LIBRARY MANAGEMENT SYSTEM

Session: 2022 – 2026

Submitted by:

AbdulRehman 2022-CS-57

Supervised by:

Madam Maida Shahid

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Table of Contents

LIBRARY MANAGEMENT SYSTEM 1
PROJECT DESCRIPTION 3
USERS OF APPLICATION 3
FUNCTIONAL REQUIREMENTS 3
WIRE FRAMES OF APPLICATION 5
    SIGN MENU 5
        ADMIN MENU 5
        CUSTOMER MENU 8
DATA STRUCTURES (PARALLEL ARRAYS): 12
FUNCTION PROTOTYPES : 12
COMPLETE CODE : 13
WEAKNESSES IN APPLICATION : 36
FUTURE DIRECTION : 36
RUBRICS : 38

PROJECT DESCRIPTION

- This project will help manage the day to day operations of a library. The main objective of this system is to provide an efficient program for organizing, storing and retrieving the information about the library. This project will automate many tasks which will save a lot time.

USERS OF APPLICATION

There are two types of users of this management system application which are as follows:

- Library Staff : This includes the librarians who manage the day to day tasks of the library.
- Clients : This includes the students and the members who will use the system to request for material.

FUNCTIONAL REQUIREMENTS

User Story ID	***As a***	***I want to perform***	***So that I can***
1	Admin & Client	View the books.	View all the present books in the library
2	Admin & Client	Search a book	Search a specific book in the catalog.
3	Admin	Add or Remove a book	Add a new book in the library or delete a book.
4	Admin	Add or Remove an account	Add a new user in the system or remove a user from the system.
5	Admin	Budget Management	

6	Admin	System Management	Show the budget and how it is used. Changing and managing the rules of library.
7	Admin	Backup	Create a backup of all the data present in the system.
8	Admin & Client	Logout	Log out of their account.
9	Client	Request a book	Create a request for a book. View and check their requested books and item.
10	Client	Change password	Change password for the account.
11	Client	Check deadline	Check a deadline for a specific book.
12	Client	Extend deadline	Extend deadline for a book.
13	Client	Notifications	Check the notifications for a requested book.
14	Client	Check fine	Check if the user has any payable fine.

S

WIRE FRAMES OF APPLICATION

SIGN MENU

```
Enter username : AbdulRehman

Enter password : ********
```

ADMIN MENU

```
   1. View the available books.
   2. Search a book.
   3. Add a book.
   4. Remove a book.
   5. Add an account.
   6. Remove an account.
   7. Change Password.
   8. Show all users.
   9. Backup data.
   10.Logout.
```

ADMIN OPTION-1

```
Name                  Author
Little Women          Alcot, L.M
The Last Steam Train  Noonan, D
The Bedtime Book      Hernerson, K
```

ADMIN OPTION-2

```
Enter the name of the book or the author : The Bedtime Book
```

ADMIN OPTION-3

```
Enter the name of the book : The Last Steam Train
Enter the name of the author : Noonan, D
```

ADMIN OPTION-4

```
Enter the name of the book : The Last Steam Train
Book deleted successfully.
```

ADMIN OPTION-5

```
Enter the username : Alyan_z
Enter the password  : *******
Enter the role : client
```

ADMIN OPTION-6

```
Enter the username : Alyan_z

User deleted successfully.
```

ADMIN OPTION-7

```
Enter the old password : *******

Enter the new password : *******
```

ADMIN OPTION-8

```
The users of this system are :

AbdulRehman

Musa

Faizan
```

ADMIN OPTION-9

```
Backup in progress

Backup completed successfully.
```

ADMIN OPTION-10

```
Logging out…
```

CUSTOMER MENU

```
  View the Books
  Search a book
  Borrow a book
  Request a book
  Show borrowed book
  Change password
  Checking fines
  Notifications
  Get help
  Logout
```

CUSTOMER OPTION-1

```
No.  Name                 Author
1    Little Women         Alcot, L.M
2    The Last Steam Train  Noonan, D
3    The Bedtime Book     Hernerson, K
```

CUSTOMER OPTION-2

```
Enter the name of the book or the author : Perfect Run

Book not found !!
```

CUSTOMER OPTION-3

```
Enter the name of the book : The Last Steam Train

The books has been added to your list.
```

CUSTOMER OPTION-4

```
Enter the name of the book : Perfect Run

Request has been accepted. You will be notified when the
book has arrived.
```

CUSTOMER OPTION-5

```
The borrowed books are :

Prince of Thorns

King of Thorns
```

CUSTOMER OPTION-6

```
Enter the old password : *******

Enter the new password : *******
```

CUSTOMER OPTION-7

```
You have no fines due.
```

CUSTOMER OPTION-8

```
You have no notifications.
```

CUSTOMER OPTION-9

```
Enter the problem :
```

CUSTOMER OPTION-10

```
Logging out…
```

DATA STRUCTURES (PARALLEL ARRAYS):

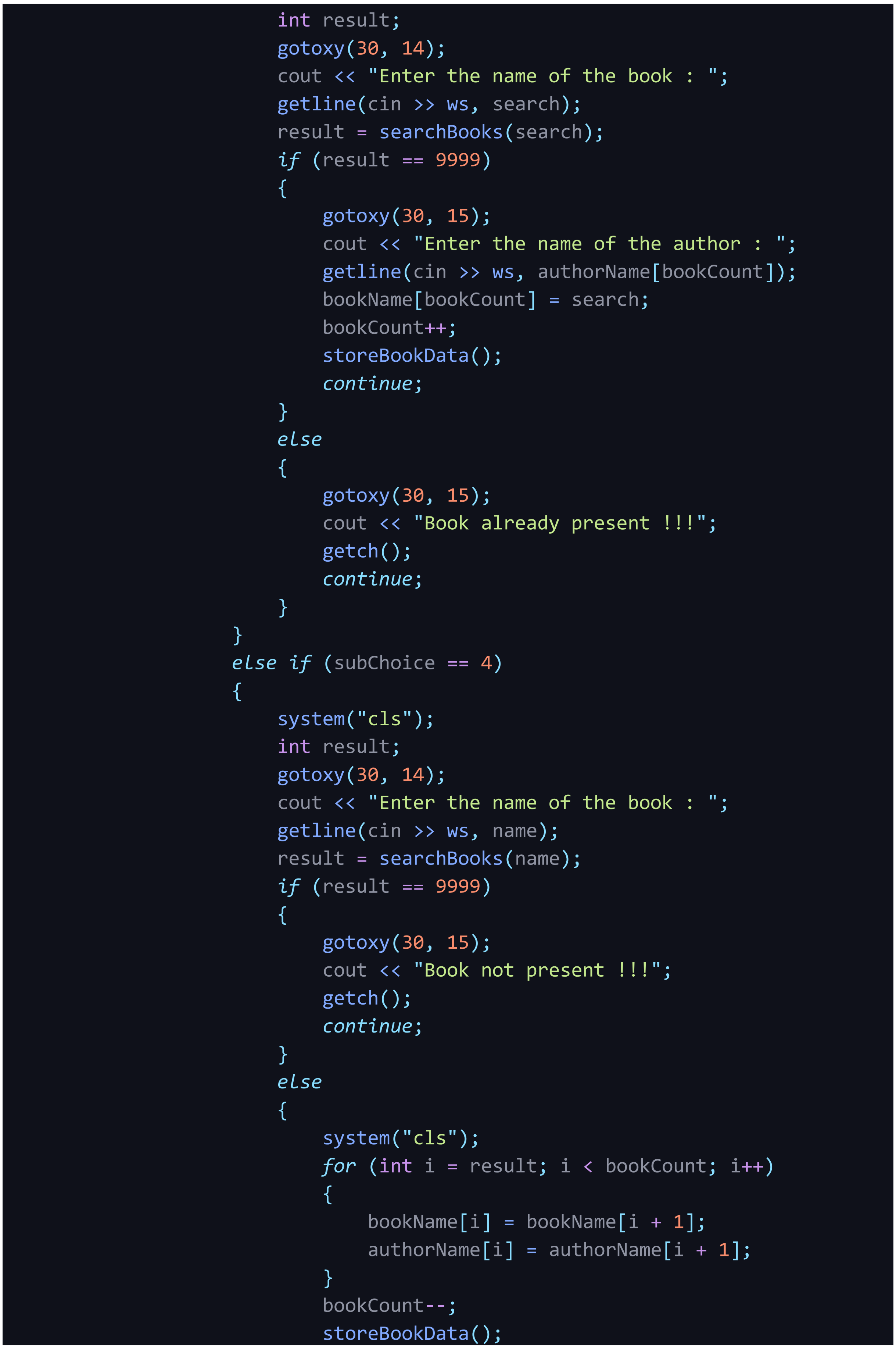
```
string usernames[100];
string passwords[100];
string roles[100];
string borrowedBooks[50][50];
string bookName[500];
string authorName[500];
int userCount = 0;
int bookCount = 0;
int accountTrack = 0;
int navChoice = 1;
int bookCounter = 0;
```

FUNCTION PROTOTYPES :

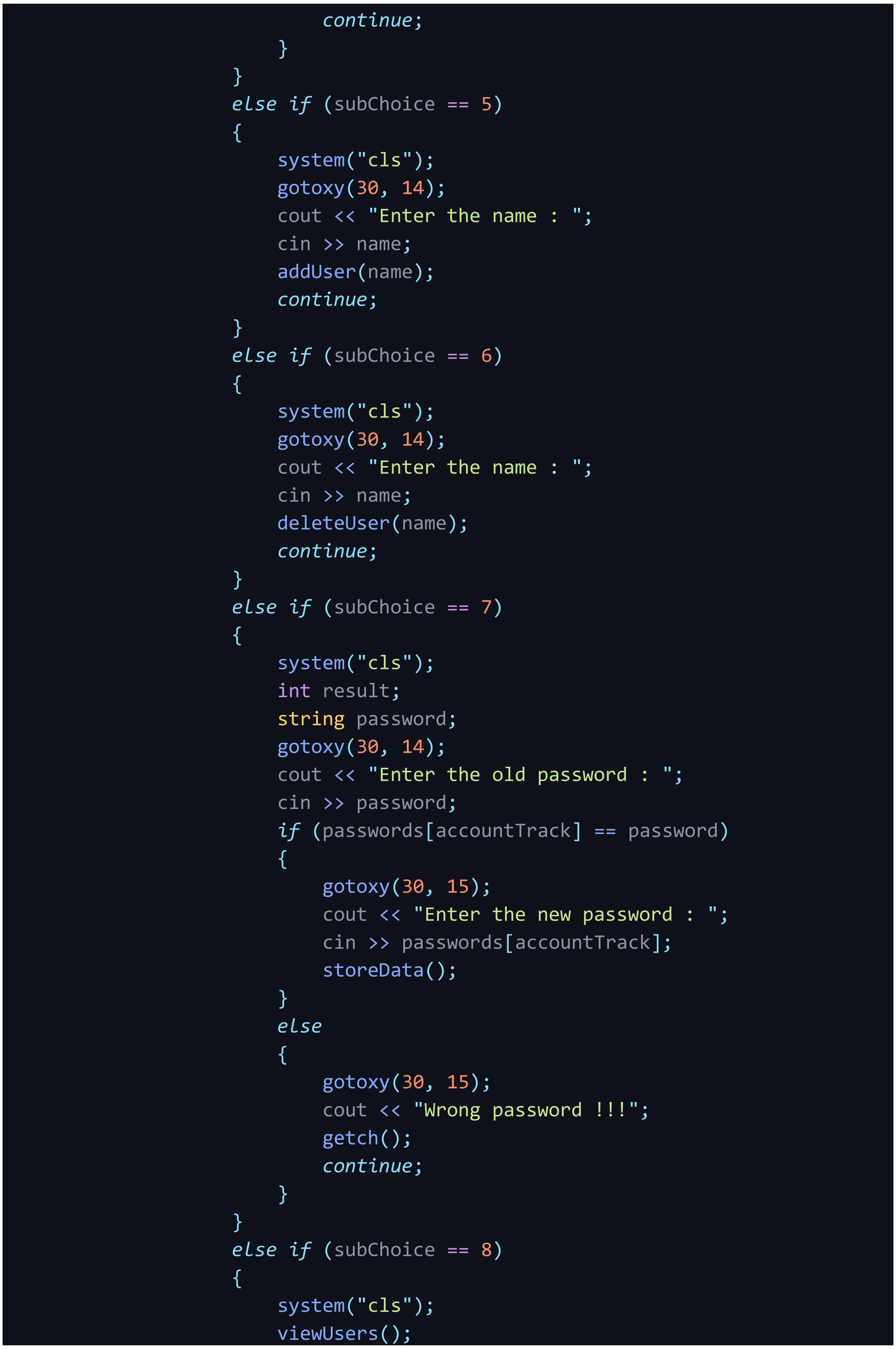
```
void signUpAUser(string username, string password);
string signInAUser(string username, string password);
void viewUsers();
void storeData();
int isValid(string username);
void clearPage(int choice, int bookCount);
int signMenu();
void readData();
void readBookData();
void storeBookData();
void logo();
int adminMenu();
int userMenu();
void viewBooks();
int searchBooks(string);
void gotoxy(int, int);
void showDateAndTime();
void addUser(string);
void deleteUser(string);
void backup();
int navigation(int);
void requestBook(string, string);
void adminSubMenu();
void userSubMenu();
void mainMenu();
void readBorrowedBooks(string);
string getField(string, int);
int searchBorrowedBook(string name);
```

COMPLETE CODE :

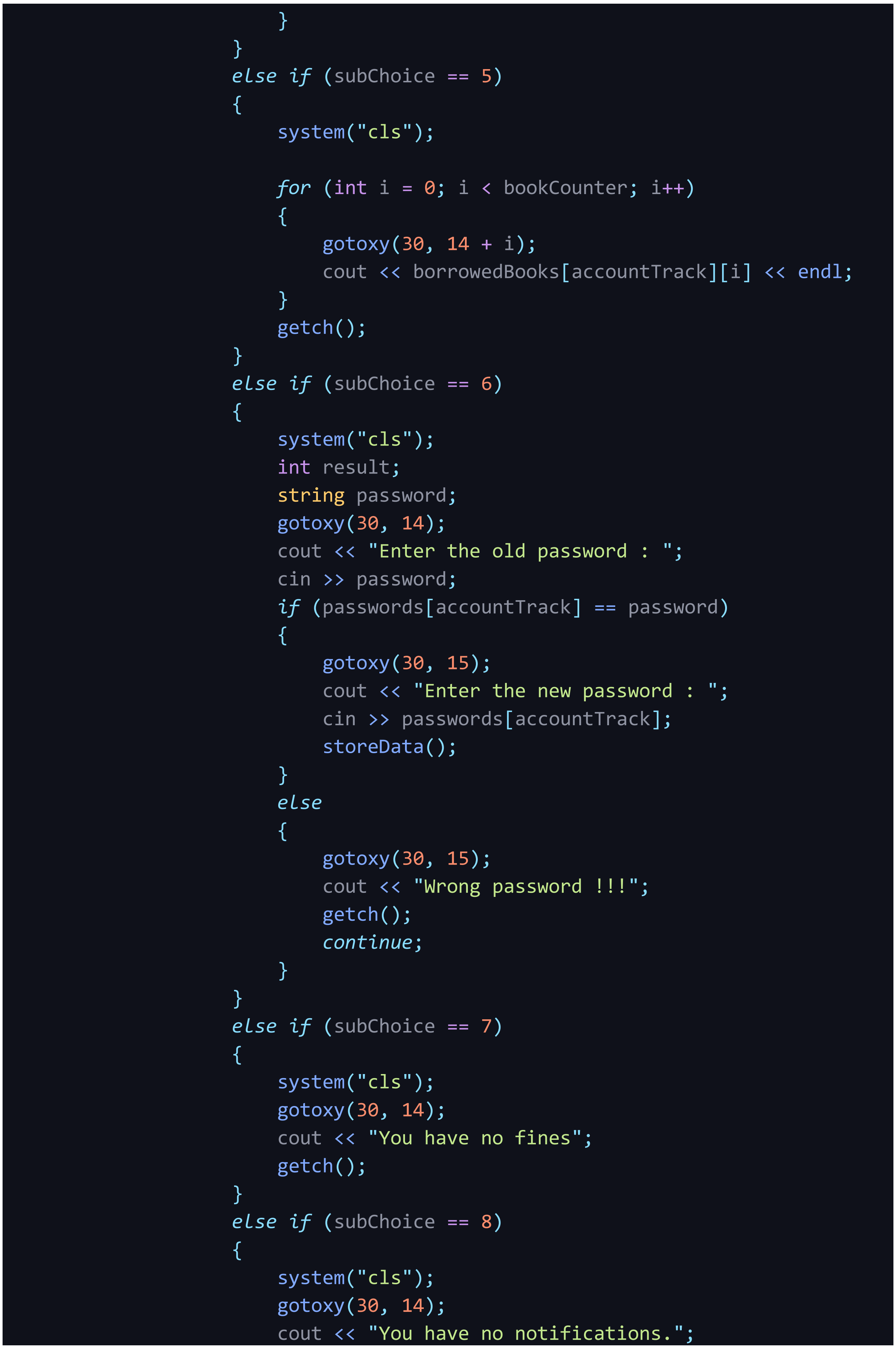
```cpp
#include <iostream>
#include <fstream>
#include <windows.h>
#include <conio.h>
#include <string>
#include <ctime>

using namespace std;

// Global Variables

string usernames[100];
string passwords[100];
string roles[100];
string borrowedBooks[50][50];
string bookName[500];
string authorName[500];
int userCount = 0;
int bookCount = 0;
int bookCounter = 0;
int accountTrack = 0;
int navChoice = 1;
fstream myFile;
HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
time_t t = time(0);
tm *ta = localtime(&t);

// Function Prototypes

void signUpAUser(string username, string password);
string signInAUser(string username, string password);
void viewUsers();
void storeData();
int isValid(string username);
void clearPage(int choice, int bookCount);
int signMenu();
void readData();
void readBookData();
void storeBookData();
void logo();
int adminMenu();
int userMenu();
void viewBooks();
int searchBooks(string);
void gotoxy(int, int);
void showDateAndTime();
void addUser(string);
void deleteUser(string);
void backup();
int navigation(int);
void requestBook(string, string);
void adminSubMenu();
void userSubMenu();
void mainMenu();
void readBorrowedBooks(string);
string getField(string, int);
int searchBorrowedBook(string name);

int main()
{
    // Loading Data From Files
    readData();
    readBookData();
    string username, password, name;
    int choice;
    int subChoice;
    mainMenu();
    // The Start of the Modules
    while (true)
    {
        choice = signMenu();
        if (choice == 1) // SignUp Menu, adds a user is not already present.
        {
            system("cls");
            gotoxy(30, 14);
            cout << "Enter the name : ";
            getline(cin >> ws, username);
            int res = isValid(username);
            if (res == 9999)
            {
                gotoxy(30, 15);
                cout << "Enter the password : ";
                cin >> password;
                signUpAUser(username, password);
            }
            else
            {
                cout << "User already present" << endl;
                getch();
                continue;
            }
        }
        else if (choice == 2) // Sign In Menu, signs in a user if present
        {
            system("cls");
            gotoxy(30, 14);
            cout << "Enter the name : ";
            getline(cin >> ws, username);
            gotoxy(30, 15);
            cout << "Enter the password : ";
            cin >> password;
            if (signInAUser(username, password) == "admin")
            {
                while (true)
                {
                    subChoice = adminMenu();
                    if (subChoice == 1)
                    {
                        system("cls");
                        viewBooks();
                        continue;
                    }
                    else if (subChoice == 2)
                    {
                        system("cls");
                        int result;
                        gotoxy(30, 14);
                        cout << "Enter the name of the book or the name of the 
author: ";
                        getline(cin >> ws, name);
                        result = searchBooks(name);
                        if (result == 9999)
                        {
                            gotoxy(30, 15);
                            cout << "Book not found";
                            getch();
                            continue;
                        }
                        else
                        {
                            gotoxy(30, 15);
                            cout << bookName[result] << "\t\t\t" << 
authorName[result];
                            getch();
                            continue;
                        }
                    }
                    else if (subChoice == 3)
                    {
                        system("cls");
                        string search;
```



```
                        int result;
                        gotoxy(30, 14);
                        cout << "Enter the name of the book : ";
                        getline(cin >> ws, search);
                        result = searchBooks(search);
                        if (result == 9999)
                        {
                            gotoxy(30, 15);
                            cout << "Enter the name of the author : ";
                            getline(cin >> ws, authorName[bookCount]);
                            bookName[bookCount] = search;
                            bookCount++;
                            storeBookData();
                            continue;
                        }
                        else
                        {
                            gotoxy(30, 15);
                            cout << "Book already present !!!";
                            getch();
                            continue;
                        }
                    }
                    else if (subChoice == 4)
                    {
                        system("cls");
                        int result;
                        gotoxy(30, 14);
                        cout << "Enter the name of the book : ";
                        getline(cin >> ws, name);
                        result = searchBooks(name);
                        if (result == 9999)
                        {
                            gotoxy(30, 15);
                            cout << "Book not present !!!";
                            getch();
                            continue;
                        }
                        else
                        {
                            system("cls");
                            for (int i = result; i < bookCount; i++)
                            {
                                bookName[i] = bookName[i + 1];
                                authorName[i] = authorName[i + 1];
                            }
                            bookCount--;
                            storeBookData();
```



```
                            continue;
                        }
                    }
                    else if (subChoice == 5)
                    {
                        system("cls");
                        gotoxy(30, 14);
                        cout << "Enter the name : ";
                        cin >> name;
                        addUser(name);
                        continue;
                    }
                    else if (subChoice == 6)
                    {
                        system("cls");
                        gotoxy(30, 14);
                        cout << "Enter the name : ";
                        cin >> name;
                        deleteUser(name);
                        continue;
                    }
                    else if (subChoice == 7)
                    {
                        system("cls");
                        int result;
                        string password;
                        gotoxy(30, 14);
                        cout << "Enter the old password : ";
                        cin >> password;
                        if (passwords[accountTrack] == password)
                        {
                            gotoxy(30, 15);
                            cout << "Enter the new password : ";
                            cin >> passwords[accountTrack];
                            storeData();
                        }
                        else
                        {
                            gotoxy(30, 15);
                            cout << "Wrong password !!!";
                            getch();
                            continue;
                        }
                    }
                    else if (subChoice == 8)
                    {
                        system("cls");
                        viewUsers();
                    }
                    else if (subChoice == 9)
                    {
                        system("cls");
                        gotoxy(30, 14);
                        cout << "Creating Backup..." << endl;
                        Sleep(1000);
                        backup();
                        gotoxy(30, 15);
                        cout << "Backup Complete " << endl;
                        Sleep(1000);
                    }
                    else if (subChoice == 10)
                    {
                        break;
                    }
                }
            }
            else if (signInAUser(username, password) == "client")
            {
                while (true)
                {
                    subChoice = userMenu();
                    readBorrowedBooks(usernames[accountTrack]);
                    if (subChoice == 1)
                    {
                        system("cls");
                        viewBooks();
                        continue;
                    }
                    else if (subChoice == 2)
                    {
                        system("cls");
                        int result;
                        gotoxy(30, 14);
                        cout << "Enter the name of the book or the name of the
author: ";
                        getline(cin >> ws, name);
                        result = searchBooks(name);
                        if (result == 9999)
                        {
                            gotoxy(30, 15);
                            cout << "Book not found";
                            getch();
                            continue;
                        }
                        else
                        {
                            gotoxy(30, 15);
                            cout << bookName[result] << "\t\t\t" <<
authorName[result];
                            getch();
                            continue;
                        }
                    }
                    else if (subChoice == 3)
                    {
                        system("cls");
                        int result;
                        gotoxy(30, 14);
                        cout << "Enter the name of the book : ";
                        getline(cin >> ws, name);
                        result = searchBooks(name);
                        if (result == 9999)
                        {
                            gotoxy(30, 15);
                            cout << "Book not found";
                            getch();
                            continue;
                        }
                        else
                        {
                            requestBook(usernames[accountTrack], name);
                        }
                    }
                    else if (subChoice == 4)
                    {
                        system("cls");
                        int result;
                        gotoxy(30, 14);
                        cout << "Enter the name : ";
                        getline(cin >> ws, name);
                        result = searchBooks(name);
                        if (result == 9999)
                        {
                            gotoxy(30, 15);
                            cout << "Your request for the book has been noted.
You will be notified when it becomes available." << endl;
                            getch();
                        }
                        else
                        {
                            gotoxy(30, 15);
                            cout << "The book is already present. You can take
it from the nearest library." << endl;
                            getch();
```



```
    }
}
else if (subChoice == 5)
{
    system("cls");

    for (int i = 0; i < bookCounter; i++)
    {
        gotoxy(30, 14 + i);
        cout << borrowedBooks[accountTrack][i] << endl;
    }
    getch();
}
else if (subChoice == 6)
{
    system("cls");
    int result;
    string password;
    gotoxy(30, 14);
    cout << "Enter the old password : ";
    cin >> password;
    if (passwords[accountTrack] == password)
    {
        gotoxy(30, 15);
        cout << "Enter the new password : ";
        cin >> passwords[accountTrack];
        storeData();
    }
    else
    {
        gotoxy(30, 15);
        cout << "Wrong password !!!";
        getch();
        continue;
    }
}
else if (subChoice == 7)
{
    system("cls");
    gotoxy(30, 14);
    cout << "You have no fines";
    getch();
}
else if (subChoice == 8)
{
    system("cls");
    gotoxy(30, 14);
    cout << "You have no notifications.";
```

```
                        getch();
                    }
                    else if (subChoice == 9)
                    {
                        system("cls");
                        string word;
                        gotoxy(30, 14);
                        cout << "Enter your problem : ";
                        cin >> word;
                        system("cls");
                        gotoxy(30, 14);
                        cout << "Your request has been noted. An admin will be
contacted shortly." << endl;
                        getch();
                    }
                    else if (subChoice == 10)
                    {
                        break;
                    }
                }
            }
            else
            {
                cout << "Wrong user or password";
                getch();
                continue;
            }
        }
        else if (choice == 3)
        {
            break;
        }
    }
}

// Displays the Sign in Menu

int signMenu()
{
    SetConsoleTextAttribute(hConsole, 3);
    system("cls");
    logo();
    int choice;
    cout << endl
         << endl
         << endl
         << endl
         << endl
         << endl;
    cout << "                                 Sign Up" << endl;
    cout << "                                 Sign In" << endl;
    cout << "                                 Exit" << endl;
    choice = navigation(3);
    return choice;
}

void spaces()
{
    cout << endl
         << endl
         << endl
         << endl
         << endl
         << endl
         << endl
         << endl
         << endl
         << endl
         << endl
         << endl;
    cout << "                                      ";
}

void signUpAUser(string username, string password)
{
    usernames[userCount] = username;
    passwords[userCount] = password;
    roles[userCount] = "client";
    userCount++;
    storeData();
}

string signInAUser(string username, string password)
{
    for (int i = 0; i < userCount; i++)
    {
        if (username == usernames[i] && password == passwords[i])
        {
            if (roles[i] == "admin")
            {
                accountTrack = i;
                return "admin";
            }
            else if (roles[i] == "client")
            {
                accountTrack = i;
                return "client";
            }
        }
    }
    return "none";
}

// Reads user data from the Files, in csf format

void readData()
{
    string record;
    fstream data;
    data.open("users.txt", ios::in);
    if (data.is_open())
    {
        while (!(data.eof()))
        {
            getline(data, record);
            usernames[userCount] = getField(record, 1);
            passwords[userCount] = getField(record, 2);
            roles[userCount] = getField(record, 3);
            userCount = userCount + 1;
        }
        userCount = userCount - 1;
    }
    else
    {
        usernames[0] = "AbdulRehman";
        passwords[0] = "123456";
        roles[0] = "admin";
        userCount++;
    }
}

string getField(string record, int field)
{
    int commaCount = 1;
    string item;
    for (int x = 0; x < record.length(); x++)
    {
        if (record[x] == ',')
        {
            commaCount = commaCount + 1;
        }
        else if (commaCount == field)
        {
            item = item + record[x];
        }
    }
    return item;
}

// Reads the Book data from the Files, in csf format

void readBookData()
{
    string record;
    fstream data;
    data.open("books.txt", ios::in);
    if (data.is_open())
    {
        while (!(data.eof()))
        {
            getline(data, record);
            bookName[bookCount] = getField(record, 1);
            authorName[bookCount] = getField(record, 2);
            bookCount = bookCount + 1;
        }
        bookCount = bookCount - 1;
    }
}

// Displays Date and Time at the top of the screen

void showDateAndTime()
{
    gotoxy(150, 1);
    cout << ta->tm_mday << "/" << ta->tm_mon + 1 << "/" << (1900 + ta-
>tm_year) << endl;
    int hour = ta->tm_hour;
    if (hour > 11)
    {
        hour = hour - 12;
    }
    gotoxy(150, 2);
    if(ta->tm_min < 10)
    {
        cout << hour << ":" << "0" << ta->tm_min;
    }
    cout << hour << ":" << ta->tm_min;
}

// Shows all user present in the program

void viewUsers()
```

```
{
    for (int i = 0; i < userCount; i++)
    {
        gotoxy(30, 14 + i);
        cout << usernames[i];
    }
    getch();
}

// Displays the Main Menu

void mainMenu()
{
    int i = 1;
    system("cls");
    SetConsoleTextAttribute(hConsole, 3);
    cout << endl
         << endl
         << endl;
    cout <<
"                                                      ________________________
____________________________ " << endl;
    cout <<
"                                                     |________________________
____________________________| " << endl;
    cout << "                                                     |
__     __   ____   ___ ||  ____    ____     _  __  | " << endl;
    cout << "                                                    ||  |__ |--
|_| || |_|   |||_|**|*|__|+|+||___| ||  | | " << endl;
    cout << "                                                    ||==|^^||--|
|=||=| |=*=||| |~~|~|  |=|=|| | |~||==| | " << endl;
    cout << "                                                    ||  |##||  |
| || | |   |||-|  | |==|+|+||-|-|~||__| | " << endl;
    cout <<
"                                                     ||__|__||__|_|_||_|_|___|
||_|__|_|__|_|_||_|_|_||__|_| " << endl;
    cout <<
"                                                     ||_______________________
||__________________________| " << endl;
    cout << "                                                     |
_____________________  ||       __   __  _  __     _ | " << endl;
    cout <<
"                                                     ||=|=|=|=|=|=|=|=|=|=|=|
__..\\/ |  |_|  ||#||==|  / /| " << endl;
    cout << "                                                    || | | | | |
| | | | | |/\\ \\  \\\\|++|=|  || ||==| / / | " << endl;
```

```
    cout <<
"                                                    ||_|_|_|_|_|_|_|_|_|_|_|_|/_
/\\_.___\\__|_|__||_||__|/_/__| " << endl;
    cout <<
"                                                    |____________________
/\\~()/()~//\\ __________________| " << endl;
    cout << "                                                     |
__   __    _ _     \\_  (_ .  _/ _    ___     _____| " << endl;
    cout << "                                                     ||~~|_|..|__|
|| |_ _   \\ //\\\\ /  |=|__|~|~|___| | | | " << endl;
    cout << "                                                     ||--
|+|^^|==|1||2| | |__/\\ __ /\\__| |==|x|x|+|+|=|=|=| " << endl;
    cout <<
"                                                    ||__|_|__|__|_||_|_|
/  \\ \\  / /  \\_|__|_|_|_|_|_|_|_| " << endl;
    cout <<
"                                                    |_________________
_/    \\/\\/\\/    \\_ _______________| " << endl;
    cout << "                                                     |
_____   _   __  |/      \\../      \\|  __   __   ___| " << endl;
    cout << "                                                     ||_____|_|
|_|##|_||   |   \\/ __|   ||_|==|_|++|_|-||| " << endl;
    cout <<
"                                                    ||______||=|#|--|
|\\   \\   o    /   /| |  |~|  | | ||| " << endl;
    cout <<
"                                                    ||______||_|_|__|_|_\\
\\  o   /   /_|_|__|_|__|_|_||| " << endl;
    cout << "                                                     |_________
__________\\___\\____/___/___________ ______| " << endl;
    cout <<
"                                                    |__    _  /    ________
  ______          /| _ _ _| " << endl;
    cout << "                                                     |\\
\\  |=|/   //    /| //   /  /  / |        / ||%|%|%| " << endl;
    cout << "                                                     | \\/\\
|*/  .//____//.//   /__/__/ (_)      /  ||=|=|=| " << endl;
    cout <<
"                                                    |  \\/\\|/   /(____|/
//                    /  /||~|~|~| " << endl;
    cout <<
"                                                    |___\\_/   /_______//
________         /  / ||_|_|_| " << endl;
    cout << "                                                     |___
/   (|________/   |\\_______\\       /  /| |______| " << endl;
    cout <<
"                                                        /                 \\
|________)     /  / | |        " << endl;
```

```
    cout << endl
         << endl;
    cout <<
"                                                                                Welcome
to my library" << endl;

    getch();
}

// Checks if a user is already present

int isValid(string username)
{
    for (int i = 0; i < userCount; i++)
    {
        if (username == usernames[i])
        {
            return i;
        }
    }
    return 9999;
}

// Store the user Data in the Files, in csf format

void storeData()
{
    myFile.open("users.txt", ios::out);
    for (int i = 0; i < userCount; i++)
    {
        myFile << usernames[i] << ",";
        myFile << passwords[i] << ",";
        myFile << roles[i] << endl;
    }
    myFile.close();
}

// Store the book Data in the Files, in csf format

void storeBookData()
{
    myFile.open("books.txt", ios::out);
    for (int i = 0; i < bookCount; i++)
    {
        myFile << bookName[i] << ",";
        myFile << authorName[i] << endl;
    }
    myFile.close();
```

```
}

// Displays the logo

void logo()
{
    cout << "                                                  _      _
_                             _____           _                   " << endl;
    cout << "                                                 | |    (_)
|                           / ____|         | |                  " << endl;
    cout << "                                                 | |     _| |__  _
__ __ _ _ __ _   _  | (___  _   _ ___| |_ ___ _ __ ___   " << endl;
    cout << "                                                 | |    | | '_ \\|
'__/ _` | '__| | | |  \\___ \\| | | / __| __/ _ \\ '_ ` _ \\  " << endl;
    cout << "                                                 | |____| | |_) |
| | (_| | |  | |_| |  ____) | |_| \\__ \\ ||  __/ | | | | | " << endl;
    cout <<
"                                                |______|_|_.__/|_|  \\__,_|_|
  \\__, | |_____/ \\__, |___/\\__\\___|_| |_| |_| " << endl;
    cout <<
"
   __/ |          __/ |                           " << endl;
    cout <<
"
  |___/          |___/                            " << endl;
    cout << endl;
    cout <<
"******************************************************************************
*******************************************************************************
************* " << endl;
    cout << endl;
}

// Displays the logo

void adminSubMenu()
{
    cout <<
"
                                                              _           _
    __  __                  " << endl;
    cout << "                                                    /\\      |
|         (_)      |  \\/  |                  " << endl;
    cout << "                                                   /  \\   __| |_
__ ___  _ _ __   | \\  / | ___ _ __  _   _  " << endl;
    cout << "                                                  / /\\ \\ / _` |
'_ ` _ \\| | '_ \\  | |\\/| |/ _ \\ '_ \\| | | | " << endl;
    cout << "                                                 / ____ \\ (_| | |
| | | | | | | | | |  | |  __/ | | | |_| | " << endl;
```

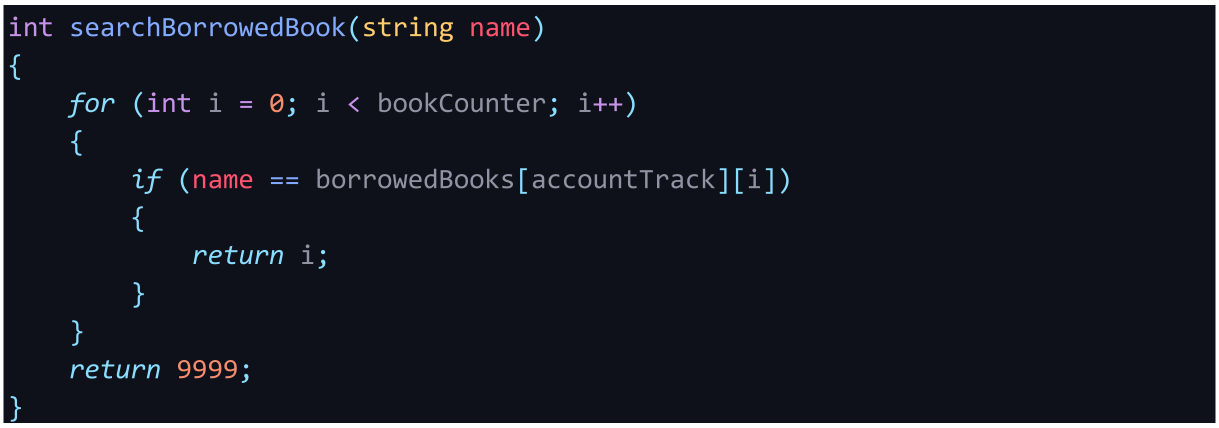
```
    cout <<
"                                               /_/      \\_\\__,_|_| |_|
|_|_|_|_| |_| |_|  |_|\\___|_| |_|\\__,_| " << endl;
    cout << endl;
    cout <<
"*****************************************************************************
******************************************************************************
************* " << endl;
    cout << endl;
    cout << endl
         << endl
         << endl
         << endl
         << endl
         << endl
         << endl
         << endl;
}

// Displays the admin options

int adminMenu()
{
    system("cls");
    SetConsoleTextAttribute(hConsole, 4);
    int choice = 0;
    adminSubMenu();
    cout << "                            View all the book available" << endl;
    cout << "                            Search a book" << endl;
    cout << "                            Add a book" << endl;
    cout << "                            Remove a book" << endl;
    cout << "                            Add an account" << endl;
    cout << "                            Delete an account " << endl;
    cout << "                            Change password" << endl;
    cout << "                            Show all users" << endl;
    cout << "                            Backup data" << endl;
    cout << "                            Log out" << endl;
    choice = navigation(10);
    return choice;
}

// Displays the logo

void userSubMenu()
{
    cout << "                                          _____ _
_            _     __  __                    " << endl;
```

```
    cout << "                                         / ____| (_)          |
|   |  \\/  |                  " << endl;
    cout << "                                        | |      | |_  ___ _ __ |
|_  | \\  / | ___ _ __  _   _  " << endl;
    cout << "                                        | |      | | |/ _ \\ '_ \\|
__| | |\\/| |/ _ \\ '_ \\| | | | " << endl;
    cout << "                                        | |___| | |  __/ | | |
|_  | |  | |  __/ | | | |_| | " << endl;
    cout << "                                         \\_____|_|_|\\___|_|
|_|\\__| |_|  |_|\\___|_| |_|\\__,_| " << endl;
    cout << endl;
    cout <<
"******************************************************************************
*******************************************************************************
************* " << endl;
    cout << endl;
    cout << endl
         << endl
         << endl
         << endl
         << endl
         << endl
         << endl
         << endl;
}

// Displays the User options

int userMenu()
{
    system("cls");
    SetConsoleTextAttribute(hConsole, 2);
    int option;
    userSubMenu();
    cout << "                            View the books" << endl;
    cout << "                            Search the books" << endl;
    cout << "                            Borrow a book" << endl;
    cout << "                            Request a book" << endl;
    cout << "                            Show borrowed book" << endl;
    cout << "                            Change password" << endl;
    cout << "                            Checking fines" << endl;
    cout << "                            Notifications" << endl;
    cout << "                            Get help" << endl;
    cout << "                            Log out" << endl;
    option = navigation(10);
    return option;
}
```

```
// Shows all of the available books, can navigate using arrow keys

void viewBooks()
{
    system("cls");
    logo();
    int choice = 1;
    while (true)
    {
        int j = 0;
        gotoxy(10, 17);
        cout << "Book";
        gotoxy(55, 17);
        cout << "Author";
        int totalPages = (bookCount / 10 + 1);
        if (GetAsyncKeyState(VK_LEFT))
        {
            if (choice > 1)
            {
                clearPage(choice, bookCount);
                choice--;
            }
        }
        else if (GetAsyncKeyState(VK_RIGHT))
        {
            if (choice < totalPages)
            {
                clearPage(choice, bookCount);
                choice++;
            }
        }
        else if (GetAsyncKeyState(VK_ESCAPE))
        {
            break;
        }
        for (int i = (0 + 10 * (choice - 1)); i < 10 * choice && i <=
bookCount && (j <= bookCount && j <= 9); i++, j++)
        {
            gotoxy(10, 18 + j);
            cout << bookName[i];
            gotoxy(55, 18 + j);
            cout << authorName[i];
        }
        Sleep(50);
    }
}

// Used to clear a page of books
```

```
void clearPage(int choice, int bookCount)
{
    int j = 0;
    for (int i = (0 + 10 * (choice - 1)); i < 10 * choice && (j <= bookCount
&& j <= 9); i++, j++)
    {
        gotoxy(10, 18 + j);
        cout << "                                  ";
        gotoxy(55, 18 + j);
        cout << "                                  ";
    }
}

// Used to move the cusror to a specific position

void gotoxy(int x, int y)
{
    COORD coord;
    coord.X = x;
    coord.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
}

// Used to search for a specific book

int searchBooks(string name)
{
    for (int i = 0; i < bookCount; i++)
    {
        if (name == bookName[i] || name == authorName[i])
        {
            return i;
        }
    }
    return 9999;
}

// Used by admin to add a new user

void addUser(string name)
{
    string password, role;
    if (isValid(name) == 9999)
    {
        gotoxy(30, 15);
        cout << "Enter the password : ";
        cin >> password;
        signUpAUser(name, password);
        gotoxy(30, 16);
        cout << "Enter the role (admin, client) : ";
        cin >> role;
        roles[userCount - 1] = role;
        storeData();
    }
}

// Used by admin to delete a user

void deleteUser(string name)
{
    int result = isValid(name);
    if (result == 9999)
    {
        cout << "User not present !!!";
        getch();
    }
    else
    {
        for (int i = result; i < userCount; i++)
        {
            usernames[i] = usernames[i + 1];
            passwords[i] = passwords[i + 1];
        }
        userCount--;
    }
    storeData();
}

// Used to save all the data oof library in seperate files

void backup()
{
    myFile.open("backupusers.txt", ios::out);
    for (int i = 0; i < userCount; i++)
    {
        myFile << usernames[i] << ",";
        myFile << passwords[i] << ",";
        myFile << roles[i] << endl;
    }
    myFile.close();
    myFile.open("backupbooks.txt", ios::out);
    for (int i = 0; i < bookCount; i++)
    {
        myFile << bookName[i] << ",";
        myFile << authorName[i] << ",";
    }
    myFile.close();
}

// Used to enable the user to select options using arrow keys

int navigation(int options)
{
    navChoice = 1;
    while (true)
    {
        showDateAndTime();
        gotoxy(25, 17 + navChoice);
        cout << ">";
        // if (GetAsyncKeyState(VK_UP))
        // {
        //     if (navChoice > 1)
        //     {
        //         gotoxy(7, 8 + navChoice);
        //         cout << " ";
        //         navChoice--;
        //     }
        // }
        if (GetAsyncKeyState(VK_UP))
        {
            navChoice--;
            gotoxy(25, 17 + navChoice);
            cout << " ";
            gotoxy(25, 18 + navChoice);
            cout << " ";
            if (navChoice < 1)
            {
                navChoice = options;
            }
        }
        else if (GetAsyncKeyState(VK_DOWN))
        {
            gotoxy(25, 17 + navChoice);
            cout << " ";
            navChoice++;
            if (navChoice > options)
            {
                navChoice = 1;
            }
        }
        // else if (GetAsyncKeyState(VK_RIGHT))
        // {
        //     return navChoice;
```

```cpp
        // }
        else if (getch() == 13)
        {
            return navChoice;
        }
        Sleep(70);
    }
}

// Used by user to borrow a book and store the data;

void requestBook(string username, string book)
{
    int result = searchBorrowedBook(book);
    if (result == 9999)
    {
        username = username + ".txt";
        borrowedBooks[accountTrack][bookCounter] = book;
        myFile.open(username, ios::app);
        myFile << borrowedBooks[accountTrack][bookCounter] << endl;
        myFile.close();
        bookCounter++;
    }
    else
    {
        gotoxy(30, 15);
        cout << "Book already borrowed";
        getch();
    }
}

void readBorrowedBooks(string username)
{
    username = username + ".txt";
    myFile.open(username, ios::in);
    if (myFile.is_open())
    {
        int i = 0;
        while (!myFile.eof())
        {
            getline(myFile >> ws, borrowedBooks[accountTrack][i]);
            i++;
        }
        bookCounter = i - 1;
    }
    myFile.close();
}
```



```
int searchBorrowedBook(string name)
{
    for (int i = 0; i < bookCounter; i++)
    {
        if (name == borrowedBooks[accountTrack][i])
        {
            return i;
        }
    }
    return 9999;
}
```

WEAKNESSES IN APPLICATION :

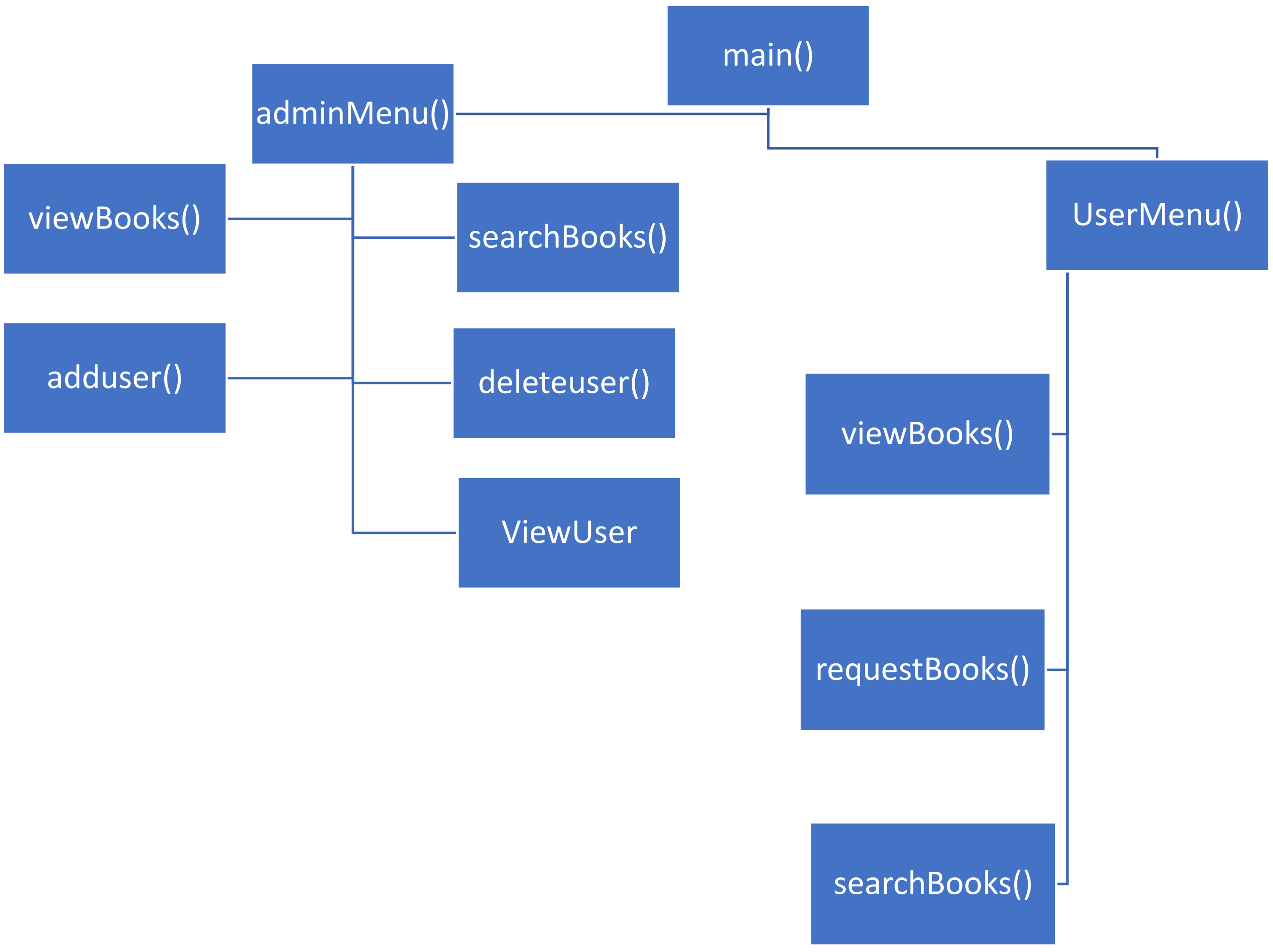
- Limited number of users can use the program.
- Only a limited amount of books can be added.
- The passwords are not securely saved.

FUTURE DIRECTION :

- User can track the books they have borrowed, check deadlines, receive fines and have a working notification system.
- Adding covers, summaries, and reviews to the book so that the user can have some information about the books

S

FUNCTION FLOW DIAGRAM :



RUBRICS :

Student Reg. No. : 2022-CS-57 **Student Name.** AbdulRehman

	A-Extensive Evidence	B-Convincing Evidence	C-Limited Evidence	D-No Evidence
Documentation Formatting **Grade:**	All the documentation meets all the criteria.	Documentation is well formatted but some of the criteria is not fulfilled.	Documentation is required a lot of improvement.	Documentation is not Available
Documentation Formatting Criteria: In **Binder**, **Title** Page, **Header**-Footers, Font **Style**, Font **Size** all are all consistence and according to given **guidelines**. Project **Poster** is professionally design and well presented				
Documentation Contents **Grade:**	Documentation includes all of the criteria.	Documentation meet more than 80% of the criteria given.	Documentation meet more than 50% of the criteria.	When the documentation meet less than 50% of the criteria.
Documentation Contents Criteria: Title Page - **Table** of Contents - Project **Abstract** - **Functional** Requirements - **Wire** Frames –**Data Flow** Diagram- **Data** Structure (Arrays)-**Function** Headers and Description -Project **Code.** - **Weakness** in the Project and **Future** Directions. - **Conclusion** and What your **Learn** from the Project and Course and What is your **Future** Planning.				
Project Complexity **Grade:**	Project has at least 2 user's types and each user has at least 5 functionalities.	Project complexity meet 80% criteria given in extensive evidence	Project complexity meet 50% criteria given in extensive evidence	Project complexity meet less than 50% criteria given in extensive evidence
Code Style **Grade:**	All Code style criteria is followed	All code style criteria followed but some improvements required	lot of improvements required in coding style.	**Did not follow** code style,
Code Style Criteria: Consistent code style. Code is well indented. Variable and Function names are well defined. White Spaces are well used. Comments are added.				
Code Documentation Mapping **Grade:**	Code and documentation is synchronized.	Code and documentation does not synchronized at **some** places	Code and documentation does not synchronized at **many** places	Code and documentation **does not** synchronized.
Data Structure (Arrays) **Grade:**	Data structure is sufficient for the project requirements	Data Structure is sufficient but require improvement to meet project requirements.	Data structure is not sufficient and need a lot of improvement	Data Structure is not properly identified and declared.
Modularity **Grade:**	Meet all Modularity criteria	Meet all Modularity criteria but at some places it is missing	Do not sufficiently meet the modularity criteria.	No modularity or very minimum modularity.
Modularity criteria: Functions are defined for each major feature. Functions are independent (identify from parameter list and return types).				
Validations **Grade:**	Validations on all number type inputs are applied	Validations are applied but at some places it is missing.	Validations are missing at lot of places	No Validations are used
File Handling **Grade:**	Separate files for separate data. Data in csv format	File handing require some improvements	File handing require a lot of improvements	Not implemented
Aesthetics of the User Interface **Grade:**	UI is presentable. Proper coloring, Headers and clear screen is done	UI require some improvements	UI require a lot of improvements	Not implemented
Presentation and Demo **Grade:**	Presentation and Demo was 100% working	Presentation and Demo require some improvements	Presentation and Demo require a lot of improvements	Presentation was not ok and Demo was not working
Student Understanding with the Code. **Grade:**	Student has complete understanding how the code is working and knows the concept.	Student has good understand but some place he does not know the concepts	Student has a very little understand and lack the major concepts.	Student does not have any level of understanding of the code.

Checked by:	
Comments:	