both administrators and users to manage books efficiently through a straightforward and interactive terminal-based interface.

Objectives

1. A system that users can borrow and return books without any problems faced.
2. To enable administrators to add, update, and manage library records.
3. To keep track of every record of books that have gone out and the people who have them.
4. To keep track of the penalties for books not returned on time automatically.
5. The system will use files to hold and retrieve data for effective record management and maintenance.

Flowchart of the System

This section shows how the Library Management System actually works. Developers used the flowchart as a visual map, a blueprint that illustrates the steps, decisions, and checks the software performs every procedure to complete every task, from logging in, adding and removing books, to calculating a late fee. This allows anyone to clearly follow the operational logic, ensuring every part of the system works exactly as intended, whether adding a new book or returning a borrowed one. It's the plan for the software's behavior.

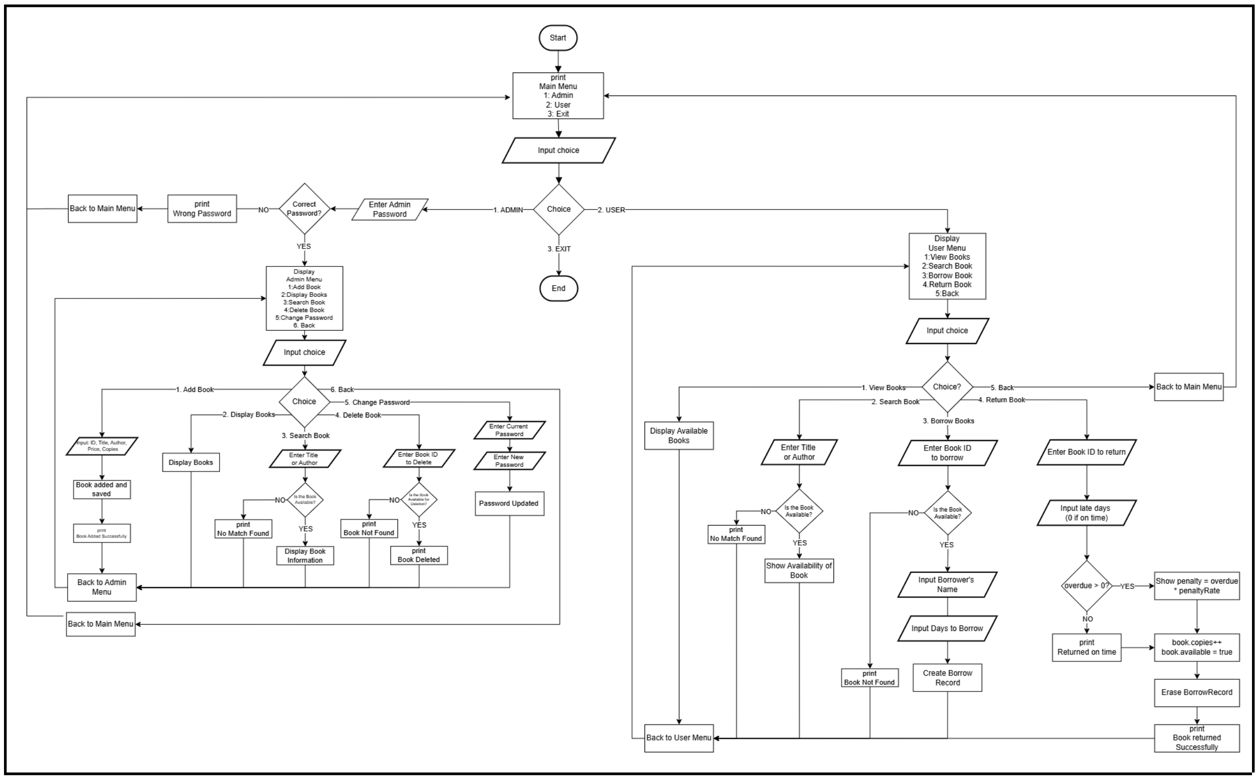


Figure 1: Library Management System Flowchart

Figure 1: Library Management System Flowchart

The Library Management System Flowchart shows the full diagram of how the program works. It starts at the beginning and shows the main choice point, directing the user toward the Admin, User, or Exit paths. This main diagram helps see the whole structure. It shows how all parts of the program, including the necessary password check for the Admin, flow back to the main menu. This figure establishes the basic way the program moves from one step to the next.

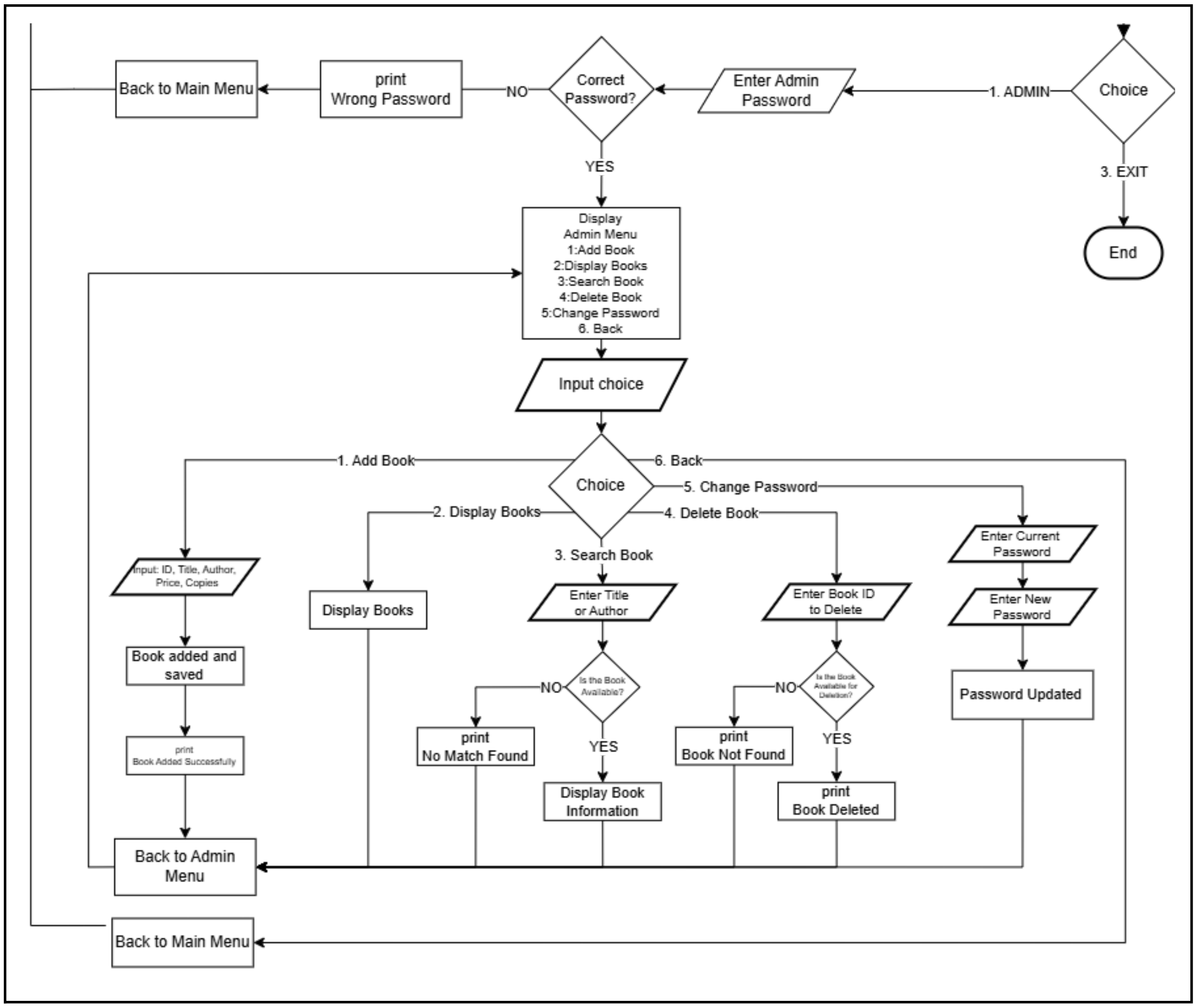


Figure 1.1: Admin Menu and Options

Figure 1.1: Admin Menu and Options

This figure focuses only on the Admin side of the program. This diagram details all the actions an admin can take after logging in. The figure shows the steps for controlling the system's data, such as Add Book and Delete Book to manage the library's stock. It also includes the flow for security, like the Change Password option. This flowchart confirms that all data changes are handled by the Admin and that the system goes back to the Admin menu after each task.

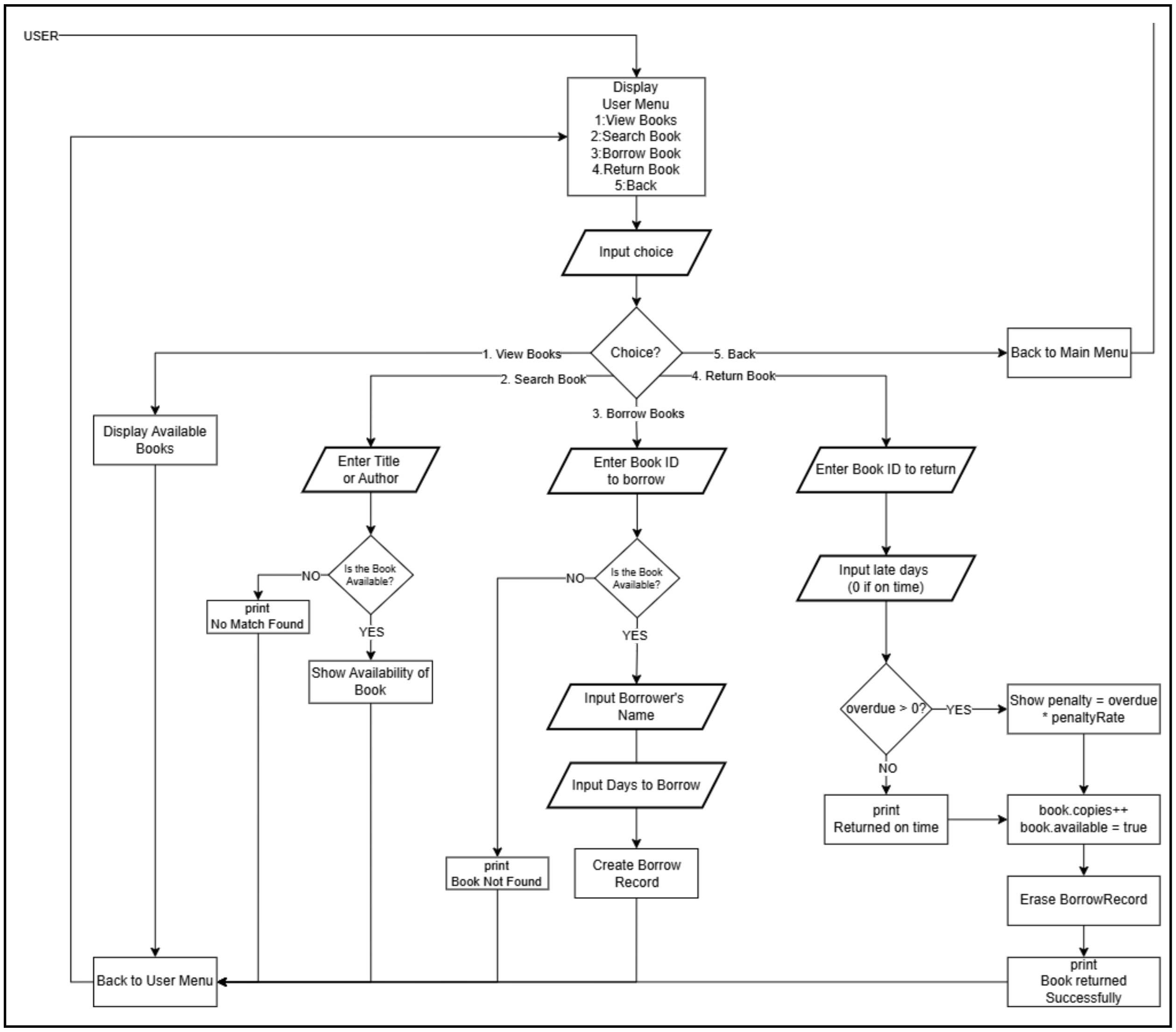


Figure 1.2: User Menu and Options

Figure 1.2: User Menu and Options

This figure shows the steps for borrowing and returning books. For borrowing, the chart shows the necessary check to make sure a copy is available. The return path involves a few steps including checking the correct borrow record, lowering the book's borrowed count, calculating any late fees based on the days entered, and saving all the changes to the system file.

Pseudocode

The Library Management System is depicted in the pseudocode as an application that simplifies the process of managing books for both the administrators and the users. Initially, the program retrieves previously saved data such as the lists of books, borrowers' records, and fines. The main menu shows three choices: Admin, User, and Exit. The admin part, secured by a password, is where new books can be added, penalty fees updated, and the total number of books along with the number of borrowed ones viewed. The system allows the reading public to see, take, and return the books. In the event of borrowing, the program takes down the details of the borrower and the number of available copies is decreased. When the book is returned, the system first tells how many days after the due date the book has been and then applies a penalty if one is warranted. The records are kept up-to-date by making all modifications instantly saved. The whole process is indeed simple and efficient as described by the pseudocode, which is a clear path for library operations and keeping proper book records.

```
BEGIN
  LOAD library data from file
  LOAD borrowed books data from file

  REPEAT
    DISPLAY "Library Management System"
    DISPLAY "1. Admin"
    DISPLAY "2. User"
    DISPLAY "3. Exit"
    INPUT choice

    IF choice = 1 THEN
      CALL AdminMenu
    ELSE IF choice = 2 THEN
      CALL UserMenu
    ELSE IF choice = 3 THEN
      SAVE all data to file
      DISPLAY "Goodbye!"
    ELSE
      DISPLAY "Invalid option"
    ENDIF
  UNTIL choice = 3
END


FUNCTION AdminMenu
  ASK for password
  IF password is correct THEN
    REPEAT
      DISPLAY "1. Add Book"
```

```
            DISPLAY "2. Set Penalty Rate"
            DISPLAY "3. View All Books"
            DISPLAY "4. View Borrowed Books"
            DISPLAY "5. Back"
            INPUT adminChoice

            SWITCH adminChoice
                CASE 1: CALL AddBook
                CASE 2: CALL SetPenalty
                CASE 3: CALL DisplayBooks
                CASE 4: CALL DisplayBorrowed
            END SWITCH
        UNTIL adminChoice = 5
    ELSE
        DISPLAY "Wrong password"
    ENDIF
END FUNCTION


FUNCTION UserMenu
    REPEAT
        DISPLAY "1. View Available Books"
        DISPLAY "2. Borrow Book"
        DISPLAY "3. Return Book"
        DISPLAY "4. Back"
        INPUT userChoice

        SWITCH userChoice
            CASE 1: CALL DisplayBooks
            CASE 2: CALL BorrowBook
            CASE 3: CALL ReturnBook
        END SWITCH
    UNTIL userChoice = 4
END FUNCTION


FUNCTION AddBook
    INPUT category, title, id, shelf, price, copies
    SET available = (copies > 0)
    ADD new book to library list
    SAVE data
    DISPLAY "Book added"
END FUNCTION
```

```
FUNCTION BorrowBook
    INPUT bookID
    FIND book with given ID
    IF found AND copies > 0 THEN
        INPUT borrower name
        INPUT allowed days
        RECORD borrow date as today
        ADD record to borrowed list
        DECREASE book copies
        UPDATE availability
        SAVE data
        DISPLAY borrow details
    ELSE
        DISPLAY "Book not found or unavailable"
    ENDIF
END FUNCTION


FUNCTION ReturnBook
    INPUT bookID
    FIND matching borrowed record
    IF found THEN
        COMPUTE days borrowed = current date - borrow date
        COMPUTE overdue = days borrowed - allowed days
        IF overdue > 0 THEN
            CALCULATE penalty = overdue * penaltyRate
            DISPLAY penalty
        ELSE
            DISPLAY "Returned on time"
        ENDIF
        INCREASE book copies
        UPDATE availability
        REMOVE borrow record
        SAVE data
    ELSE
        DISPLAY "No record found"
    ENDIF
END FUNCTION

FUNCTION DisplayBooks
    IF no books THEN
        DISPLAY "No books available"
```

```
  ELSE
    SORT books by category
    DISPLAY book details (category, title, ID, etc.)
  ENDIF
END FUNCTION

FUNCTION DisplayBorrowed
  IF no borrowed books THEN
    DISPLAY "No borrowed books"
  ELSE
    DISPLAY borrowed records (bookID, title, borrower, date, etc.)
  ENDIF
END FUNCTION

FUNCTION SetPenalty
  DISPLAY current penalty rate
  INPUT new penalty rate
  UPDATE penaltyRate
  SAVE data
  DISPLAY "Penalty updated"
END FUNCTION
```

Figure 2: Pseudo code of <system>

Data Dictionary

The table represents all the data types that the program in the project used. All these data types are included in the program and are used to define a certain characteristic of an item. The data names and data types are all specified for each description.

Table 1: Data Dictionary

Data Name	Size	Data Type	Description
1. Category	Memory (bytes) = Number of elements × Size of each element	String	Categories of the books. (fiction, non-fiction, etc..)
2. Title	Memory (bytes) = Number of elements × Size of each element	String	Name of the book/s.
3. ID	4 Bytes	Integer	ID number of a book. (Set by the admin.)
4. Shelf	Memory (bytes) = Number of elements × Size of each	String	Shelf number and location of the book. (Set by the

	element		admin.)
5. Price	8 bytes	Double	Price of the book. (Set by the admin.)
6. Copies	4 bytes	Integer	Number of copies per book. (Set by the admin.)
7. Availability	1 byte	Boolean	Indicates if a book is available.

Code

This program is a Library Management System written in C++ designed to help manage books and borrowing transactions efficiently. It provides two main roles: admin and user. The admin can add, delete, search, and display books, as well as update the system password. Users can view available books, borrow and return them, and the system automatically calculates penalties for late returns. The program uses structures (struct) to store book information and borrowed records, while vectors are used for flexible data storage. It also applies file handling through a text file named *library_data.txt* to save and load data permanently, ensuring that all records are retained even after the program closes. Additionally, a simple encryption and decryption system protects the admin password. Overall, this code demonstrates the use of modular programming, file input and output operations, loops, and conditionals to create a functional and user-friendly library management application.

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <iomanip>
#include <algorithm>
#include <cctype>
#include <cstdlib> // for system("pause")

using namespace std;

struct Book {
    string title, author, category;
    int id, copies, borrowed;
    double price;
};

struct BorrowedBook {
    int id;
    string borrower;
    int daysBorrowed;
};

const string DATA_FILE = "library_data.txt";
string adminPassword = "admin"; // default password
vector<Book> books;
vector<BorrowedBook> borrowedBooks;
const double PENALTY_RATE = 2.0; // pesos per late day


string toLowerCase(const string &s) {
    string out = s;
    transform(out.begin(), out.end(), out.begin(), ::tolower);
    return out;
}

string encrypt(const string &text) {
    string result = text;
    for (char &c : result) c += 3;
    return result;
}

string decrypt(const string &text) {
    string result = text;
    for (char &c : result) c -= 3;
    return result;
}


void saveData() {
    ofstream fout(DATA_FILE.c_str());
    if (!fout) {
        cout << "Error saving data!\n";
        return;
    }

    fout << encrypt(adminPassword) << endl;
    fout << books.size() << endl;
    for (auto &b : books)
        fout << b.id << endl
             << b.title << endl
             << b.author << endl
             << b.category << endl
             << b.price << endl
             << b.copies << endl
             << b.borrowed << endl;

    fout << borrowedBooks.size() << endl;
    for (auto &bb : borrowedBooks)
```

```
        fout << bb.id << endl << bb.borrower << endl << bb.daysBorrowed << endl;

    fout.close();
}

void loadData() {
    ifstream fin(DATA_FILE.c_str());
    if (!fin) {
        cout << "No saved data found. Creating new file...\n";
        ofstream fout(DATA_FILE.c_str());
        fout << encrypt(adminPassword) << endl; // default password
        fout << 0 << endl; // no books
        fout << 0 << endl; // no borrowed records
        fout.close();
        return;
    }

    string encPass;
    getline(fin, encPass);
    adminPassword = decrypt(encPass);

    int n;
    fin >> n; fin.ignore();
    books.clear();
    for (int i = 0; i < n; ++i) {
        Book b;
        fin >> b.id; fin.ignore();
        getline(fin, b.title);
        getline(fin, b.author);
        getline(fin, b.category);
        fin >> b.price >> b.copies >> b.borrowed;
        fin.ignore();
        books.push_back(b);
    }

    int m;
    fin >> m; fin.ignore();
    borrowedBooks.clear();
    for (int i = 0; i < m; ++i) {
        BorrowedBook bb;
        fin >> bb.id; fin.ignore();
        getline(fin, bb.borrower);
        fin >> bb.daysBorrowed; fin.ignore();
        borrowedBooks.push_back(bb);
    }

    fin.close();
}
```

```
void addBook() {
    Book b;
    cout << "\nEnter Book ID: "; cin >> b.id; cin.ignore();
    cout << "Enter Title: "; getline(cin, b.title);
    cout << "Enter Author: "; getline(cin, b.author);
    cout << "Enter Category: "; getline(cin, b.category);
    cout << "Enter Price: "; cin >> b.price;
    cout << "Enter Copies: "; cin >> b.copies;
    b.borrowed = 0;
    books.push_back(b);
    saveData();
    cout << "Book added successfully!\n";
}

void displayBooks() {
    if (books.empty()) { cout << "No books available.\n"; return; }
    cout << "\n========== BOOK LIST ==========\n";
    cout << left << setw(6) << "ID"
         << setw(25) << "Title"
         << setw(20) << "Author"
         << setw(15) << "Category"
         << setw(8) << "Copies"
         << setw(10) << "Borrowed"
         << setw(10) << "Price" << endl;
    for (auto &b : books)
        cout << left << setw(6) << b.id
             << setw(25) << b.title
             << setw(20) << b.author
             << setw(15) << b.category
             << setw(8) << b.copies
             << setw(10) << b.borrowed
             << fixed << setprecision(2) << b.price << endl;
}

void searchBook() {
    string key;
    cout << "\nEnter title or author: ";
    cin.ignore();
    getline(cin, key);
    string lowKey = toLowerCase(key);

    bool found = false;
    for (auto &b : books) {
        if (toLowerCase(b.title).find(lowKey) != string::npos ||
            toLowerCase(b.author).find(lowKey) != string::npos) {
            cout << "ID: " << b.id << " | " << b.title << " by " << b.author
                 << " (" << b.copies - b.borrowed << " available)\n";
            found = true;
        }
    }
    if (!found) cout << "No match found.\n";
}

void deleteBook() {
    int id; cout << "\nEnter Book ID to delete: "; cin >> id;
    for (size_t i = 0; i < books.size(); ++i) {
        if (books[i].id == id) {
            books.erase(books.begin() + i);
            saveData();
            cout << "Book deleted.\n";
            return;
        }
    }
    cout << "Book not found.\n";
}

void changeAdminPassword() {
    string oldPass, newPass;
    cout << "\nEnter current password: "; cin >> oldPass;
    if (oldPass != adminPassword) { cout << "Wrong password!\n"; return; }
    cout << "Enter new password: "; cin >> newPass;
    adminPassword = newPass;
    saveData();
    cout << "Password updated successfully!\n";
}


void borrowBook() {
    string name; int id, days;
    cout << "\nEnter your name: "; cin.ignore(); getline(cin, name);
    cout << "Enter Book ID to borrow: "; cin >> id;
    cout << "Enter days to borrow: "; cin >> days;

    for (auto &b : books) {
        if (b.id == id) {
            if (b.copies - b.borrowed > 0) {
                b.borrowed++;
                borrowedBooks.push_back({id, name, days});
                saveData();
                cout << "Book borrowed successfully!\n";
                return;
            } else {
                cout << "No available copies.\n";
                return;
            }
        }
    }
    cout << "Book not found.\n";
}
```

```
void returnBook() {
   string name; int id, lateDays;
   cout << "\nEnter your name: "; cin.ignore(); getline(cin, name);
   cout << "Enter Book ID to return: "; cin >> id;
   cout << "Enter late days (0 if on time): "; cin >> lateDays;

   for (size_t i = 0; i < borrowedBooks.size(); ++i) {
      if (borrowedBooks[i].id == id && borrowedBooks[i].borrower == name) {
         borrowedBooks.erase(borrowedBooks.begin() + i);
         for (auto &b : books)
            if (b.id == id) b.borrowed--;
         saveData();
         if (lateDays > 0)
            cout << "Penalty: ?" << (lateDays * PENALTY_RATE) << endl;
         else
            cout << "Returned on time. Thank you!\n";
         return;
      }
   }
   cout << "Record not found.\n";
}


void adminMenu() {
   while (true) {
      cout << "\n===== ADMIN MENU =====\n";
      cout << "1. Add Book\n2. Display Books\n3. Search Book\n4. Delete Book\n5. Change Password\n6.
Back\nChoice: ";
      int choice; cin >> choice;
      switch (choice) {
         case 1: addBook(); break;
         case 2: displayBooks(); break;
         case 3: searchBook(); break;
         case 4: deleteBook(); break;
         case 5: changeAdminPassword(); break;
         case 6: return;
         default: cout << "Invalid choice.\n";
      }
   }
}

void userMenu() {
   while (true) {
      cout << "\n===== USER MENU =====\n";
      cout << "1. View Books\n2. Search Book\n3. Borrow Book\n4. Return Book\n5. Back\nChoice: ";
      int choice; cin >> choice;
      switch (choice) {
         case 1: displayBooks(); break;
         case 2: searchBook(); break;
         case 3: borrowBook(); break;
```

```
            case 4: returnBook(); break;
            case 5: return;
            default: cout << "Invalid choice.\n";
        }
    }
}


int main() {
    loadData();

    while (true) {
        cout << "\n===== LIBRARY SYSTEM =====\n";
        cout << "1. Admin\n2. User\n3. Exit\nChoice: ";
        int choice; cin >> choice;

        if (choice == 1) {
            string pass; cout << "Enter Admin Password: "; cin >> pass;
            if (pass == adminPassword) adminMenu();
            else cout << "Wrong password.\n";
        }
        else if (choice == 2) userMenu();
        else if (choice == 3) { saveData(); cout << "Data saved. Goodbye!\n"; break; }
        else cout << "Invalid choice.\n";
    }

    system("pause");
    return 0;
}
```

Figure 3. Program Code

Results and Discussion

The results presented in this section demonstrate the functional performance of the Library Management System program. The system was designed to handle both administrative and user-level operations such as adding, searching, borrowing, and returning books. During program execution, the main interface allows the user to select between the Admin and User modes. Each operation performed within the system updates the stored data file, ensuring that all changes made by the administrator or users are saved for future sessions. The implementation highlights the use of file handling for persistent data storage, allowing the program to maintain records of books, borrowed transactions, and user information even after termination. It also incorporates input validation, password protection, and a simple text-based user interface for ease of navigation. The results obtained from running various features of the system are presented and discussed in the following section to show how each function behaves and how effectively it manages typical library operations such as book inventory control, data retrieval, and fine computation for late returns.

Upon running the program, the Library Management System displays a main menu that allows users to select between Admin, User, and Exit options. When the administrator logs in using the correct password, access is granted to a separate admin interface where core management tasks can be performed. During testing, the administrator successfully added multiple books by entering details such as book ID, title, author, category, price, and number of copies. The system immediately reflected these entries in the stored file, confirming that the file handling mechanism worked as intended. The displayed book list appeared in a clear tabular format showing each book's information along with the number of copies and borrowed count.

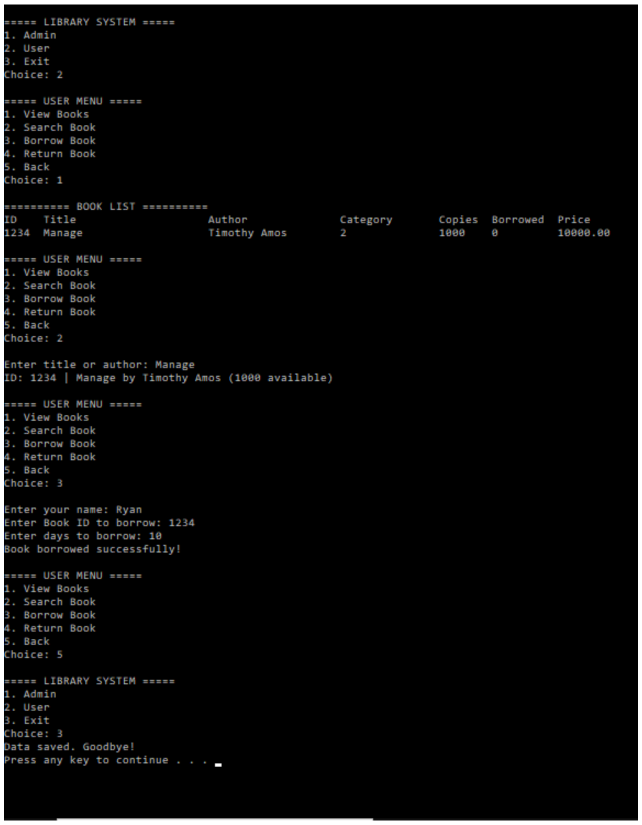
The search function was tested by entering either a book title or an author name. The program correctly retrieved the corresponding book record regardless of case sensitivity, demonstrating effective use of string manipulation and comparison functions. Deleting a book was also verified to work properly; when a valid book ID was entered, the record was successfully removed from both memory and the data file. Additionally, changing the admin password function was tested and confirmed to ensure that the new password was encrypted and stored securely, thereby enhancing the program's basic security feature.

For the user interface, several operations were performed to evaluate its reliability. Users could view the available books and their details, providing a clear overview of what was currently in the library's inventory. The borrowing feature allowed a user to borrow a book as long as copies were available. Once borrowed, the system increased the borrowed count and saved the record under the borrower's name. Returning a book also worked effectively; the program located the record and decreased the borrowed count upon return. When the user entered late days, the system correctly computed and displayed the penalty amount at a rate of ₱2.00 per day.

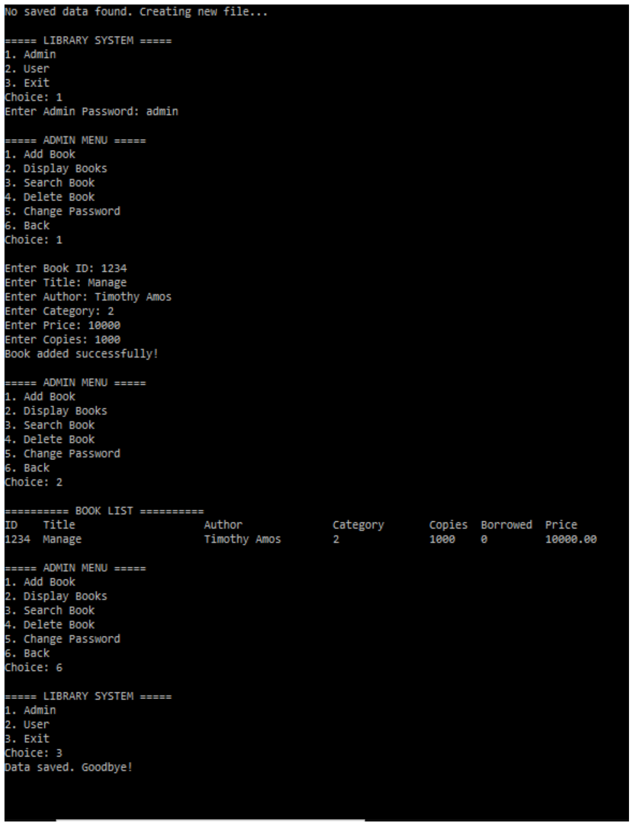
Overall, the results show that the program efficiently handled both administrative and user operations. It maintained accurate book records, applied penalties correctly, and ensured data persistence through file storage. The system provided a simple but effective console-based interface suitable for small-scale library operations. Although it lacks graphical elements, its structure and logic demonstrate solid programming fundamentals,particularly in file handling, conditional control, data management, and user interaction. The program's results confirm that it can serve as a foundational model for a more advanced library management system with future enhancements

such as automated due date tracking, improved search filters, and integration with a database or graphical

user interface.



```
===== LIBRARY SYSTEM =====
1. Admin
2. User
3. Exit
Choice: 2

===== USER MENU =====
1. View Books
2. Search Book
3. Borrow Book
4. Return Book
5. Back
Choice: 1

========== BOOK LIST ==========
ID    Title                    Author              Category       Copies  Borrowed  Price
1234  Manage                   Timothy Amos        2              1000    0         10000.00

===== USER MENU =====
1. View Books
2. Search Book
3. Borrow Book
4. Return Book
5. Back
Choice: 2

Enter title or author: Manage
ID: 1234 | Manage by Timothy Amos (1000 available)

===== USER MENU =====
1. View Books
2. Search Book
3. Borrow Book
4. Return Book
5. Back
Choice: 3

Enter your name: Ryan
Enter Book ID to borrow: 1234
Enter days to borrow: 10
Book borrowed successfully!

===== USER MENU =====
1. View Books
2. Search Book
3. Borrow Book
4. Return Book
5. Back
Choice: 5

===== LIBRARY SYSTEM =====
1. Admin
2. User
3. Exit
Choice: 3
Data saved. Goodbye!
Press any key to continue . . .
```



```
No saved data found. Creating new file...

===== LIBRARY SYSTEM =====
1. Admin
2. User
3. Exit
Choice: 1
Enter Admin Password: admin

===== ADMIN MENU =====
1. Add Book
2. Display Books
3. Search Book
4. Delete Book
5. Change Password
6. Back
Choice: 1

Enter Book ID: 1234
Enter Title: Manage
Enter Author: Timothy Amos
Enter Category: 2
Enter Price: 10000
Enter Copies: 1000
Book added successfully!

===== ADMIN MENU =====
1. Add Book
2. Display Books
3. Search Book
4. Delete Book
5. Change Password
6. Back
Choice: 2

========== BOOK LIST ==========
ID    Title                    Author              Category       Copies  Borrowed  Price
1234  Manage                   Timothy Amos        2              1000    0         10000.00

===== ADMIN MENU =====
1. Add Book
2. Display Books
3. Search Book
4. Delete Book
5. Change Password
6. Back
Choice: 6

===== LIBRARY SYSTEM =====
1. Admin
2. User
3. Exit
Choice: 3
Data saved. Goodbye!
```

Conclusion

ShelfAware, the Library Management System, was developed to support the sharing of knowledge in an educational environment where books can be easily located and managed. When faced with shelves full of titles that can be overwhelming to search through, ShelfAware serves as an efficient solution by performing its intended functions such as adding, searching, borrowing, returning, and deleting books while maintaining accurate and up-to-date records through file handling. Furthermore, the system's input validation, password protection, and structured text-based interface ensure both security and ease of use. Overall, it demonstrates solid programming fundamentals and serves as a reliable prototype for library operations

References

Adhavan, S., Surya, A., Sujithkumar, K., Vinotheni, M. S., & Lokeshwaran, R. (2024). *A comprehensive library management system for efficient book handling and user interaction*. *International Journal of Novel Research and Development, 9*(11). https://www.ijnrd.org/papers/IJNRD2411227.pdf

Araya, N. T. W. (2020). Designing web-based library management system. *International Journal of Engineering Research And*, *V9*(10). https://doi.org/10.17577/ijertv9is100131

Barua, N. (2018). ISSUES AND CHALLENGES IN LIBRARY SYSTEM MIGRATION: A CASE STUDY IN INDIA. *INTERNATIONAL JOURNAL OF LIBRARY AND INFORMATION SCIENCE*, *7*(2). https://doi.org/10.34218/ijlis.7.2.2018.007

Iwayemi, A., & Oyeniyi, S. (2019). Development of a robust library management system. *International Journal of Computer Applications*, *178*(12), 9–16. https://doi.org/10.5120/ijca2019918850

Padilla, R. C. (2022). Assessment of Library users' problems on Transactional Procedures: Basis for Library Management System Development. *International Journal of Scientific and Management Research*, *05*(06), 10–17. https://doi.org/10.37502/ijsmr.2022.5602

Radha Krishna, A., Sireesha, K., Sravanthi, V. N. T., Keerthana, P., Satwik, K., & Ram Lakshmi, M. C. (2022). *Library management systems – A survey*. International Journal of Research Publication and Reviews, 3(2). Library Management Systems – A Survey

Singh, H., Rana, P., Kumar, G., Thakur, S., Singh, H. K., & Tiwari, S. (2024). *Revolutionizing libraries: A review of advanced library management systems*. *International Journal of Research and Analytical Reviews, 11*(2). https://www.researchgate.net/publication/387079186_Revolutionizing_Libraries_A_Review_of_Advanced_Library_Management_Systems?brid=J8F678bHRki9jLhhc2o-ow