Lanka Nippon BizTech Institute

Project Case Study

IT11024

Programming Fundamentals

Name	:	**Dinithi Lokugamage**
Index number	:	**UGC0122030**
Batch	:	**UGC SE 01**

“LIBRARY LUMINAR ”

LIBRARY MANAGEMENT SYSTEM

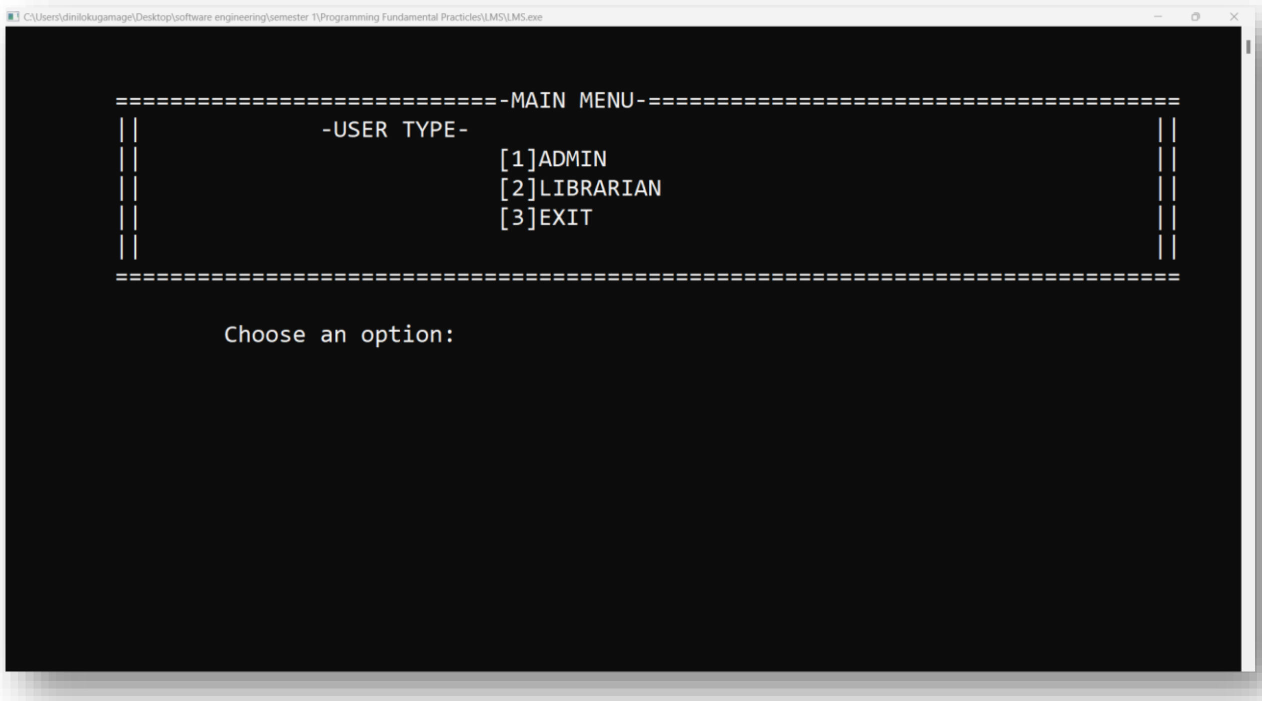
Table of Contents

1. Introduction

Keeping track of every detail in a busy library with a wide range of volumes may be challenging.My "Luminar" Library Management System approach to overcoming this difficulty. The major objective of LMS is to efficiently manage and arrange all jobs for the library. The Admin can easily manage librarians, modify penalties, and establish borrowing restrictions thanks to this technology. Librarians can handle book problems and returns efficiently, add, modify, or delete both borrower and book information, and even keep track of late fees for overdue books. Consider the time wasted on manual record-keeping; with our system, everything is quick and simple.

2. Provide Evidence for the system implementation.

(1) <u>Main menu</u>

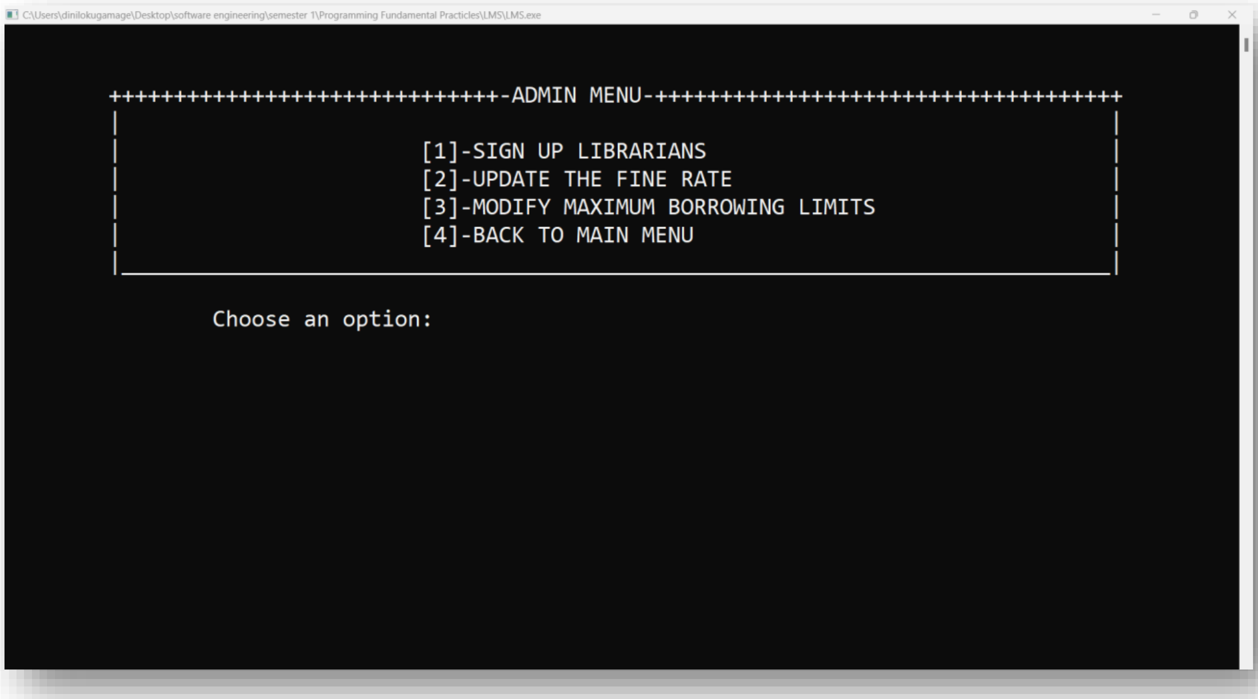


```
C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practicles\LMS\LMS.exe

        =============================-MAIN MENU-========================================
        ||             -USER TYPE-                                                    ||
        ||                           [1]ADMIN                                         ||
        ||                           [2]LIBRARIAN                                     ||
        ||                           [3]EXIT                                          ||
        ||                                                                            ||
        ================================================================================

                Choose an option:
```

The "Main Menu" acts as the gateway to the system, directing users based on their responsibilities. Individuals can select from "ADMIN" or "LIBRARIAN". Choosing "ADMIN" unlocks the ability to adjust system settings and advanced management features.

On the other side, the "LIBRARIAN" selection gives users the ability to manage, organize and track books. Additionally, if users who want to quit from the system, they can do so by clicking "EXIT".

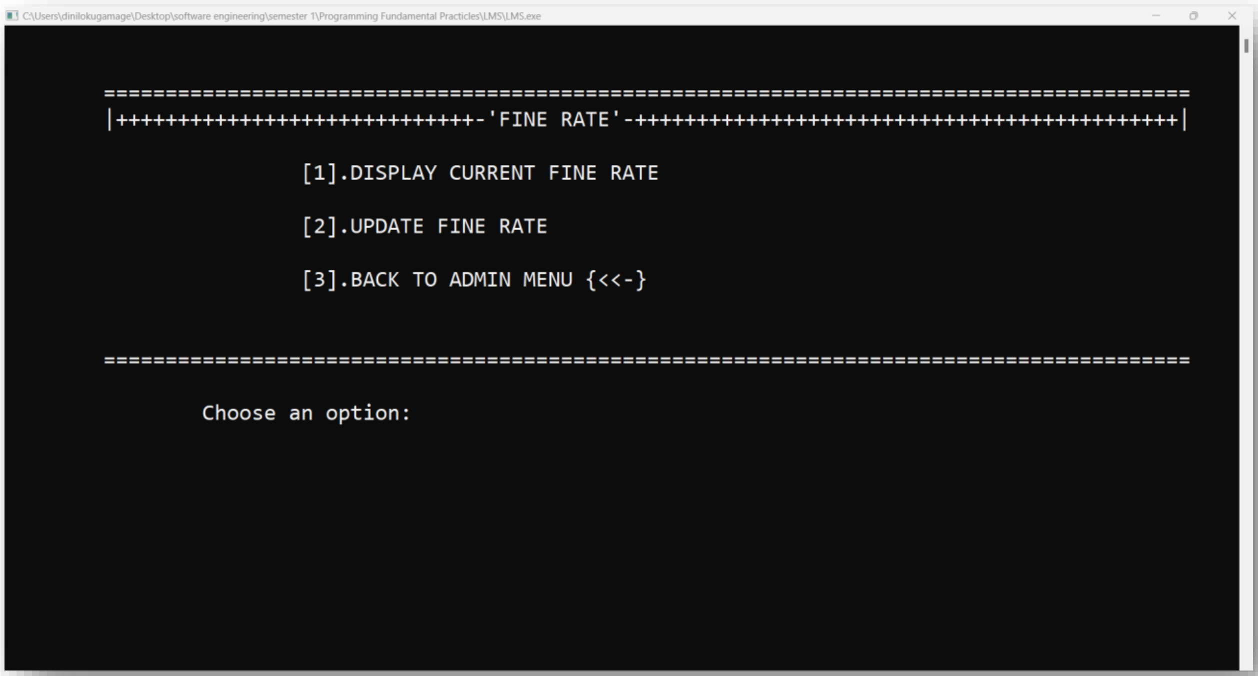
(2) Admin menu



```
C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practicles\LMS\LMS.exe

        ++++++++++++++++++++++++++++++-ADMIN MENU-++++++++++++++++++++++++++++++++++++
        |                                                                            |
        |                       [1]-SIGN UP LIBRARIANS                               |
        |                       [2]-UPDATE THE FINE RATE                             |
        |                       [3]-MODIFY MAXIMUM BORROWING LIMITS                  |
        |                       [4]-BACK TO MAIN MENU                                |
        |____________________________________________________________________________|

                Choose an option:
```

This is the main access point for the admin tasks. By selecting the first option, administrators can add "SIGN UP LIBRARIANS". The second option gives administrators the option to "UPDATE THE FINE RATE", allows them to adjust and set fines as necessary. They may "MODIFY MAXIMUM BORROWING LIMITS" in the third function to change the number of things a user can borrow continuously. Also, Administrators can travel "BACK TO THE MAIN MENU" using the fourth option if they want to go back to the beginning.

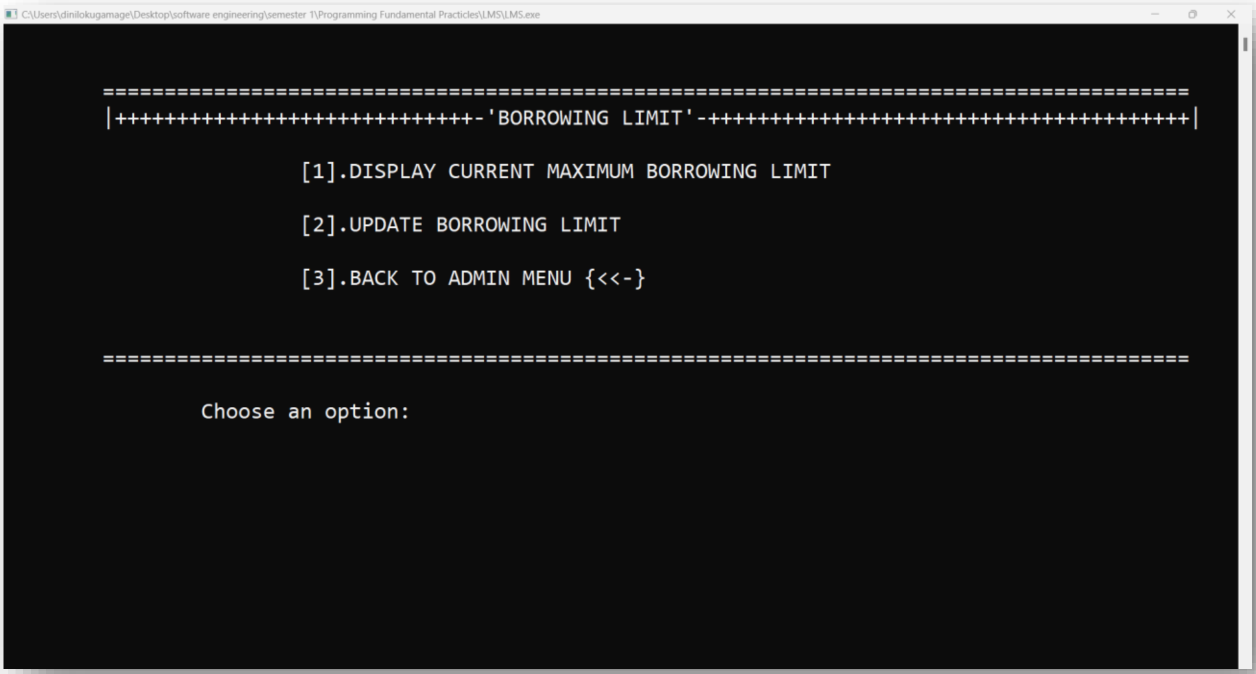
(3) Fine rate menu



```
C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practicles\LMS\LMS.exe

========================================================================================
|+++++++++++++++++++++++++++++-'FINE RATE'-+++++++++++++++++++++++++++++++++++++++++++|

                [1].DISPLAY CURRENT FINE RATE

                [2].UPDATE FINE RATE

                [3].BACK TO ADMIN MENU {<<-}


========================================================================================

        Choose an option:
```

Librarians can quickly view the current fine rate using option 1. If changes are required, the update feature allows for easy adjustments, ensuring fair and updated penalty charges using option 2.

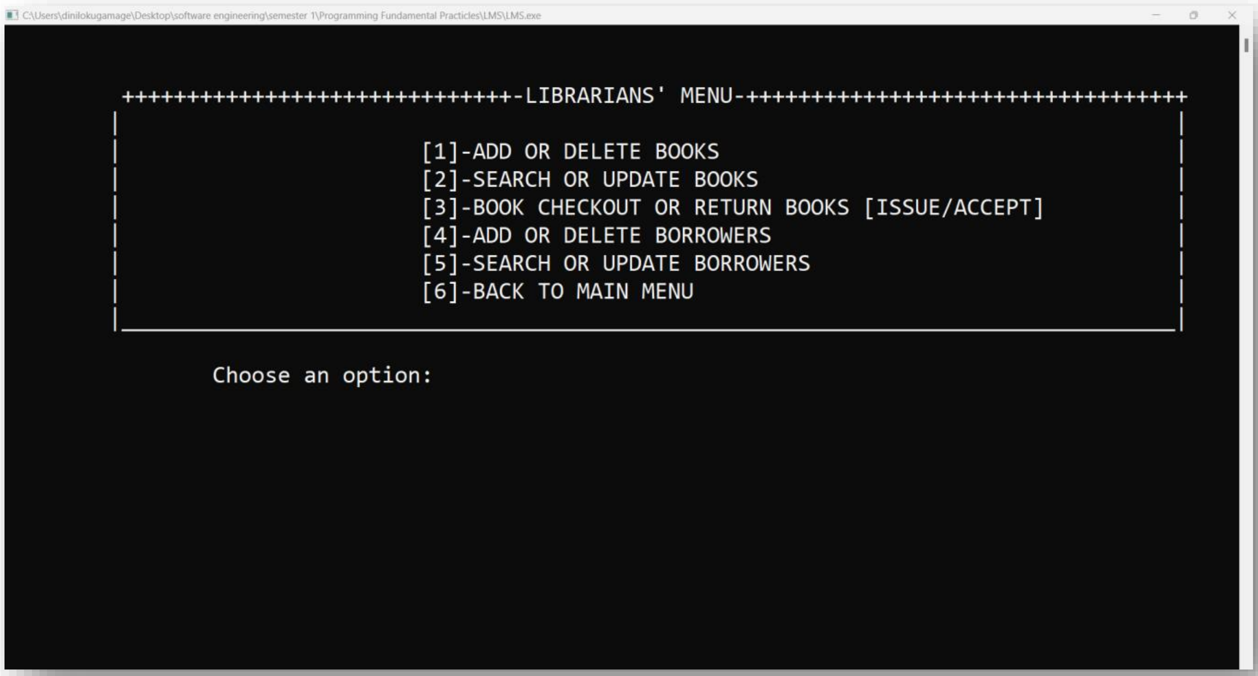
(4) <u>Borrowing limit menu</u>



```
C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practicles\LMS\LMS.exe

        =========================================================================================
        |++++++++++++++++++++++++++++++-'BORROWING LIMIT'-++++++++++++++++++++++++++++++++++++++++|

                        [1].DISPLAY CURRENT MAXIMUM BORROWING LIMIT

                        [2].UPDATE BORROWING LIMIT

                        [3].BACK TO ADMIN MENU {<<-}

        =========================================================================================

                Choose an option:
```

Using option 1, Librarians can quickly view the Current maximum borrowing limit. If changes are required,(option 2) the update feature allows for easy adjustments, in maximum borrowing limit.

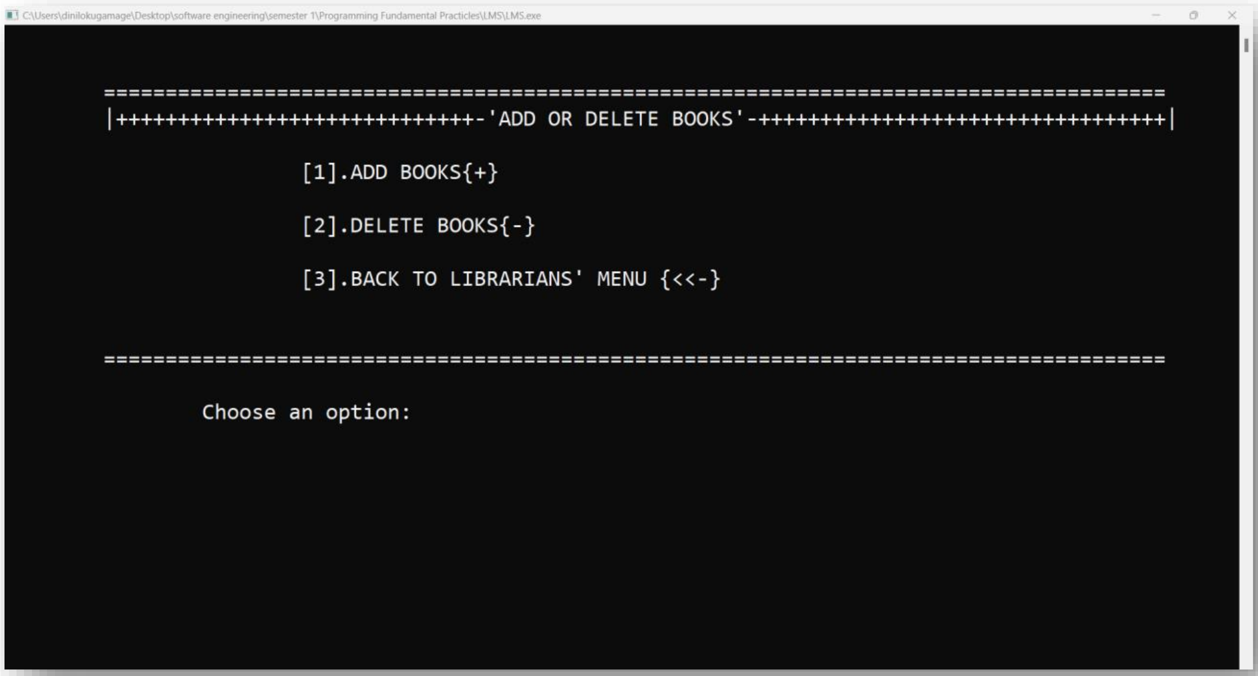
(5) <u>Librarian menu</u>



```
C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practicles\LMS\LMS.exe

         ++++++++++++++++++++++++++++++-LIBRARIANS' MENU-++++++++++++++++++++++++++++++++++
        |                                                                                  |
        |                       [1]-ADD OR DELETE BOOKS                                    |
        |                       [2]-SEARCH OR UPDATE BOOKS                                 |
        |                       [3]-BOOK CHECKOUT OR RETURN BOOKS [ISSUE/ACCEPT]           |
        |                       [4]-ADD OR DELETE BORROWERS                                |
        |                       [5]-SEARCH OR UPDATE BORROWERS                             |
        |                       [6]-BACK TO MAIN MENU                                      |
        |__________________________________________________________________________________|

                Choose an option:
```

As the heart of our library system, the "Librarians' Menu" simplifies essential responsibilities for librarians. The main objective is to enhance book management by making it simple to add, remove, or modify book data. This menu facilitates the borrowing and returning of books and enables quick transactions. Also, Librarians can easily add, delete or update borrower information, assuring they always have the most current information.

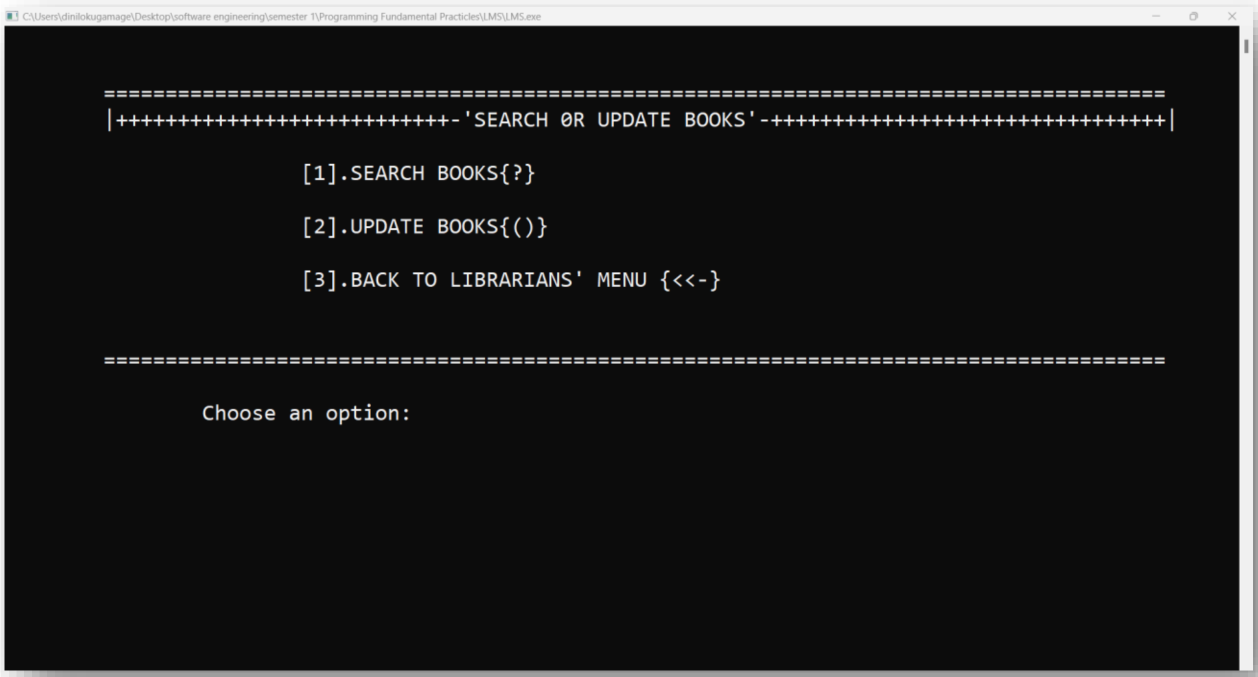
(6) <u>Add or delete book menu</u>



```
C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practicles\LMS\LMS.exe

        ========================================================================================
        |++++++++++++++++++++++++++++++-'ADD OR DELETE BOOKS'-++++++++++++++++++++++++++++++++++|

                        [1].ADD BOOKS{+}

                        [2].DELETE BOOKS{-}

                        [3].BACK TO LIBRARIANS' MENU {<<-}


        ========================================================================================

                Choose an option:
```

In the "Add or Delete Books" module, which was developed for simple usage for librarians. The system makes sure that proper ISBN entries are made when adding to prevent duplication. The librarian can add title, user genre, price, publisher, availability. The title or ID of the book is used for when deleting a book.

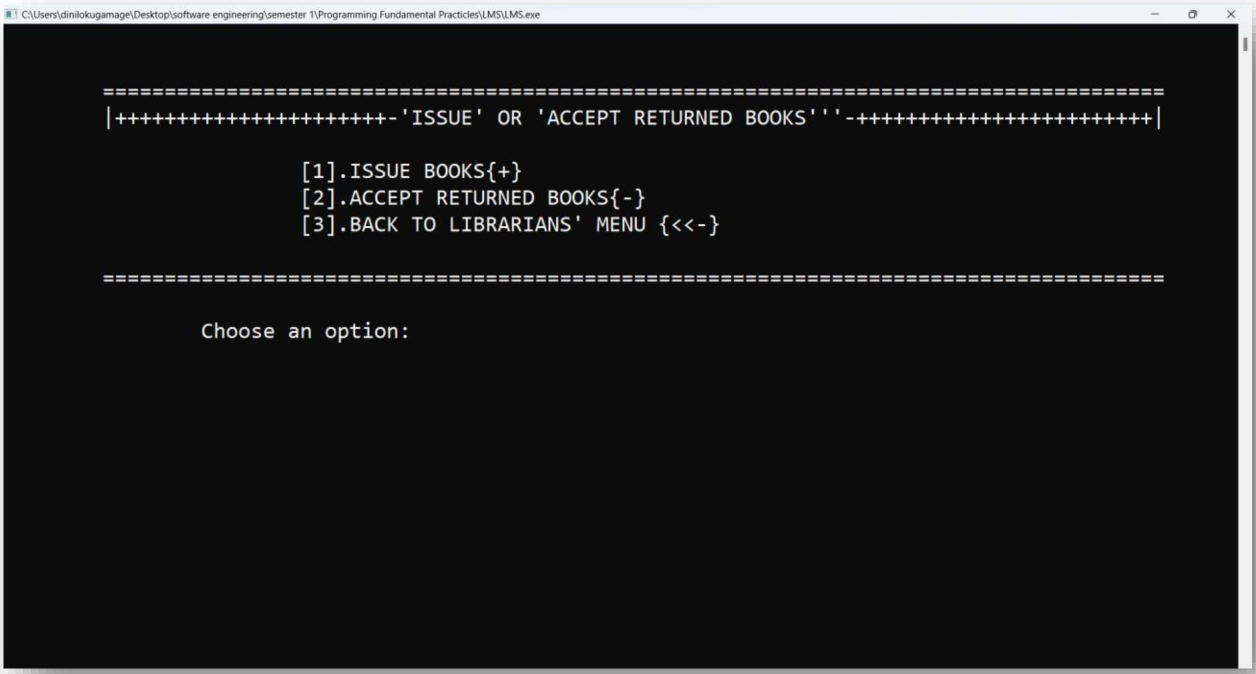
(7) <u>Search or update books menu</u>



```
C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practicles\LMS\LMS.exe

======================================================================================
|+++++++++++++++++++++++++++-'SEARCH 0R UPDATE BOOKS'-++++++++++++++++++++++++++++++++|

                [1].SEARCH BOOKS{?}

                [2].UPDATE BOOKS{()}

                [3].BACK TO LIBRARIANS' MENU {<<-}


======================================================================================

        Choose an option:
```

Librarians can search for books using a variety of criteria, such as ID, title, or author, availability, publisher, or the book entered librarians Id with the results being categorized. Book information, such as the genre, title, availability, author may be updated and modified, as necessary. whether the requested book is found or not, a positive user experience is guaranteed by the built-in error handling.

(8) <u>Issue or accept return books menu</u>

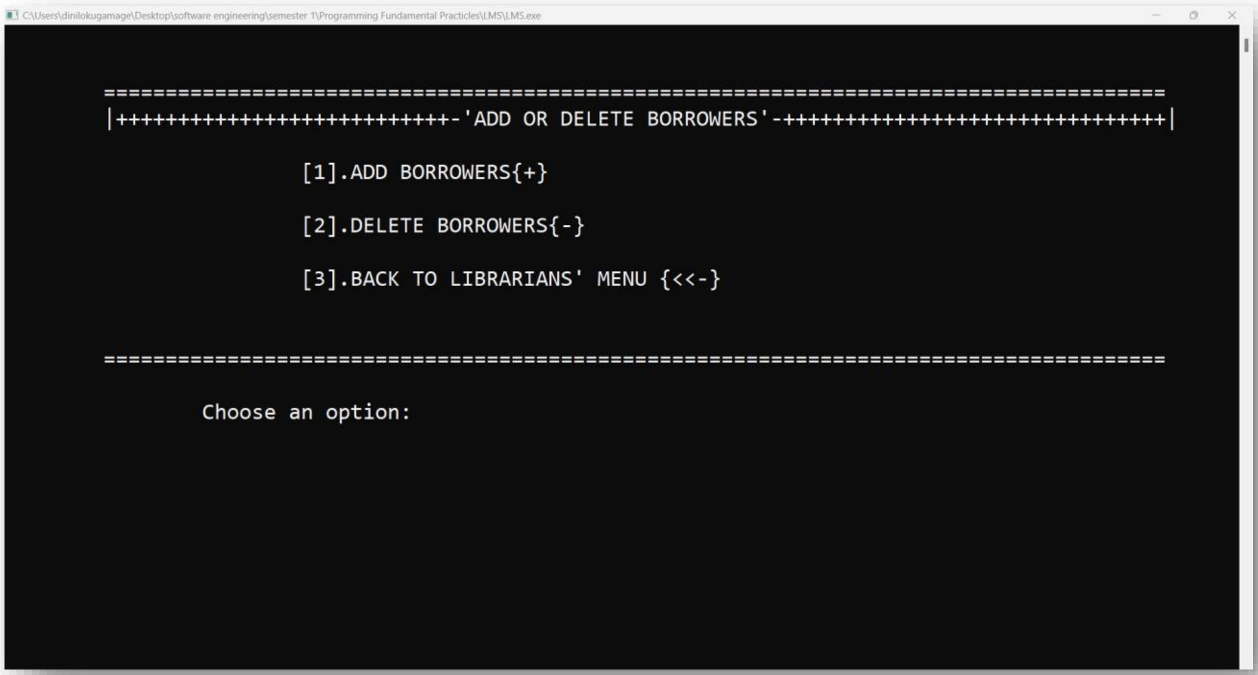


C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practicles\LMS\LMS.exe

```
======================================================================================
|++++++++++++++++++++++-'ISSUE' OR 'ACCEPT RETURNED BOOKS'''-++++++++++++++++++++++++|

                [1].ISSUE BOOKS{+}
                [2].ACCEPT RETURNED BOOKS{-}
                [3].BACK TO LIBRARIANS' MENU {<<-}

======================================================================================

        Choose an option:
```

first option, issue Books manages the lending process by keeping track of details on the book, the borrower, the issue date, and the due date as well as updating the statuses of the book and the borrower. The second option handles return and determines possible penalties for submissions that are delivered after the due date based on the interval between the due and return dates.

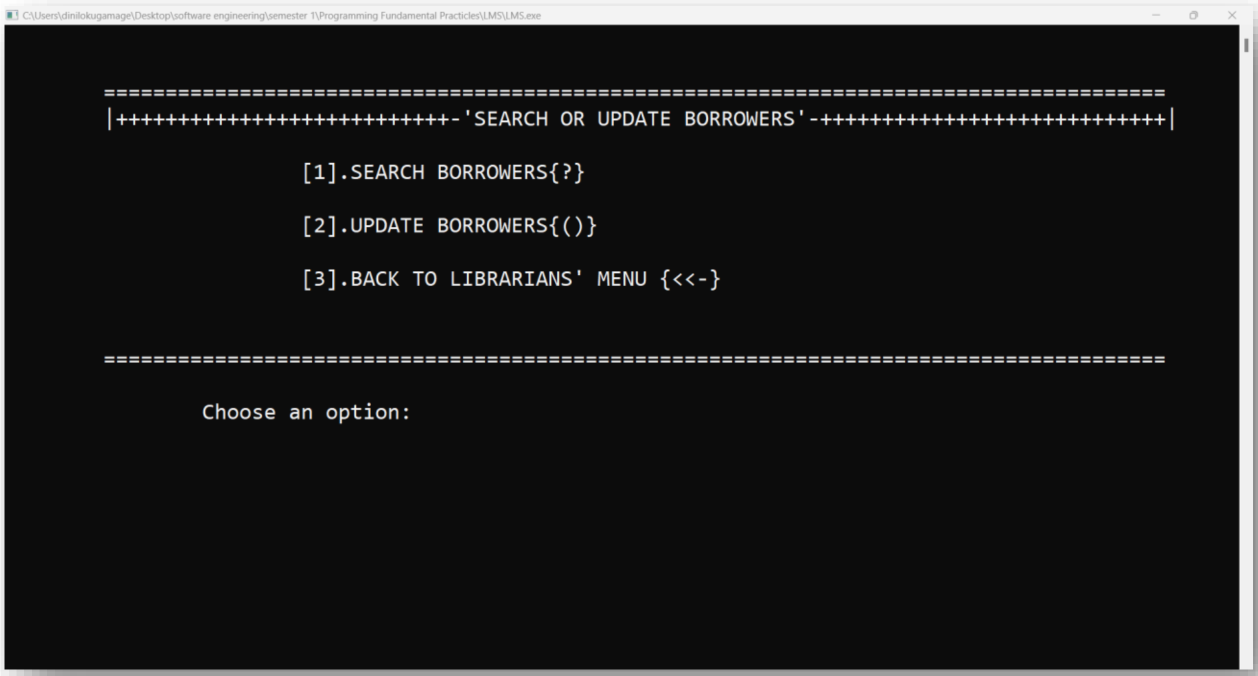
(9) Add or delete borrowers menu



```
C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practicles\LMS\LMS.exe

        ======================================================================================
        |+++++++++++++++++++++++++++-'ADD OR DELETE BORROWERS'-+++++++++++++++++++++++++++++++|

                        [1].ADD BORROWERS{+}

                        [2].DELETE BORROWERS{-}

                        [3].BACK TO LIBRARIANS' MENU {<<-}


        ======================================================================================

                Choose an option:
```

Using the NIC (National Identity Card) number, a librarian can enter information on a possible borrower using the first option, add Borrowers. If the system cannot find a borrower matching the supplied NIC, it prompts the librarian to provide the borrower's name, phone number, email address, and address. The feature switches back to the librarian's menu if a borrower with the same NIC already exists. By using either their ID number or name, the second option makes it easier to delete borrowers.

(10) Search or update borrowers menu



```
C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practicles\LMS\LMS.exe

        ======================================================================================
        |+++++++++++++++++++++++++++-'SEARCH OR UPDATE BORROWERS'-++++++++++++++++++++++++++++|

                        [1].SEARCH BORROWERS{?}

                        [2].UPDATE BORROWERS{()}

                        [3].BACK TO LIBRARIANS' MENU {<<-}

        ======================================================================================

                Choose an option:
```

Using the first option, Librarians can search for borrowers using multiple criteria, including borrower ID, name, phone number, email, address, or even the ID of the librarian who entered the borrower's details. Important borrower details such as their name, contact information can be easily updated or modified using second option, ensuring records are always current and accurately .

3.Test plan

Library Management System	
Test Plan ID	LMS01
Brief introduction about the system	The Library Management System (LMS) is designed to manage the day-to-day operations of a library. It provides different roles for admins and librarians to facilitate system administration and library operation.
Test Objectives	• Ensure all functionalities for Admin and Librarian roles work as intended. • Ensure proper data validations and error-handling mechanisms are in place.
Features to be tested	• Sign-up librarians. • Modify find rate. • Modify borrowing limit. • Add books. • Delete Books. • Search Books. • Update Books. • Issue Books. • Return Books. • Add Borrowers. • Delete Borrowers. • Search Borrowers. • Update Borrowers.
Test Environment	Dev C++
Test Approach	Black Box
Testing Tasks	• Test planning • Test design • Test development • Test execution • Test evaluation
Test Deliverables	• Test plan • Test environment • Test summary • Test result • Test evaluation report
Schedule	Date: - 8/19/2023 Time: - 7.30 am

4. Test Cases

Case 1

Test case	
Test Unit: Sign-up librarians	**Tester**: Dinithi
Test Case ID: 01	**Test Type**: Black box
Test Description: Sign-up new librarians to the system	**Test Execution Date**: 8/09/2023
Title: Add librarians	**Test Execution Time**: 12.45 pm

Step No	Test Step	Test Case ID	Test Input	Expected Result	Actual Result	Test Result (Pass/ Fail)
01	Enter a valid NIC number with 9 or 12 digits.	01	NIC: 992872772 or NIC: 200351501449	The functions proceed without errors.	The functions proceed without errors.	Pass
02	Enter a valid NIC number	01	NIC:20034354	Show "INVALID NIC NUMBER!"	Show "INVALID NIC NUMBER!"	pass
03	Generated librarian ID already exists in the system.	01	Generated librarian id (According to entered correct NIC)	Show "Librarian already exist in the system."	Show "Librarian already exist in the system."	Pass
04	Generated librarian ID does not exist in the system.	01	Generated librarian id (According to entered incorrect NIC)	Show "Librarian does not exist in the system."	Show "Librarian does not exist in the system."	Pass

Case 2

Test case	
Test Unit: Change fine rate	**Tester**: Dinithi
Test Case ID: 02	**Test Type**: Black box
Test Description: Display the current fine rate and allow to modify it	**Test Execution Date**: 8/19/2023
Title: Modify the fine rate	**Test Execution Time**: 12.55 pm

Step No	Test Step	Test Case ID	Test Input	Expected Result	Actual Result	Test Result (Pass/Fail)
01	Fine value > 0.	02	Fine rate: 10	Show "FINE CHANGED SUCCESSFULLY."	Show "FINE CHANGED SUCCESSFULLY."	Pass
02	Fine value < 0	02	Fine rate: -10	Show "INVALID FINE RATE."	Show "INVALID FINE RATE."	Pass

Case 3

Test case	
Test Unit: Change borrowing rate	**Tester**: Dinithi
Test Case ID:	**Test Type**: Black box
Test Description: Display the borrowing limit and allow to modify it	**Test Execution Date**: 8/19/2023
Title: Modify the borrowing rate	**Test Execution Time**: 01:05 pm

Step No	Test Step	Test Case ID	Test Input	Expected Result	Actual Result	Test Result (Pass/Fail)
01	Borrowing limit => 1.	03	Borrowing limit: 5	Show "BORROWING LIMIT CHANGED SUCCESSFULLY."	Show "BORROWING LIMIT CHANGED SUCCESSFULLY."	Pass
02	Borrowing limit < 1.	03	Borrowing limit: -5	Show "INVALID BORROWING LIMIT."	Show "INVALID BORROWING LIMIT."	Pass

Case 4

Test case	
Test Unit: Enter book details	**Tester**: Dinithi
Test Case ID: 04	**Test Type**: Black box
Test Description: Add books to the system	**Test Execution Date**: 8/19/2023
Title: Add books	**Test Execution Time**: 01.15 pm

Step No	Test Step	Test Case ID	Test Input	Expected Result	Actual Result	Test Result (Pass/Fail)
01	Enter a valid ISBN that equals to 10 or 13 digits.	04	ISBN: 1234567891 Or 1234323456789	The functions proceed without errors.	The functions proceed without errors.	Pass
02	Enter an invalid ISBN.	04	ISBN: 2442424	Show "Invalid ISBN Number."	Show "Invalid ISBN Number."	Pass
03	Generate a book ID, which already exists in the system.	04	Generated librarian id (According to entered correct ISBN already exist one)	Show "THIS BOOK ALREADY EXIST."	Show "THIS BOOK ALREADY EXIST."	Pass
04	Generate a book ID, which does not exist in the system.	04	Generated librarian id (entered incorrect ISBN already not exist one)	Show "THIS BOOK DOES NOT EXIST."	Show "THIS BOOK DOES NOT EXIST."	Pass
05	Enter book details.	04	Title: mal mama, author: kamal, publisher: sarasavi, genre: kids price:300	Show "THE BOOK SUCCESSFULLY ADDED."	Show "THE BOOK SUCCESSFULLY ADDED."	Pass

Case 5

Test case	
Test Unit: Enter book details	**Tester**: Dinithi
Test Case ID: 05	**Test Type**: Black box
Test Description: Delete books from the system	**Test Execution Date**: 8/19/2023
Title: Delete books	**Test Execution Time**: 1.25 pm

Step No	Test Step	Test Case ID	Test Input	Expected Result	Actual Result	Test Result (Pass/Fail)
01	Enter valid book id or title.	05	Book ID - LBOOK#1234567891, title – mal mama	Show “THE BOOK DELETED SUCCESSFULLY.”	Show “THE BOOK DELETED SUCCESSFULLY.”	Pass
02	Enter an invalid book id or title.	05	Book ID - LBOOK200351504, title – mal ma	Show “BOOK NOT FOUND.”	Show “BOOK NOT FOUND.”	Pass

Case 6

Test case	
Test Unit: Enter books details	**Tester**: Dinithi
Test Case ID: 06	**Test Type**: Black box
Test Description: Search books that are registered in the system using various criteria	**Test Execution Date**: 8/19/2023
Title: Search books	**Test Execution Time**: 1.35 pm

Step No	Test Step	Test Case ID	Test Input	Expected Result	Actual Result	Test Result (Pass/Fail)
01	Enter valid book id or title, author, publisher, genre, availability, entered librarian.	06	Book ID: LBOOK#1233456789 1/ title: Mal mama/ publisher: Sarasavi/ genre: Kids/ Price: 300/ availability : available	Show book details.	Show book details.	Pass
02	Enter an invalid book id or title, author, publisher, genre, availability, entered librarian.	06	Book ID: LBOOK200351/ title: mal mam/ publisher: dsdds/ genre: ggg/ availability : none	Show “THE BOOK IS NOT FOUND.”	Show “THE BOOK IS NOT FOUND.”	Pass

Case 7

Test case	
Test Unit: Enter books details	**Tester**: Dinithi
Test Case ID: 07	**Test Type**: Black box
Test Description: Update books that are registered in the system according to given criteria	**Test Execution Date**: 8/19/2023
Title: Update books	**Test Execution Time**: 1.45 pm

Step No	Test Step	Test Case ID	Test Input	Expected Result	Actual Result	Test Result (Pass/Fail)
01	Enter valid book id or title.	07	Book ID - LBOOK#1234567891, title – mal mama	Show “THE BOOK UPDATED SUCCESSFULLY.”	Show “THE BOOK UPDATED SUCCESSFULLY.”	Pass
02	Enter invalid book id or title.	07	Book ID - LBOK#200351, title – mal	Show “BOOK NOT FOUND”	Show “BOOK NOT FOUND”	Pass

Case 8

Test case	
Test Unit: Enter books details	**Tester**: Dinithi
Test Case ID: 08	**Test Type**: Black box
Test Description: Enable the librarians to issue books to the borrowers	**Test Execution Date**: 8/19/2023
Title: Issue books	**Test Execution Time**: 2.05 pm

Step No	Test Step	Test Case ID	Test Input	Expected Result	Actual Result	Test Result (Pass/Fail)
01	Enter valid borrower id and book id, also borrowing limit.	08	Book ID: LBOOK#1234567891, Borrower ID: LBOR#200351501449, borrowing limit <= 5	Show "BOOK ISSUED SUCCESSFULLY."	Show "BOOK ISSUED SUCCESSFULLY."	Pass
02	Enter invalid book id.	08	Book ID: LBOK#035150	Show "YOU CAN'T BORROW."	Show "YOU CAN'T BORROW."	Pass
03	Enter invalid borrower id.	08	Borrower ID - LIBB#35150	Show "BORROWER DOES NOT EXIST"	Show "BORROWER DOES NOT EXIST"	Pass
04	Borrower has value greater than borrowing limit	08	Borrowing limit > 5	Show "BORROWER EXCEDED THE MAXIMUM BORROWING LIMIT."	Show "BORROWER EXCEDED THE MAXIMUM BORROWING LIMIT."	Pass

Case 9

Test case	
Test Unit: Enter books details	**Tester**: Dinithi
Test Case ID: 09	**Test Type**: Back box
Test Description: Enable the librarians to return issued books from the borrowers	**Test Execution Date**: 8/19/2023
Title: Return books	**Test Execution Time**: 2:10 pm

Step No	Test Step	Test Case ID	Test Input	Expected Result	Actual Result	Test Result (Pass/Fail)
01	Enter valid borrower id and book id, also borrowing limit.	09	Book ID - LBOOK#1234567891, Borrower ID - LBOR#200351501449, borrowing limit <= 5 and borrowing limit > 5	Show "BOOK ISSUED SUCCESSFULLY."	None	Fail
02	Enter invalid book id.	09	Book ID - LBOK#035150144	Show "YOU CAN'T BORROW"	Show "YOU CAN'T BORROW."	Pass
03	Enter invalid borrower id.	09	Borrower ID - LIBR#200351501449	Show "BORROWER DOES NOT EXIST"	Show "BORROWER DOES NOT EXIST"	Pass

Case 10

Test case	
Test Unit: Enter borrower details	**Tester**: Dinithi
Test Case ID: 10	**Test Type**: Black box
Test Description: Add borrowers to the system	**Test Execution Date**: 8/19/2023
Title: Add borrowers	**Test Execution Time**: 2:15 pm

Step No	Test Step	Test Case ID	Test Input	Expected Result	Actual Result	Test Result (Pass/Fail)
01	Enter a valid NIC that equals to 9 or 12 digits.	10	NIC: 992872772 or NIC:200351501449	The functions proceed without errors.	The functions proceed without errors.	Pass
02	Enter an invalid NIC.	10	NIC - 9938734 or NIC – 2001449	Show “INVALID NIC NUMBER.”	Show “INVALID NIC NUMBER.”	Pass
03	Generate a borrower ID, which already exists in the system.	10	Generated librarian id (Valid already added NIC)	Show “THIS BORROWER ALREADY EXIST.”	Show “THIS BORROWERALREADY EXIST.”	Pass
04	Generate a borrower ID, which does not exist in the system.	10	Generated librarian id (New NIC)	Show “THIS BORROWERDOES NOT EXIST.”	Show “THIS BORROWER DOES NOT EXIST.”	Pass
05	Enter borrower details.	10	Name: Sasindu, Phone Number: 078456524, Email: sasi@gmail.com, address: Piliyandala	Show “THE BORROWER SUCCESSFULLY ADDED.”	Show “THE BORROWER SUCCESSFULLY ADDED.”	Pass

Case 11

Test case	
Test Unit: Enter borrower details	**Tester**: Dinithi
Test Case ID: 11	**Test Type**: Black box
Test Description: Delete borrowers from the system	**Test Execution Date**: 8/19/2023
Title: Delete borrowers	**Test Execution Time**: 2:25 pm

Step No	Test Step	Test Case ID	Test Input	Expected Result	Actual Result	Test Result (Pass/Fail)
01	Enter a valid borrower id or name.	11	Borrower ID:LBOR#992872772, name: Sasindu	Show “THE BORROWER DELETED SUCCESSFULLY.”	Show “THE BORROWER DELETED SUCCESSFULLY.”	Pass
02	Enter an in valid borrower id or name.	11	Borrower ID:LIBB#0351501449, name – upali	Show “BORROWER NOT FOUND.”	Show “BORROWER NOT FOUND.”	Pass

Case 12

Test case	
Test Unit: Enter borrower details	**Tester**: Dinithi
Test Case ID: 12	**Test Type**: Black box
Test Description: Search borrowers that are registered in the system using various criteria	**Test Execution Date**: 8/19/2023
Title: Search borrowers	**Test Execution Time**: 2:30 pm

Step No	Test Step	Test Case ID	Test Input	Expected Result	Actual Result	Test Result (Pass/Fail)
01	Enter valid borrower id or name, phone, address, email, entered librarians.	12	Borrower ID: LBOR#992872772 / name: Sasindu/ Phone: 078456524 / address:Piliyandala/ email: sasi@gmail.com/ entered librarian: LLIB#200351501449.	Show borrower details.	Show borrower details.	Pass
02	Enter an invalid book id or title, author, publisher, genre, availability, entered librarian.	12	Borrower ID:LIBB#35150/ name:gisds/ phone: 078445/ address: sdsd/email:di@gmail.com/ entered librarian:LLIB #3454323	Show “THE BORROWER IS NOT FOUND.”	Show “THE BORROWER IS NOT FOUND.”	Pass

Case 13

Test case	
Test Unit: Enter borrower details	**Tester**: Dinithi
Test Case ID: 13	**Test Type**: Black box
Test Description: Update borrowers that are registered in the system according to given criteria	**Test Execution Date**: 8/19/2023
Title: Update borrowers	**Test Execution Time**: 2:45 pm

Step No	Test Step	Test Case ID	Test Input	Expected Result	Actual Result	Test Result (Pass/Fail)
01	Enter valid borrower id or name.	13	Borrower ID - LBOR##992872772, Name:kavindu	Show “THE BORROWER UPDATED SUCCESSFULLY.”	Show “THE BORROWER UPDATED SUCCESSFULLY.”	Pass
02	Enter valid borrower id or name.	13	Borrower ID - LIBB#251501449,	Show “ BORROWER NOT FOUND”	Show “BORROWER NOT FOUND””	Pass

5. Data validation and Error handling

(1) Data validation

1. Input Range Validity:

The system ensures that specific inputs match the expected character length to maintain consistency and data accuracy.

Librarian Signup and borrowers:

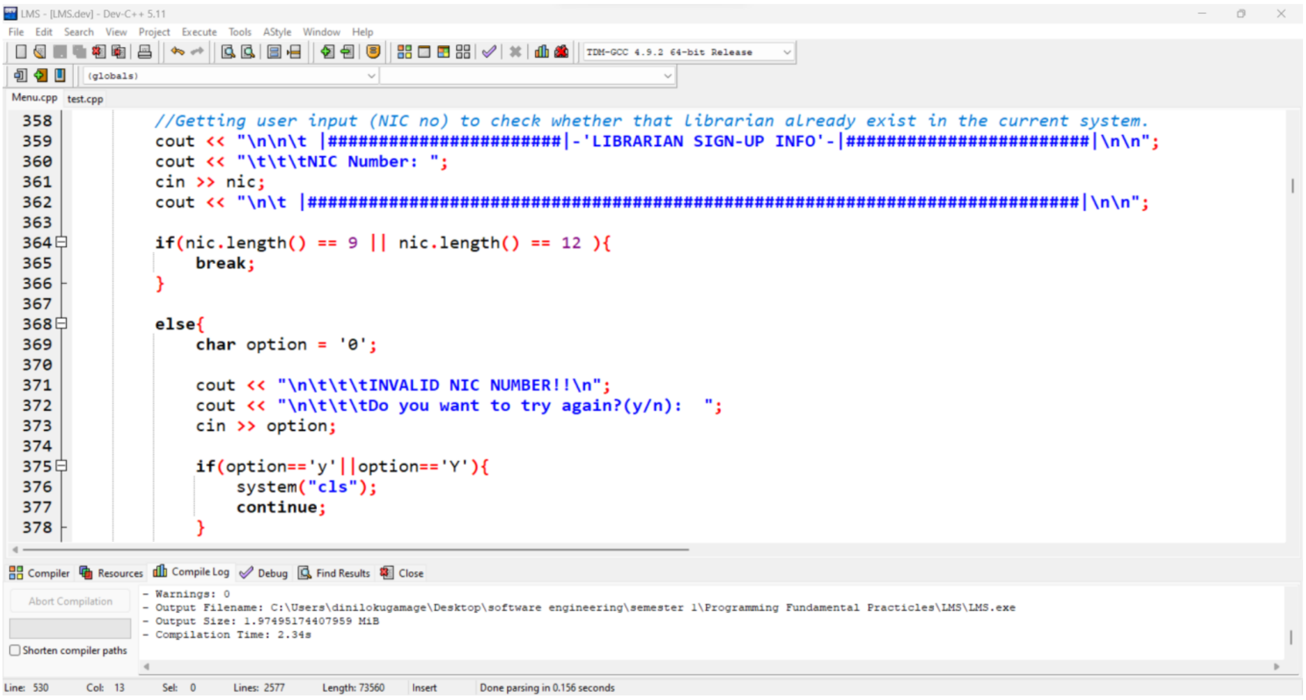
During the registration process for librarians/borrowers, the National Identity Card (NIC) number is used to create a unique librarian ID/borrower ID. The system validates that the entered NIC has either 9 or 12 characters, in line with standard NIC formats.

Book Addition:

When adding a book to the library's inventory, the system uses the International Standard Book Number (ISBN) as a unique identifier. It checks whether the entered ISBN has either 10 or 13 characters, which are standard ISBN formats.

Fine Rate, Modify Borrowing limit:

checked whether they are positive numbers or not.



```
        //Getting user input (NIC no) to check whether that librarian already exist in the current system.
        cout << "\n\n\t |########################|-'LIBRARIAN SIGN-UP INFO'-|########################|\n\n";
        cout << "\t\t\tNIC Number: ";
        cin >> nic;
        cout << "\n\t |#############################################################################|\n\n";

        if(nic.length() == 9 || nic.length() == 12 ){
            break;
        }

        else{
            char option = '0';

            cout << "\n\t\t\tINVALID NIC NUMBER!!\n";
            cout << "\n\t\t\tDo you want to try again?(y/n):  ";
            cin >> option;

            if(option=='y'||option=='Y'){
                system("cls");
                continue;
            }
```

```
- Warnings: 0
- Output Filename: C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practicles\LMS\LMS.exe
- Output Size: 1.97495174407959 MiB
- Compilation Time: 2.34s
```

LMS - [LMS.dev] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Menu.cpp test.cpp
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Line: 530 Col: 13 Sel: 0 Lines: 2577 Length: 73560 Insert Done parsing in 0.156 seconds

2. Case Sensitivity and Data Matching:

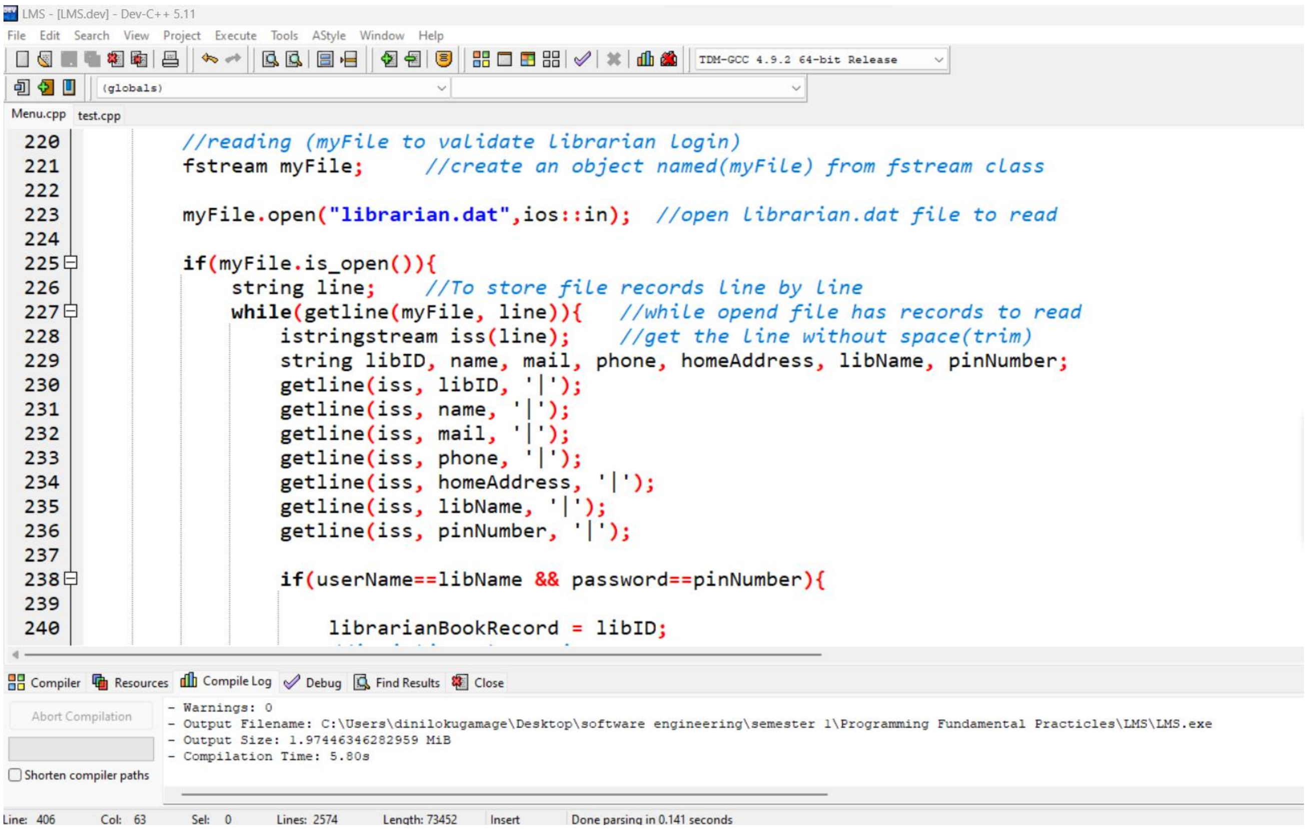
a. Security validation

Admin Login:

The system cross-verifies the entered username and password with hardcoded credentials. Input is processed to ensure case-sensitive matching, safeguarding the system from unauthorized access.

Librarian Login:

The login procedure for librarians verifies that the username and password supplied match the credentials already recorded in the file. The system's security is improved as a result of the reduction of unauthorized logins.


LMS - [LMS.dev] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Menu.cpp test.cpp
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths

```
    //reading (myFile to validate librarian login)
    fstream myFile;     //create an object named(myFile) from fstream class

    myFile.open("librarian.dat",ios::in);  //open librarian.dat file to read

    if(myFile.is_open()){
        string line;    //To store file records line by line
        while(getline(myFile, line)){   //while opend file has records to read
            istringstream iss(line);    //get the line without space(trim)
            string libID, name, mail, phone, homeAddress, libName, pinNumber;
            getline(iss, libID, '|');
            getline(iss, name, '|');
            getline(iss, mail, '|');
            getline(iss, phone, '|');
            getline(iss, homeAddress, '|');
            getline(iss, libName, '|');
            getline(iss, pinNumber, '|');

            if(userName==libName && password==pinNumber){

                librarianBookRecord = libID;
```

```
- Warnings: 0
- Output Filename: C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practicles\LMS\LMS.exe
- Output Size: 1.97446346282959 MiB
- Compilation Time: 5.80s
```

```
Line: 406    Col: 63    Sel: 0    Lines: 2574    Length: 73452    Insert    Done parsing in 0.141 seconds
```

b. Validation for verifying data duplication and accuracy

Sign up Librarians:

During the librarian signup process, the system verifies if there's an existing librarian with the same NIC number. This prevents the duplication of entries.

Add books or borrowers:

Similarly, when adding books or borrowers, their unique IDs are cross-referenced with the system's records. Only if no match is found, the system proceeds with the addition.

Issue or Accept return books:

The process of issuing or returning books involves validation checks to ensure both the book and borrower IDs exist in their respective files. Only upon successful verification does the system allow the transaction.

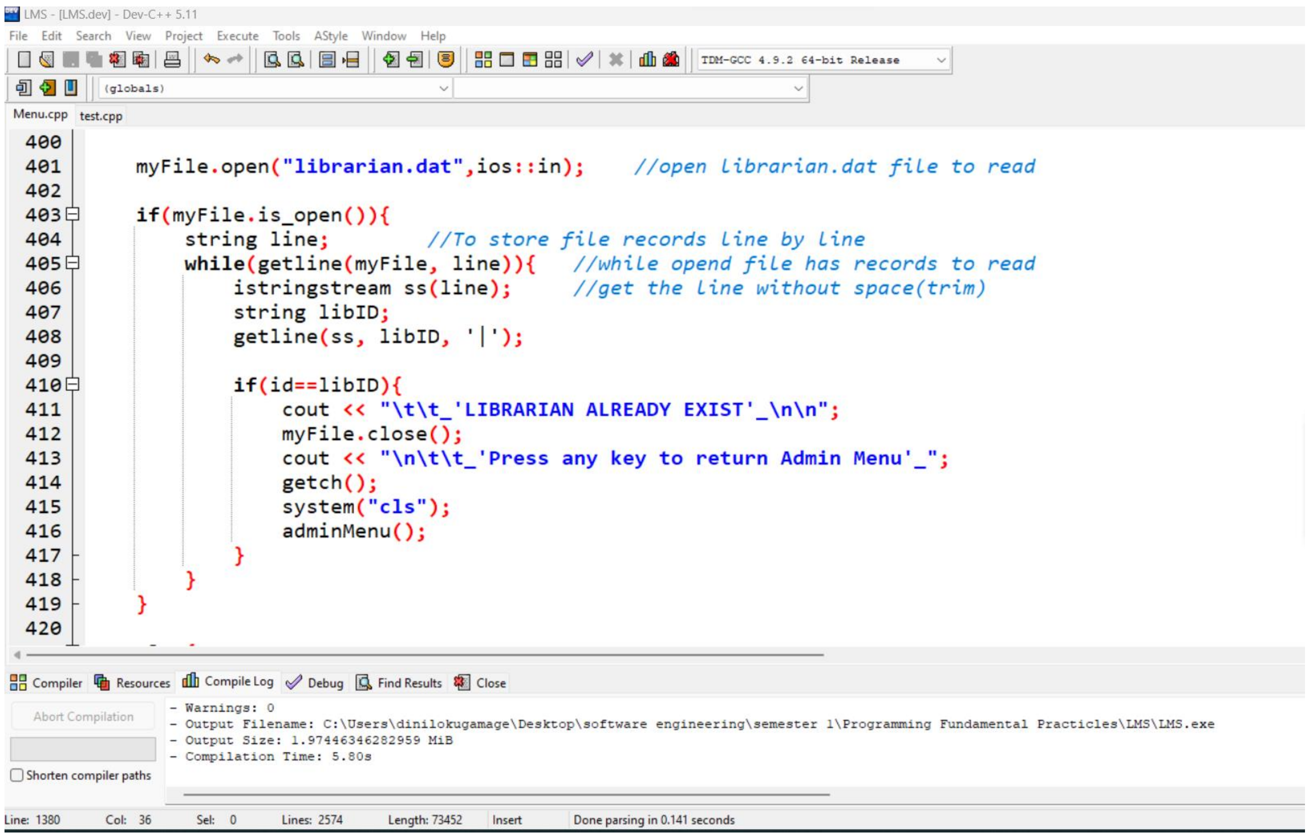
Delete books or borrowers:

For deleting records, the system first validates the existence of the book or borrower in the relevant files. Only genuine records can be removed, ensuring that invalid deletions are minimized.

update and search books or borrowers:

The updating mechanism follows a similar structure. Before modifications are made to books or borrower records, a check is executed to validate their existence in the system.

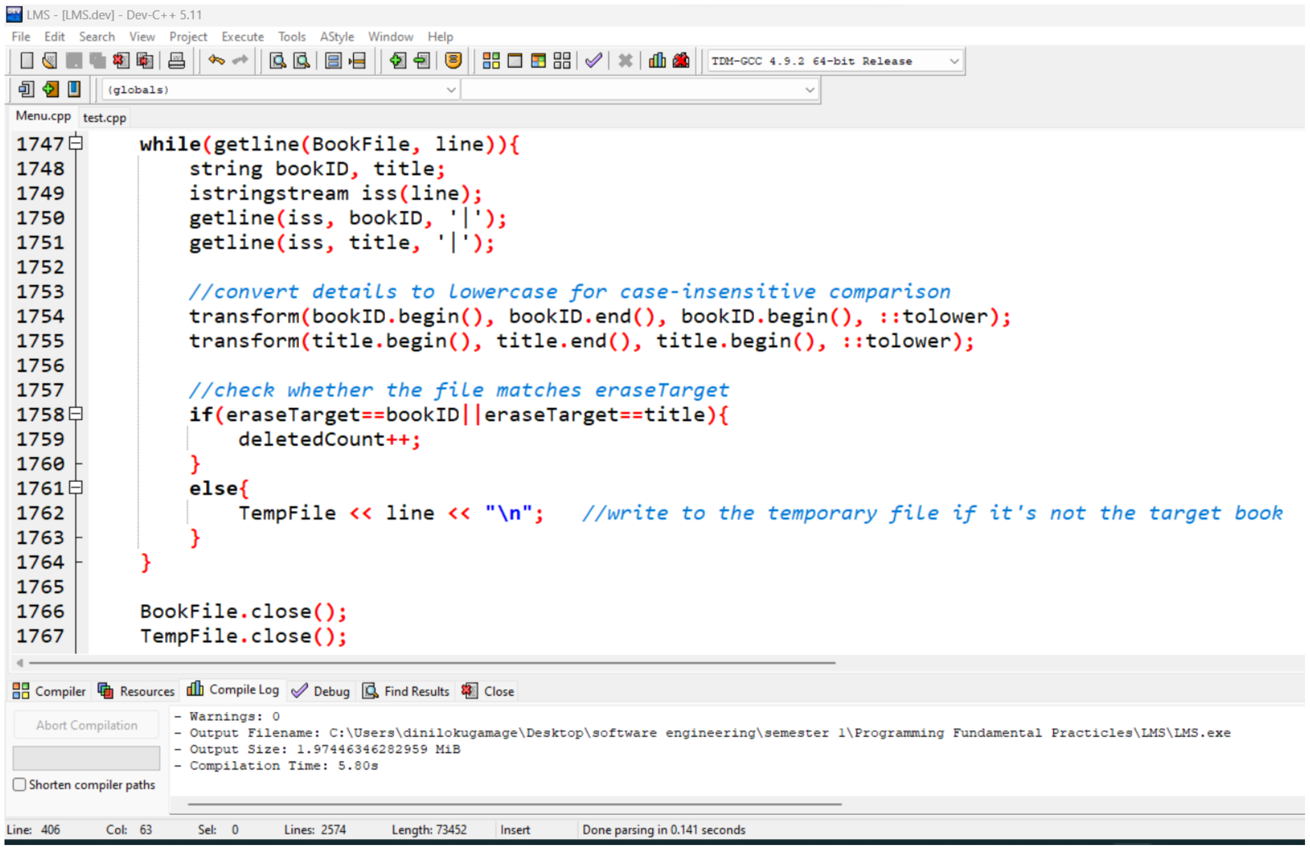
The transform () function of the system helps in these validation efforts by allowing flexible data matching and maintaining a balance between user convenience and data integrity.


LMS - [LMS.dev] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Menu.cpp test.cpp
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths

```
    myFile.open("librarian.dat",ios::in);     //open librarian.dat file to read

    if(myFile.is_open()){
        string line;         //To store file records line by line
        while(getline(myFile, line)){    //while opend file has records to read
            istringstream ss(line);     //get the line without space(trim)
            string libID;
            getline(ss, libID, '|');

            if(id==libID){
                cout << "\t\t_'LIBRARIAN ALREADY EXIST'_\n\n";
                myFile.close();
                cout << "\n\t\t_'Press any key to return Admin Menu'_";
                getch();
                system("cls");
                adminMenu();
            }
        }
    }
```

```
- Warnings: 0
- Output Filename: C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practicles\LMS\LMS.exe
- Output Size: 1.97446346282959 MiB
- Compilation Time: 5.80s
```

Line: 1380 Col: 36 Sel: 0 Lines: 2574 Length: 73452 Insert Done parsing in 0.141 seconds


LMS - [LMS.dev] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Menu.cpp test.cpp
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths

```
    while(getline(BookFile, line)){
        string bookID, title;
        istringstream iss(line);
        getline(iss, bookID, '|');
        getline(iss, title, '|');

        //convert details to lowercase for case-insensitive comparison
        transform(bookID.begin(), bookID.end(), bookID.begin(), ::tolower);
        transform(title.begin(), title.end(), title.begin(), ::tolower);

        //check whether the file matches eraseTarget
        if(eraseTarget==bookID||eraseTarget==title){
            deletedCount++;
        }
        else{
            TempFile << line << "\n";    //write to the temporary file if it's not the target book
        }
    }

    BookFile.close();
    TempFile.close();
```

```
- Warnings: 0
- Output Filename: C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practicles\LMS\LMS.exe
- Output Size: 1.97446346282959 MiB
- Compilation Time: 5.80s
```

Line: 406 Col: 63 Sel: 0 Lines: 2574 Length: 73452 Insert Done parsing in 0.141 seconds

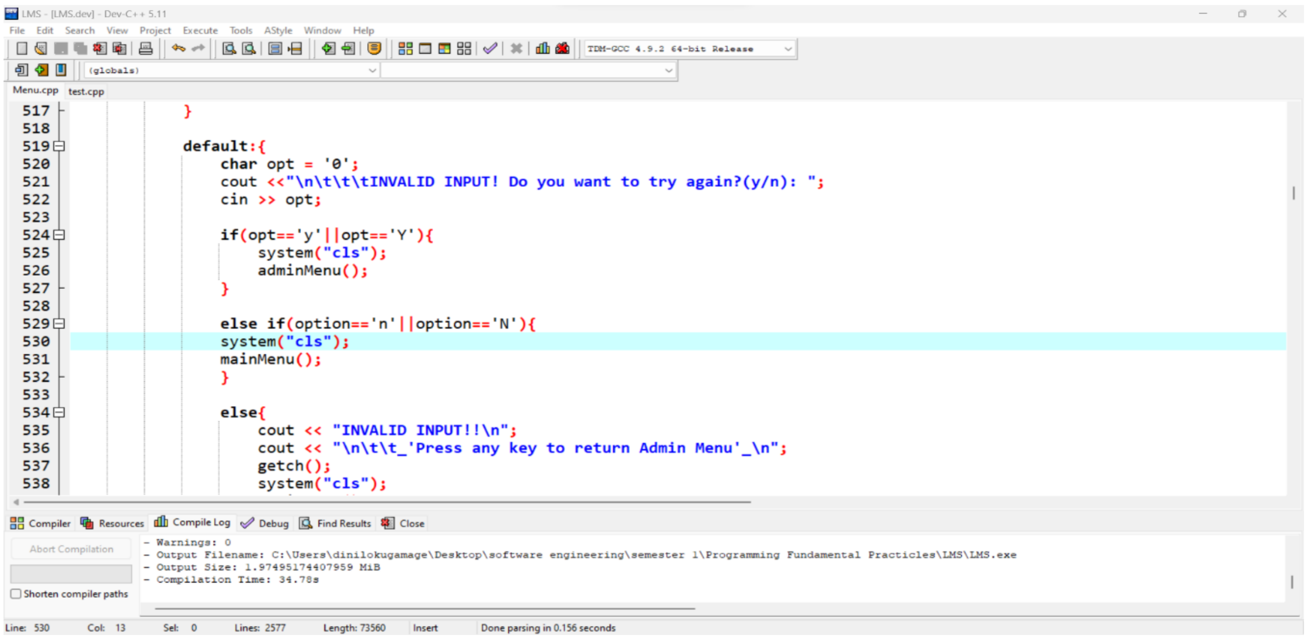
-Delete books

(2) Error handling

The way a system reacts to unexpected issues is frequently used to evaluate a system's durability. In order to establish an error-free environment, the Library Management System was designed with the user in mind. Error management involves more than simply discovering faults.

Navigating the Menus (User Input Verifications):

The system contains an automatic way to evaluate the accuracy of the user's decision. Whether it's the main menu, admin, librarian, or various sub-menus related to books and borrowers, the system constantly checks if the user's choice is within the accepted range. Users are given a clear indication when wrong choices are made: "Invalid input, do you want to try again?" This design strengthens user confidence by not just pointing out the error but also offering an immediate chance for correction.


LMS - [LMS.dev] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Menu.cpp test.cpp
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths

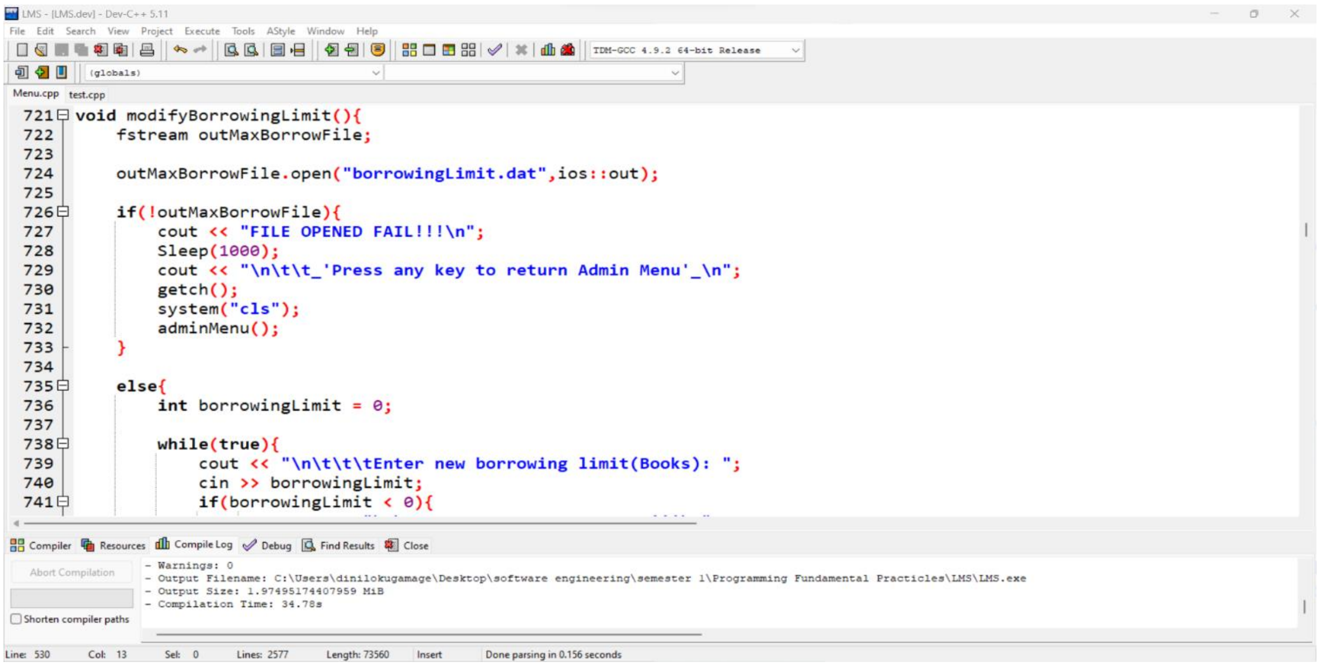
```
            }

            default:{
                char opt = '0';
                cout <<"\n\t\t\tINVALID INPUT! Do you want to try again?(y/n): ";
                cin >> opt;

                if(opt=='y'||opt=='Y'){
                    system("cls");
                    adminMenu();
                }

                else if(option=='n'||option=='N'){
                system("cls");
                mainMenu();
                }

                else{
                    cout << "INVALID INPUT!!\n";
                    cout << "\n\t\t_'Press any key to return Admin Menu'_\n";
                    getch();
                    system("cls");
```

```
- Warnings: 0
- Output Filename: C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practicles\LMS\LMS.exe
- Output Size: 1.97495174407959 MiB
- Compilation Time: 34.78s
```

```
Line: 530    Col: 13    Sel: 0    Lines: 2577    Length: 73560    Insert    Done parsing in 0.156 seconds
```

File Existence Verification:

Checking for file integrity is essential, especially when the system depends on the files for important tasks. The system checks for the availability of relevant files for different features such librarian login, signup, changing borrowing limits, handling fine rates, and actions linked to books and borrowers. Instead of freezing or crashing when a file is missing, the system gently alerts the user with a "File does not exist" message. This quick response guards against any data loss or corruption and gives the user a direct channel to carry on uninterrupted.



```
void modifyBorrowingLimit(){
    fstream outMaxBorrowFile;

    outMaxBorrowFile.open("borrowingLimit.dat",ios::out);

    if(!outMaxBorrowFile){
        cout << "FILE OPENED FAIL!!!\n";
        Sleep(1000);
        cout << "\n\t\t_'Press any key to return Admin Menu'_\n";
        getch();
        system("cls");
        adminMenu();
    }

    else{
        int borrowingLimit = 0;

        while(true){
            cout << "\n\t\t\tEnter new borrowing limit(Books): ";
            cin >> borrowingLimit;
            if(borrowingLimit < 0){
```

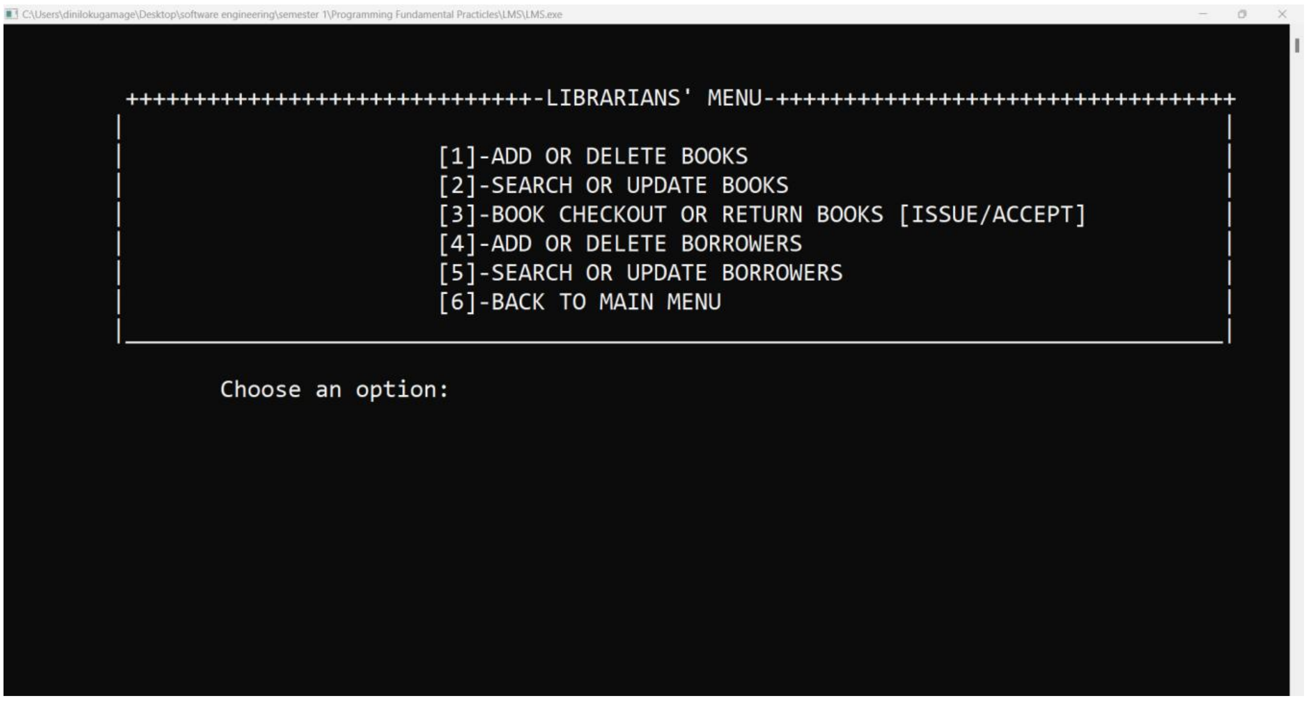
```
- Warnings: 0
- Output Filename: C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practicles\LMS\LMS.exe
- Output Size: 1.97495174407959 MiB
- Compilation Time: 34.78s
```

LMS - [LMS.dev] - Dev-C++ 5.11
Menu.cpp
test.cpp
Line: 530 Col: 13 Sel: 0 Lines: 2577 Length: 73560 Insert Done parsing in 0.156 seconds

<u>Durable System Navigation:</u>

An efficient error-handling approach includes not only the detection of problems but also the implementation of user-friendly means of navigating them.

Retries and System Flow: To prevent users from being left stranded after an error, the system frequently gives users the choice to attempt again or to return to the prior state. This strategy makes sure that users are always aware of their position inside the system and what to do next.ex: every menu has a direction to go ahead and back to previous menu.

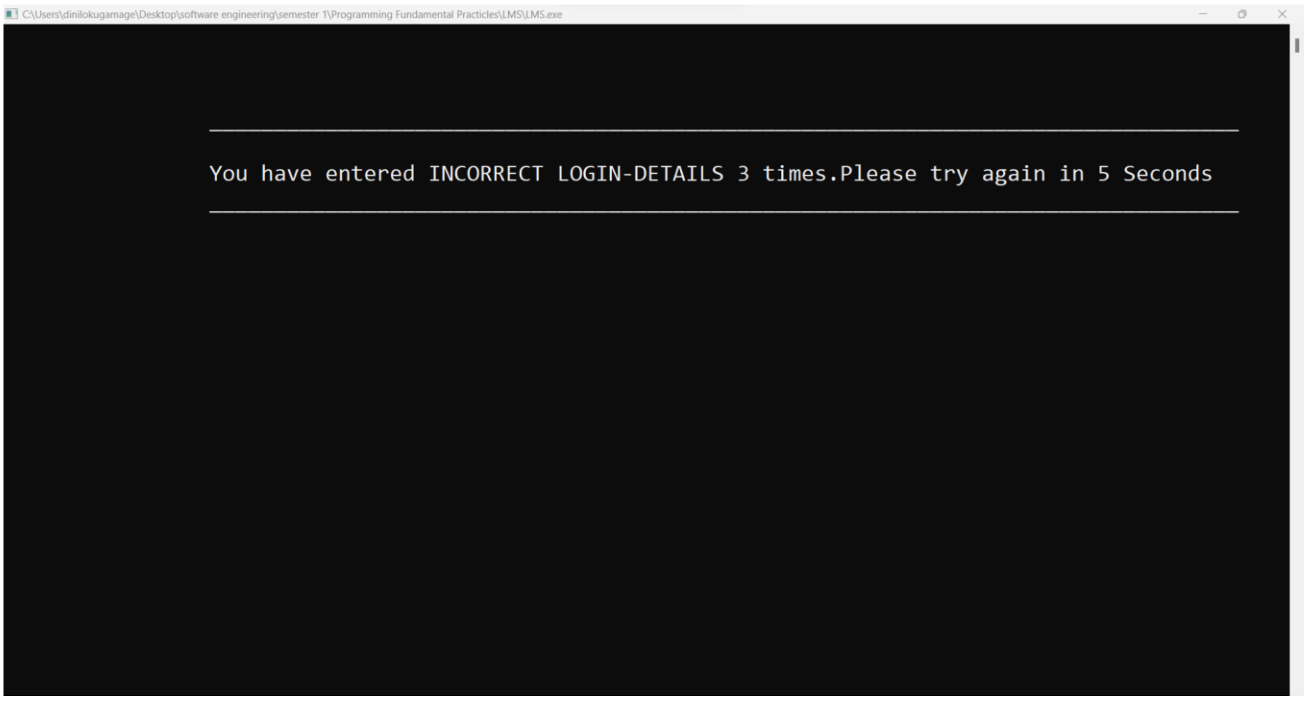


```
+++++++++++++++++++++++++++++++-LIBRARIANS' MENU-+++++++++++++++++++++++++++++++++++
|                                                                                   |
|                        [1]-ADD OR DELETE BOOKS                                    |
|                        [2]-SEARCH OR UPDATE BOOKS                                 |
|                        [3]-BOOK CHECKOUT OR RETURN BOOKS [ISSUE/ACCEPT]           |
|                        [4]-ADD OR DELETE BORROWERS                                |
|                        [5]-SEARCH OR UPDATE BORROWERS                             |
|                        [6]-BACK TO MAIN MENU                                      |
|___________________________________________________________________________________|

       Choose an option:
```

6. Additional features

1. Strengthened Security Measures:

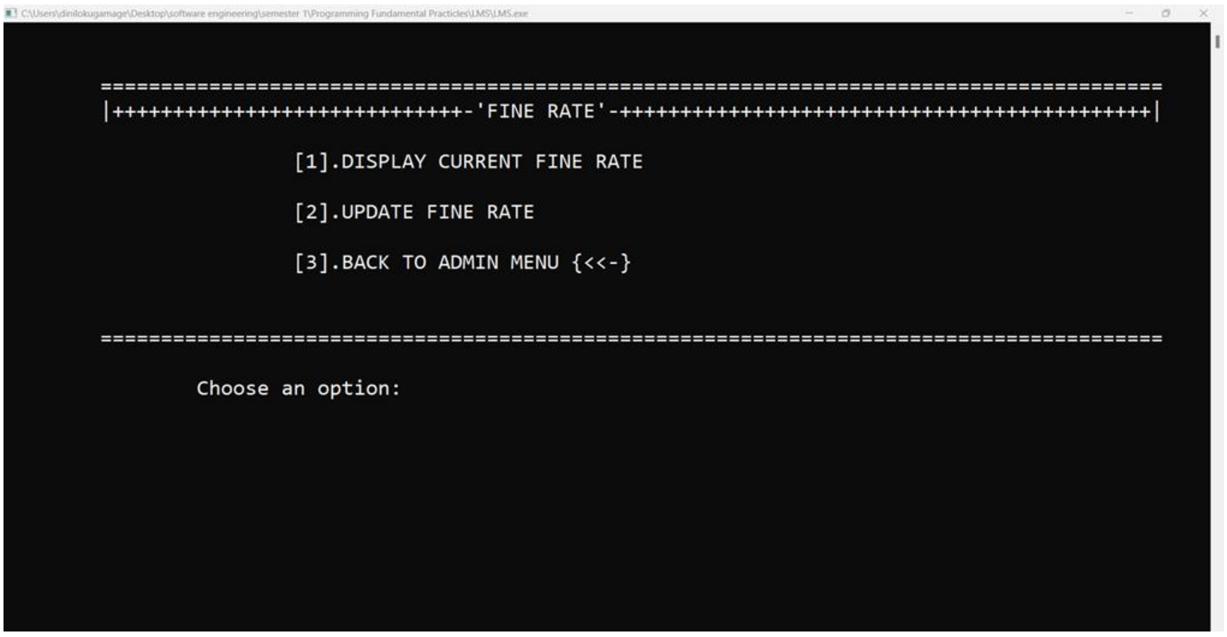
Protecting personal information of users is essential in the current digital era. Our system has been designed with a modern safety function, similar to those seen in present mobile devices. More than 3 inaccurate ID or password entries result in the system placing the user on a temporary hold, preventing access for 5 seconds. This prevents any unwanted access attempts and ensures that only trustworthy users are allowed admission.



```
C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practicles\LMS\LMS.exe

________________________________________________________________________________

You have entered INCORRECT LOGIN-DETAILS 3 times.Please try again in 5 Seconds
________________________________________________________________________________
```

2. Simplified Modifications of the Fine and Borrowing Limit:

flexibility is key to a user-friendly experience. Recognizing this, the system has dedicated menus for both 'Fine Rate' and 'Maximum Borrowing Limit'. The two-tiered approach allows users first to view the current status and then, if necessary, modify it. This clear division ensures a smoother operation and reduces the chances of unintentional errors.

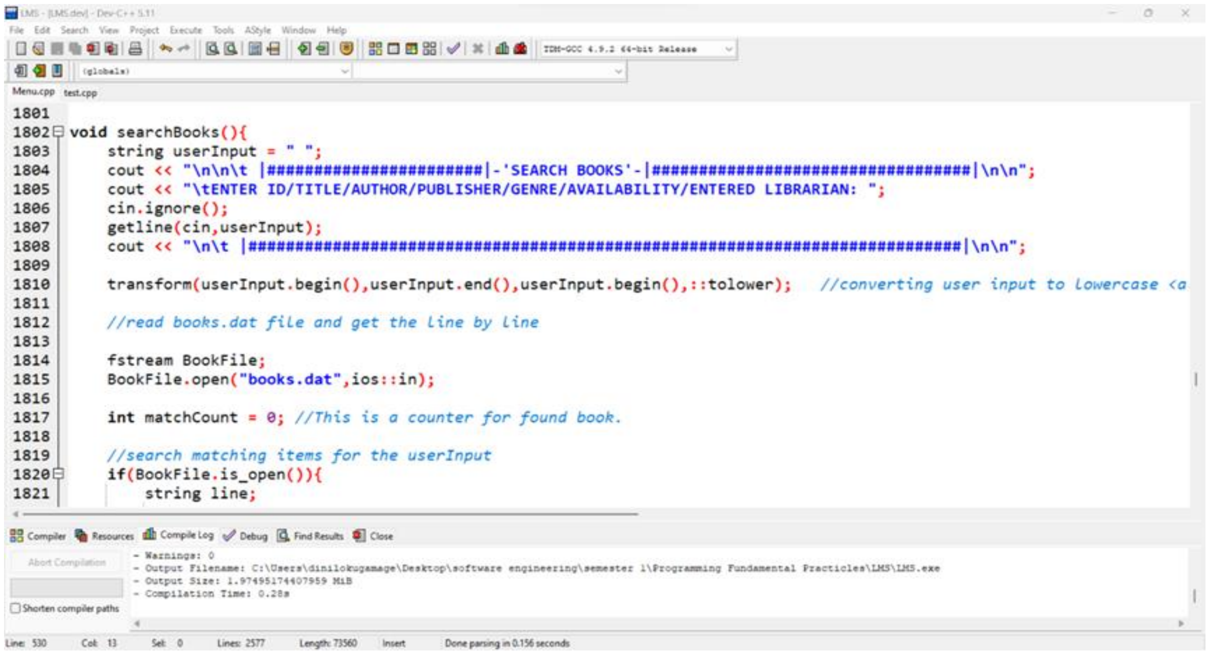


```
C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practicles\LMS\LMS.exe

==========================================================================================
|++++++++++++++++++++++++++++++-'FINE RATE'-++++++++++++++++++++++++++++++++++++++++++++++|

                [1].DISPLAY CURRENT FINE RATE

                [2].UPDATE FINE RATE

                [3].BACK TO ADMIN MENU {<<-}

==========================================================================================

        Choose an option:
```

3. Advanced Book Searching Capabilities and Displaying borrower information for users (when borrowing a book):

A library's efficiency is often judged by how swiftly a user can find a desired book. To ensure this, our system has a comprehensive book search function. Users can now search books based on various criteria like availability status, genre, publisher, and even the librarian who logged the book entry. This granular search capability significantly reduces the time taken to locate specific titles, leading to a more satisfying user experience.

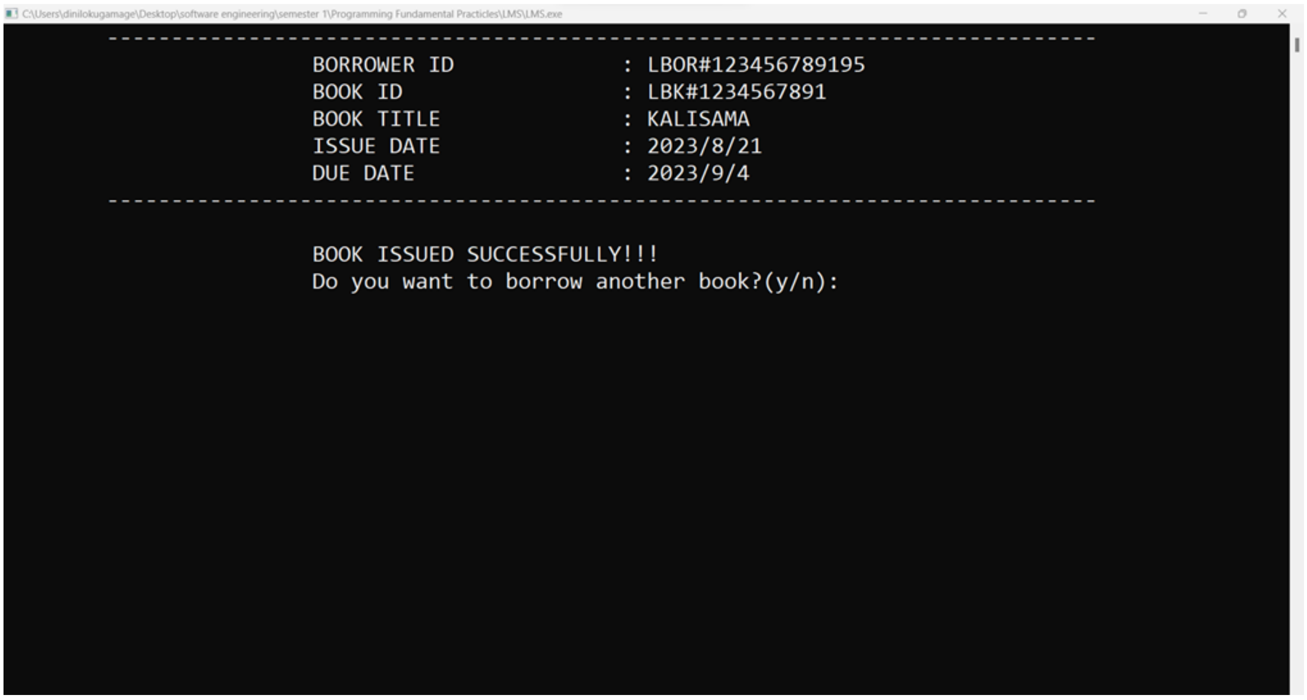
When borrowing a book, at the end after writing to the file system displays borrower record details in the console.



```cpp

void searchBooks(){
    string userInput = " ";
    cout << "\n\n\t |#######################|-'SEARCH BOOKS'-|##################################|\n\n";
    cout << "\tENTER ID/TITLE/AUTHOR/PUBLISHER/GENRE/AVAILABILITY/ENTERED LIBRARIAN: ";
    cin.ignore();
    getline(cin,userInput);
    cout << "\n\t |############################################################################|\n\n";

    transform(userInput.begin(),userInput.end(),userInput.begin(),::tolower);   //converting user input to lowercase <a

    //read books.dat file and get the line by line

    fstream BookFile;
    BookFile.open("books.dat",ios::in);

    int matchCount = 0; //This is a counter for found book.

    //search matching items for the userInput
    if(BookFile.is_open()){
        string line;
```

```
- Warnings: 0
- Output Filename: C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practicles\LMS\LMS.exe
- Output Size: 1.97495174407959 MiB
- Compilation Time: 0.28s
```

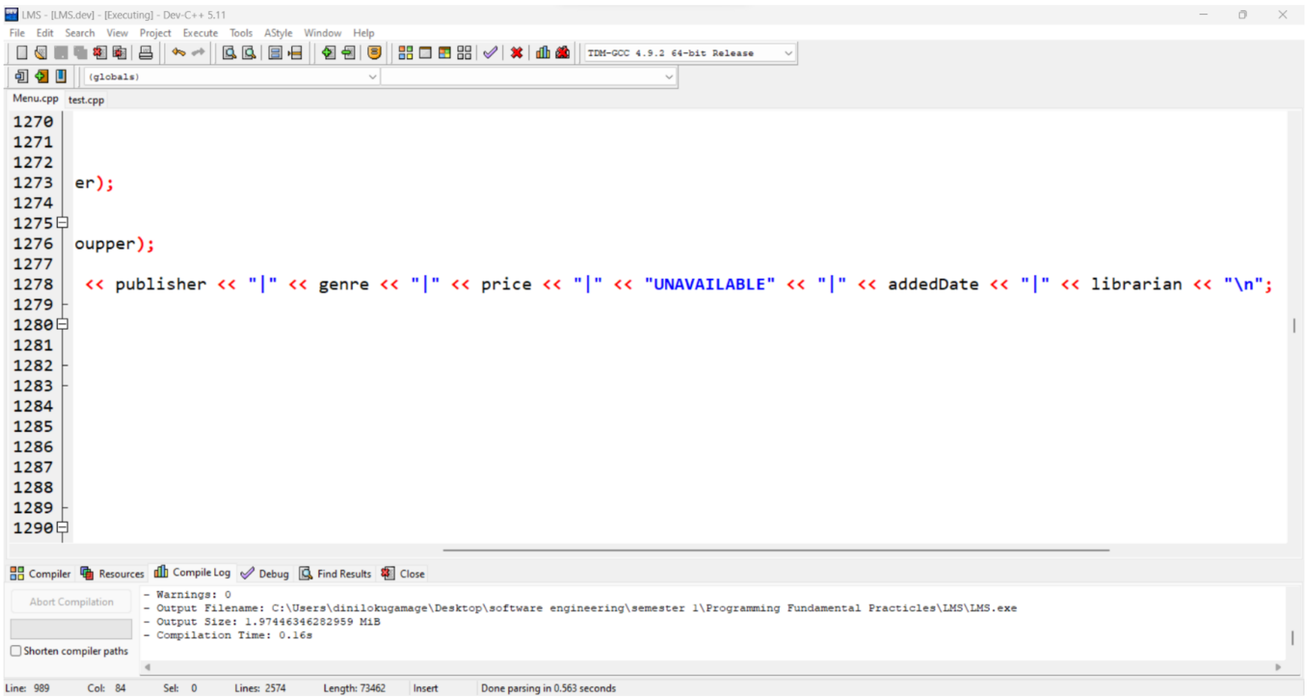
LMS - [LMS.dev] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Menu.cpp test.cpp
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Line: 530 Col: 13 Sel: 0 Lines: 2577 Length: 73560 Insert Done parsing in 0.156 seconds



```
 ------------------------------------------------------------------------------
                BORROWER ID               : LBOR#123456789195
                BOOK ID                   : LBK#1234567891
                BOOK TITLE                : KALISAMA
                ISSUE DATE                : 2023/8/21
                DUE DATE                  : 2023/9/4
 ------------------------------------------------------------------------------

                BOOK ISSUED SUCCESSFULLY!!!
                Do you want to borrow another book?(y/n):
```

4. Accountability and Transparency:

Mistakes do happen, and it's important to identify the source of any errors so that corrective action can be taken. Every time a book or a borrower is entered into the file, the responsible librarian's ID is also recorded. This ensures accountability by providing a transparent audit trail. In the rare instance of a mistake or oversight, the system makes sure that management can quickly pinpoint the problem's source, ensuring fast corrective action.



```
er);

oupper);

 << publisher << "|" << genre << "|" << price << "|" << "UNAVAILABLE" << "|" << addedDate << "|" << librarian << "\n";
```

```
- Warnings: 0
- Output Filename: C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practicles\LMS\LMS.exe
- Output Size: 1.97446346282959 MiB
- Compilation Time: 0.16s
```

LMS - [LMS.dev] - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Menu.cpp test.cpp
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Line: 989 Col: 84 Sel: 0 Lines: 2574 Length: 73462 Insert Done parsing in 0.563 seconds

7. User Documentation

Guidance on Using the Library Management System.

[1] Main Menu:

1.Admin Login:

Choose option 1 to login as an Admin.

Enter your admin credentials (username and password).

If your credentials are correct, you will be directed to the Admin Menu. If not, you'll receive an error and will need to re-enter the correct credentials.

2.Librarian Login:

Choose option 2 to login as a Librarian.

Enter your librarian credentials (username and password).

If your credentials are correct, you'll access the Librarian Menu. If not, you'll receive an error and will need to re-enter your credentials.

3.Back to Main Menu:

Choose option 3 to return to the main menu at any point.

[1.1] Admin menu

1.Sign Up Librarians

2.Update the Fine Rate

3.Modify Maximum Borrowing Limits

4.Back to Main Menu

1. Sign Up Librarians:

How to Access: From the Admin Menu, enter "1" to select the "Sign Up Librarians" option.

Functionality:

*NIC Number Verification:

You'll first be asked to provide the NIC (National Identification Card) Number. The system checks the NIC for its validity (whether it's 9 or 12 characters long). If it's invalid, you'll have the chance to re-enter it.

Existing Librarian Check: The system checks if a librarian with the entered NIC already exists. If the librarian exists, you'll be notified and returned to the Admin Menu.

Librarian Sign Up: If the librarian doesn't exist, you can proceed to sign them up. You'll need to provide the following details:

1)Full Name

2)Email Address

3)Contact Number

4)Residential Address

5)Username

6)Password

These details are saved in a file ("librarian.dat").

2. Update the Fine Rate:

How to Access: From the Admin Menu, enter "2" to select the "Update the Fine Rate" option.

Functionality:

You can either display the current fine rate or update it.

Display Current Fine Rate: Choose the "DISPLAY CURRENT FINE RATE" option to view the existing fine rate.

Update Fine Rate: Choose the "UPDATE FINE RATE" option to set a new fine rate.

3. Modify Maximum Borrowing Limits:

How to Access: From the Admin Menu, enter "3" to select the "Modify Maximum Borrowing Limits" option.

Functionality:

You can either display the current borrowing limit or update it.

Display Current Borrowing Limit: Choose the "DISPLAY CURRENT MAXIMUM BORROWING LIMIT" option to view the existing borrowing limit.

Update Borrowing Limit: Choose the "UPDATE BORROWING LIMIT" option to set a new borrowing limit.

4. Back to Main Menu:

How to Access: From the Admin Menu, enter "4" to return to the Main Menu.

[1.2] Librarian Menu

1.Adding or Deleting Books

2.Searching or Updating Books

3.Book Checkout or Return

4.Adding or Deleting Borrowers

5.Searching or Updating Borrowers

6.Returning to Main Menu

Option [1] - This option allows you to add new books to the system or delete existing ones.

[1.1] Add Books: This will prompt you to enter the details of the new book.

When you select this option, you'll be prompted to enter the essential details of the new book, such as title, author, publication year, and any other pertinent information. After entering the necessary details, the book will be added to the system's File.

[1.2] Delete Books: Here, you can delete books from the system by providing the

specific details of the book you wish to remove. Selecting this option will allow you to remove a book from the system. You’ll typically need to provide a specific detail, such as the book's ID or title, to locate the book you want to delete. Once found, you can proceed with the deletion.

Option [2] - This option lets you search for books or update their details.

[2.1] Search Books: Allows you to search for a book by its ID or name. This will display all the details of the book. By choosing this, you can search for a specific book using its ID, title, or any other relevant criteria. The system will display the book's details, including its status (whether it's available or checked out).

[2.2] Update Books: Enter the ID or name of the book you wish to update, in this option u have a chance to choose what details should be updated. According to the user input then user have a chance to update the new details.

Option [3] - Manages the process of checking out books to borrowers or accepting returned books.

[3.1] Issue Book: Here, you can issue a book to a borrower. You'll need to provide both the book and borrower's IDs The system will then mark the book as "checked out" and note the borrower's details.

When a book is issued, selecting this option allows you to mark it as "unavailable" in the system.

[3.2] Return Book: This option facilitates the return of a book. Similar to the issue option, you'll need the book and borrower's IDs.

When a book is returned, selecting this option allows you to mark it as "available" in the system.

Option [4] - Manage borrower details.

[4.1] Add Borrowers: Input the details of the new borrower. First you have to enter borrowers' NIC number. If it is a valid number, user will have a chance to continue by providing their details name, email, phone number, address etc. Once all details are provided, confirm to add the borrower to the system.

[4.2] Delete Borrowers: Here, you can delete a borrower's record from the system by entering the borrower ID.

Option [5] - Allows you to find specific borrowers or update their details.

[5.1] Search Borrowers: Input one of the borrower details like the borrower's ID, name, or contact information. Then you can see borrower details

[5.2] Update Borrowers: Modify the details of an existing borrower.

Input a unique identifier to find the desired borrower. Like ID in this option u have a chance to choose what details should be updated. According to the user input then user have a chance to update the new details.

Option [6] - This will take you back to the main menu of the system.

This is useful if you need to access other functionalities outside the librarian's scope.

8. Code Annex (Evidence of code implementation)

```
#include <iostream>
#include <conio.h>
#include <windows.h>
#include <string>
#include <fstream>
#include <sstream>
#include <cctype>
#include <algorithm>
#include <ctime>
#include <stdlib.h>

using namespace std;

int borrowingLimit = 3;
string availability = "available";
string librarianBookRecord = " ";


//function prototyping
void welcomeConsole();
void mainMenu();
void adminLogIn(); //Admin tasks
void adminMenu();
void libSignUp();
void changeDisplayFine();
float displayFineRate();
void updateFineRate();
void modifyDisplayBorrowingLimit();
void displayBorrowingLimit();
void modifyBorrowingLimit();
void librarianLogIn(); //Librarian tasks
void librarianMenu();
void addDeleteBooks();
void addBooks();
void deleteBooks();
void searchUpdateBooks();
void searchBooks();
void updateBooks();
void issueReturnedBooks();
void issueReturnedBooksMenu(string borrowerID, string bookID, int newBrrCount);
void issueBooks(string borrowerID, string bookID, int newBrrCount);
void acceptReturnedBooks(string borrowerID, string bookID, int newBrrCount);
void addDeleteBorrowers();
void addBorrowers();
void deleteBorrowers();
void searchUpdateBorrowers();
void searchBorrowers();
void updateBorrowers();
void borrower();

int main(){


	welcomeConsole();
	return 0;
}



void welcomeConsole(){
		system("Color 0F");
		cout << "\n\n" ;
		cout << "			**--**--**--**--**--**--**--**--**--**--**--**--**--**--**--**--**--**\n";
		cout << "\n\t\t**\t\t@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@		**";
		cout << "\n\t\t**\t\t@@                              @@        **";
		cout << "\n\t\t**\t\t@@          ~WELCOME                @@        **";
		cout << "\n\t\t**\t\t@@            TO                @@        **";
		cout << "\n\t\t**\t\t@@           LIBRARY                @@        **";
		cout << "\n\t\t**\t\t@@          LUMINAR!!!               @@        **";
		cout << "\n\t\t**\t\t@@                              @@        **";
		cout <<
"\n\t\t**\t\t@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@        **\n\n";
		cout << "           **--**--**--**--**--**--**--**--**--**--**--**--**--**--**--**--**--**--**\n\n";


		Sleep(2000);
		system("cls");
		mainMenu();
}


void mainMenu(){


	int option=0;
	do{
		//Display the user type
		cout << "\n\n\t============================-MAIN MENU-========================================\n";
		cout << "\t||        -USER TYPE-                              ||\n";
		cout << "\t||                [1]ADMIN                          ||\n";
		cout << "\t||                [2]LIBRARIAN                        ||\n";
		cout << "\t||                [3]EXIT                          ||\n";
		cout << "\t||                                          ||\n";
		cout <<
"\t===================================================================================\n\n";


		//get user input(main options: Admin,Librarian,Borrower)
		cout << "\t\tChoose an option: " ;
```

```
        cin >> option;

        switch(option){
            case 1:{
                system("cls");
                adminLogIn();
                break;
            }

            case 2:{
                system("cls");
                librarianLogIn();
                break;
            }

            case 3:{
                system("cls");
                cout << "\n\n          **--**--**--**--**--**--**--**--**--**--**--**--
**--**--**--**--**--**--**--**\n";
                cout << "\n\t\t\t  THANK YOU FOR USING LIBRARY
MANAGEMENT SYSTEM!\n\n";
                cout << "           **--**--**--**--**--**--**--**--**--**--**--**--**--
**--**--**--**--**--**--**\n\n\n";
                exit(0); //system call
            }

            default:{
                char opt = '0';
                cout <<"\n\t\t\tINVALID INPUT!! Do you want to try again?(y/n): ";
                cin >> opt;

                if(opt=='y'||opt=='Y'){
                    system("cls");
                    mainMenu();
```

```
                                        }

                                        else{
                                                system("cls");
                                                cout << "\n\n          **--**--**--**--**--**--**--**--**--**--
**--**--**--**--**--**--**--**--**--**\n";
                                                cout << "\n\t\t\t  THANK YOU FOR USING LIBRARY
MANAGEMENT SYSTEM!\n\n";
                                                cout << "           **--**--**--**--**--**--**--**--**--**--**--
**--**--**--**--**--**--**--**--**\n\n\n";
                                                exit(0); //system call
                                        }

                                }
                        }

                }while(option!=3);

}

//validate adminUserName and Password
void adminLogIn(){

        string userName;
        string password;
        int loginAttempts = 0;

        while(loginAttempts<3){
                string adminUserName = "1";
                string adminPassword = "1";

                char choice = '0';

                cout << "\n\n" ;
cout << "\t |%%%%%%%%%%%%%%%%%%%%%|-'ADMIN LOGIN'-|%%%%%%%%%%%%%%%%%%%%%%%%|\n\n";
cout << "\t\t\t [*]ENTER USERNAME: ";
cin >>  userName;
cout << "\t\t\t [*]ENTER PASSWORD: " ;
cin >> password;
cout << "\n\t |%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%|\n\n";


if(userName==adminUserName && password==adminPassword){
    //intialize the string
    string S = "\t\t\tADMIN LOGIN SUCCESSFULLY!!\n";

    //Travers the given string S
    for(int i = 0; i < S[i]; i++){
        cout << S[i];
        Sleep(100);
    }

    Sleep(2000);
    system("cls");
    adminMenu();

}

else{
    cout << "\t\t\tINCORRECT USERNAME OR PASSWORD!!\n\n";
    cout << "\t\t\tDo you want to try again?(y/n): ";
    cin >> choice;

        if (choice=='Y' || choice=='y'){
            system("cls");
```

```
                    loginAttempts++;
                    continue;
                }

                else{
                    system("cls");
                    mainMenu();
                }
            }
    }
    cout <<
"\n\n\n\t\t_____________________________________________________________________________________\n\n";
    cout << "\t\tYou have entered INCORRECT LOGIN-DETAILS 3 times.Please try again in 5
Seconds\n";
    cout <<
"\t\t_____________________________________________________________________________________\n\n";
    Sleep(5000);
    system("cls");
    adminLogIn();
}

//validate Librarians'UserName and Password (log in)
void librarianLogIn(){

    char choice = '0';
    string userName;
    string password;
    int loginAttempts = 0;

    while(loginAttempts < 3){
        cout << "\n\n" ;
        cout << "\t |%%%%%%%%%%%%%%%%%%%%%%%%|-'LIBRARIAN LOGIN'-
|%%%%%%%%%%%%%%%%%%%%%%%%%|\n\n";
```

```
                cout << "\t\t\t [*]ENTER USERNAME: ";
                cin >>  userName;
                cout << "\t\t\t [*]ENTER PASSWORD: " ;
                cin >> password;
                cout << "\n\t
|%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%|\n\n";


                //reading (myFile to validate librarian login)
                fstream myFile;                 //create an object named(myFile) from fstream class

                myFile.open("librarian.dat",ios::in);  //open librarian.dat file to read

                if(myFile.is_open()){
                        string line;       //To store file records line by line
                        while(getline(myFile, line)){  //while opend file has records to read
                                istringstream iss(line);//get the line without space(trim)
                                string libID, name, mail, phone, homeAddress, libName, pinNumber;
                                getline(iss, libID, '|');
                                getline(iss, name, '|');
                                getline(iss, mail, '|');
                                getline(iss, phone, '|');
                                getline(iss, homeAddress, '|');
                                getline(iss, libName, '|');
                                getline(iss, pinNumber, '|');

                                if(userName==libName && password==pinNumber){

                                        librarianBookRecord = libID;
                                        //intialize the string
                                        string S = "\t\t\tLIBRARIAN LOGIN SUCCESSFULLY!!\n";

                                        //Travers the given string S
                                        for(int i = 0; i < S[i]; i++){
```

```
                        cout << S[i];
                        Sleep(100);
                    }

                    Sleep(2000);
                    system("cls");
                    librarianMenu();
                }

            }

            cout << "\t\t\tINCORRECT USERNAME OR PASSWORD!!\n\n";
            cout << "\t\t\tDo you want to try again?(y/n) ";
            cin >> choice;

            if (choice=='Y' || choice=='y'){
                system("cls");
                loginAttempts++;
                continue;
            }

            else{
                system("cls");
                mainMenu();
            }

        }
        else{
        cout << "\t\t__'FILE DOES NOT EXIST!!'__";
        Sleep(2000);
        system("cls");
        librarianMenu();
        }
```

```
	}
	cout << "\n\n\n\t\t____________________________________________________________________________________________\n\n";
	cout << "\t\tYou have entered INCORRECT LOGIN-DETAILS 3 times.Please try again in 5 Seconds\n";
	cout << "\t\t_____________________________________________________________________________________\n\n";
	Sleep(5000);
	system("cls");
	librarianLogIn();

}



//admin menu panel
void adminMenu(){


	while(true){
		int option=0;


		cout << "\n\n\t+++++++++++++++++++++++++++++++-ADMIN MENU-+++++++++++++++++++++++++++++++++++++\n";
		cout << "\t|                                                   |\n";
		cout << "\t|               [1]-SIGN UP LIBRARIANS              |\n";
		cout << "\t|               [2]-UPDATE THE FINE RATE            |\n";
		cout << "\t|               [3]-MODIFY MAXIMUM BORROWING LIMITS |\n";
		cout << "\t|               [4]-BACK TO MAIN MENU               |\n";
		cout << "\t|___________________________________________________|\n\n";


		//get user input
```

```
        cout << "\t\tChoose an option: ";
        cin >> option;



        switch(option){
                case 1:{
                        system("cls");
                        libSignUp();
                        break;
                }

                case 2:{
                        system("cls");
                        changeDisplayFine();
                        break;
                }

                case 3:{
                        system("cls");
                        modifyDisplayBorrowingLimit();
                        break;
                }

                case 4:{
                        system("cls");
                        mainMenu();
                }

                default:{
                        char opt = '0';
                                cout <<"\n\t\t\tINVALID INPUT!! Do you want to try
again?(y/n): ";

                                cin >> opt;
```

```
                                        if(opt=='y'||opt=='Y'){
                                                system("cls");
                                                adminMenu();
                                        }

                                        else{
                                                system("cls");
                                                mainMenu();
                                        }
                                }
                        }
                }
        }

void libSignUp(){

        string nic,fullName,email,conNumber,address,userName,password;

        while(true){
                //Getting user input (NIC no) to check whether that librarian already exist in the
current system.
                cout << "\n\n\t |#######################|-'LIBRARIAN SIGN-UP INFO'-
|########################|\n\n";
                cout << "\t\t\tNIC Number: ";
                cin >> nic;
                cout << "\n\t
|####################################################################################|\n\n"
;

                if(nic.length() == 9 || nic.length() == 12 ){
                        break;
                }

                else{
```

```
                char option = '0';

                cout << "\n\t\t\tINVALID NIC NUMBER!!\n";
                cout << "\n\t\t\tDo you want to try again?(y/n):  ";
                cin >> option;

                if(option=='y'||option=='Y'){
                        system("cls");
                        continue;
                }

                else if(option=='n'||option=='N'){
                        system("cls");
                        adminMenu();
                }

                else{
                cout << "INVALID INPUT!!\n";
                cout << "\n\t\t_'Press any key to return Admin Menu'_";
                getch();
                system("cls");
                adminMenu();
                }
        }
}

//Create an Id for the librarian
string id = "LLIB#" + nic;              //LIB#200351501449

//reading (myFile to to check whether already exist)
fstream myFile;

myFile.open("librarian.dat",ios::in);   //open librarian.dat file to read
```

```
        if(myFile.is_open()){
                string line;                //To store file records line by line
                while(getline(myFile, line)){  //while opend file has records to read
                        istringstream ss(line);            //get the line without space(trim)
                        string libID;
                        getline(ss, libID, '|');

                        if(id==libID){
                                cout << "\t\t_'LIBRARIAN ALREADY EXIST'_\n\n";
                                myFile.close();
                                cout << "\n\t\t_'Press any key to return Admin Menu'_";
                                getch();
                                system("cls");
                                adminMenu();
                        }
                }
        }

        else{
                cout << "__'FILE DOES NOT EXIST!!'__";
                Sleep(2000);
                system("cls");
                adminMenu();
        }

        cout << "\t\t__'LIBRARIAN DOES NOT EXIST!!'__\n";
        Sleep(1000);
        system("cls");

        //getting user inputs to sign up librarians(name,email,address,contact info.)
        cout << "\n\n\t|########################|-'LIBRARIAN SIGN-UP FORM'-
|############################|\n\n";
```

```
	cout << "\t\tFULL NAME\t: ";
	cin.ignore();
	getline (cin,fullName);
	cout << "\t\tE-MAIL\t\t: ";
	cin >> email;
	cout << "\t\tCONTACT NUMBER\t: ";
	cin >> conNumber;
	cout << "\t\tADDRESS\t\t: ";
	cin >> address;


	cout << 
"\n\t_____________________________________________________________________________
___\n";
	cout << "\t^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^-LOGIN DETAILS-
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^\n\n";
	cout << "\t\tUSER NAME\t: ";
	cin >> userName;
	cout << "\t\tPASSWORD\t: ";
	cin >> password;
	cout << 
"\n\t|#############################################################################
###|\n\n";



	//saving informations in the file("librarian.dat")
	fstream myFileWrite;  //create object called myFileWrite from fstream class


	//calling open()function through File object.
	myFileWrite.open("librarian.dat",ios::app);   //file path mode




	if(!myFileWrite){
		cout << "FILE OPEN FAILED!!\n";
		Sleep(2000);
		system("cls");
```

```
			libSignUp();
		}
		else{
			myFileWrite << id << "|" << fullName << "|" << email << "|" << conNumber << "|"
<< address << "|" << userName << "|" << password << "\n";
			myFileWrite.close();

			//intialize the string
			string S = "\t\t\tLIBRARIAN SUCCESSFULLY ADDED!!\n";
			//Travers the given string S
			for(int i = 0; i < S[i]; i++){
				cout << S[i];
				Sleep(100);
			}

			Sleep(2000);
			system("cls");
			adminMenu();
		}

}



void changeDisplayFine(){
		char option = '0';
		char opt = '0';

		do{
			cout <<
"\n\n\t===================================================================
=====================\n";
			cout << "\t|+++++++++++++++++++++++++++++-'FINE RATE'-
++++++++++++++++++++++++++++++++++++++++++++|\n\n";
			cout << "\t\t\t[1].DISPLAY CURRENT FINE RATE
\n";
```

```
		cout << "\t\t\t[2].UPDATE FINE RATE                                   \n";
		cout << "\t\t\t[3].BACK TO ADMIN MENU {<<-}
\n";
		cout <<
"\n\t==========================================================================
===================\n\n";


		//get user input
		cout << "\t\tChoose an option: " ;
		cin >> opt;


		switch(opt){
			case '1':{
				displayFineRate();
				break;
			}


			case '2':{
				updateFineRate();
				break;
			}


			case '3':{
				system("cls");
				adminMenu();
				break;
			}


			default:{
				char opt = '0';
				cout <<"\n\t\t\tINVALID INPUT! Do you want to try again?(y/n): ";
				cin >> opt;


				if(opt=='y'||opt=='Y'){
```

```
                                        system("cls");
                                        adminMenu();
                                }

                                else if(option=='n'||option=='N'){
                                system("cls");
                                mainMenu();
                                }

                                else{
                                        cout << "INVALID INPUT!!\n";
                                        cout << "\n\t\t_'Press any key to return Admin Menu'_\n";
                                        getch();
                                        system("cls");
                                        adminMenu();
                                }
                        }
                }
        }while(opt==!3);

}



float displayFineRate(){

        float fineRate = 0.0;

        fstream inFineFile;

        inFineFile.open("fineRate.dat",ios::in);

        if(!inFineFile){
                cout << "\t\tFILE OPENED FAIL!!!\n";
                Sleep(1000);
                cout << "\n\t\t_'Press any key to return Admin Menu'_\n";
                getch();
                system("cls");
                adminMenu();
        }

        else{

                inFineFile >> fineRate;         //read the fine rate from the file

                cout << "\n\n\t\t----------------------------------------------------------------\n\n";
                cout << "\t\t\tCURRENT FINE RATE(Rs.): " << fineRate;
                cout << "\n\n\t\t----------------------------------------------------------------\n";

                inFineFile.close();

                Sleep(2000);
                system("cls");
                changeDisplayFine();
        }
        return fineRate;
}



void updateFineRate(){
        fstream outFineFile;

        outFineFile.open("fineRate.dat",ios::out);

        if(!outFineFile){
                cout << "FILE OPENED FAIL!!!\n";
                Sleep(1000);
                cout << "\n\t\t_'Press any key to return Admin Menu'_\n";
                getch();
                system("cls");
                adminMenu();
        }

        else{
                float fineRate = 0.0;
                while(true){
                        cout << "\n\t\t\tEnter new fine rate(Rs.): ";
                        cin >> fineRate;

                        if(fineRate < 0){
                                cout << "\t\tINVALID FINE RATE!!!\n";
                                cout << "\t\tPress any key to try again";
                                getch();
                                continue;
                        }
                        outFineFile << fineRate ;
                        outFineFile.close();
                        break;
                }
        }


        //intialize the string
        string S = "\n\t\t\tFINE CHANGED SUCCESSFULLY !!\n\n";
        //Travers the given string S
        for(int i = 0; i < S[i]; i++){
                cout << S[i];
                Sleep(100);
        }
```

```
	Sleep(1000);
	system("cls");
	changeDisplayFine();
}



void modifyDisplayBorrowingLimit(){
	char option = '0';
	char opt = '0';

	do{
		cout << 
"\n\n\t======================================================================
======================\n";
		cout << "\t|++++++++++++++++++++++++++++++-'BORROWING LIMIT'-
+++++++++++++++++++++++++++++++++++++++++|\n\n";
		cout << "\t\t\t[1].DISPLAY CURRENT MAXIMUM BORROWING LIMIT 
\n";
		cout << "\t\t\t[2].UPDATE BORROWING LIMIT 
\n";
		cout << "\t\t\t[3].BACK TO ADMIN MENU {<<-} 
\n";
		cout << 
"\n\t=======================================================================
=====================\n\n";

		//get user input
		cout << "\t\tChoose an option: " ;
		cin >> opt;

		switch(opt){
			case '1':{
				displayBorrowingLimit();
				break;
			}
```

```
case '2':{
        modifyBorrowingLimit();
        break;
}

case '3':{
        system("cls");
        adminMenu();
        break;
}

default:{
        char opt = '0';
        cout <<"\n\t\t\tINVALID INPUT! Do you want to try again?(y/n): ";
        cin >> opt;

        if(opt=='y'||opt=='Y'){
                system("cls");
                adminMenu();
        }

        else if(option=='n'||option=='N'){
        system("cls");
        mainMenu();
        }

        else{
                cout << "INVALID INPUT!!\n";
                cout << "\n\t\t_'Press any key to return Admin Menu'_\n";
                getch();
                system("cls");
                adminMenu();
        }
```

```
				}
			}
		}while(opt==!3);

}


void displayBorrowingLimit(){
		fstream inMaxBorrowFile;

		inMaxBorrowFile.open("borrowingLimit.dat",ios::in);

		if(!inMaxBorrowFile){
			cout << "\t\tFILE OPENED FAIL!!!\n";
			Sleep(1000);
			cout << "\n\t\t_'Press any key to return Admin Menu'_\n";
			getch();
			system("cls");
			adminMenu();
		}
		else{
			int borrowingLimit = 0;
			inMaxBorrowFile >> borrowingLimit;		//read the fine rate from the file
			cout << "\n\n\t\t-----------------------------------------------------------------\n\n";
			cout << "\t\t\tCURRENT BORROWING LIMIT(Books): " << borrowingLimit;
			cout << "\n\n\t\t-----------------------------------------------------------------\n";

			inMaxBorrowFile.close();

			Sleep(2000);
			system("cls");
			modifyDisplayBorrowingLimit();
		}
```

```
}



void modifyBorrowingLimit(){
	fstream outMaxBorrowFile;

	outMaxBorrowFile.open("borrowingLimit.dat",ios::out);

	if(!outMaxBorrowFile){
		cout << "FILE OPENED FAIL!!!\n";
		Sleep(1000);
		cout << "\n\t\t_'Press any key to return Admin Menu'_\n";
		getch();
		system("cls");
		adminMenu();
	}

	else{
		int borrowingLimit = 0;

		while(true){
			cout << "\n\t\t\tEnter new borrowing limit(Books): ";
			cin >> borrowingLimit;
			if(borrowingLimit < 0){
					cout << "\t\tINVALID BORROWING LIMIT!!!\n";
					cout << "\t\tPress any key to try again";
					getch();
					continue;
			}
		}
		outMaxBorrowFile << borrowingLimit ;	//write maximum borrowing limit in the
file
		outMaxBorrowFile.close();
	}



	//intialize the string
	string S = "\n\t\t\tBORROWING LIMIT CHANGED SUCCESSFULLY !!\n\n";
	//Travers the given string S
	for(int i = 0; i < S[i]; i++){
		cout << S[i];
		Sleep(100);
	}


	Sleep(1000);
	system("cls");
	modifyDisplayBorrowingLimit();
}



//librarian menu panel
void librarianMenu(){
	int opt=0;
	do{
		cout << "\n\n\t +++++++++++++++++++++++++++++++-LIBRARIANS' MENU-
+++++++++++++++++++++++++++++++++++\n";
		cout << "\t|                                                    |\n";
		cout << "\t|		[1]-ADD OR DELETE BOOKS			|\n";

		cout << "\t|		[2]-SEARCH OR UPDATE BOOKS
|\n";
		cout << "\t|		[3]-BOOK CHECKOUT OR RETURN BOOKS
[ISSUE/ACCEPT]		|\n";
		cout << "\t|		[4]-ADD OR DELETE BORROWERS
|\n";
		cout << "\t|		[5]-SEARCH OR UPDATE BORROWERS
|\n";
		cout << "\t|		[6]-BACK TO MAIN MENU			|\n";
		cout << 
"\t|________________________________________________________________________________
___|\n\n";


		//get user input
		cout << "\t\tChoose an option: ";
		cin >> opt;



		switch(opt){
			case 1:{
				system("cls");
				addDeleteBooks();
				break;
			}

			case 2:{
				system("cls");
				searchUpdateBooks();
				break;
			}

			case 3:{
				system("cls");
				issueReturnedBooks();
				break;
			}

			case 4:{
				system("cls");
				addDeleteBorrowers();
				break;
			}
```

```
case 5:{
        system("cls");
        searchUpdateBorrowers();
        break;
}

case 6:{
        system("cls");
        mainMenu();
}

default:{
        char opt = '0';
        cout <<"\n\t\t\tINVALID INPUT! Do you want to try again?(y/n): ";
        cin >> opt;

        if(opt=='y'||opt=='Y'){
                system("cls");
                librarianMenu();
        }

        else if(opt=='n'||opt=='N'){
        system("cls");
        mainMenu();
        }

        else{
        cout << "INVALID INPUT!!\n";
        cout << "\n\t\t_'Press any key to return librarian Menu'_";
        getch();
        system("cls");
        librarianMenu();
        }
```

```
                }
            }
        }while(opt!=6);
}


void addDeleteBooks(){

        int opt = 0;

        cout <<
"\n\n\t======================================================================
====================\n";
        cout << "\t|++++++++++++++++++++++++++++++-'ADD OR DELETE BOOKS'-
+++++++++++++++++++++++++++++++++++|\n\n";
        cout << "\t\t\t[1].ADD BOOKS{+}                                             \n";
        cout << "\t\t\t[2].DELETE BOOKS{-}                                           \n";
        cout << "\t\t\t[3].BACK TO LIBRARIANS' MENU {<<-}                                 \n";
        cout <<
"\n\t========================================================================
===================\n\n";

        //get user input
        cout << "\t\tChoose an option: " ;
        cin >> opt;


        switch(opt){
            case 1:{
                system("cls");
                addBooks();
                break;
            }

            case 2:{
```

```
				system("cls");
				deleteBooks();
				break;
			}

			case 3:{
				system("cls");
				librarianMenu();
				break;
			}

			default:{
				char opt = '0';
				cout <<"\n\t\t\tINVALID INPUT!! Do you want to try again?(y/n): ";
				cin >> opt;

				if(opt=='y'||opt=='Y'){
					system("cls");
					addDeleteBooks();
				}

				else{
					system("cls");
					librarianMenu();
				}
			}
		}
}

void searchUpdateBooks(){
		int opt = 0;
```

```
        cout <<
"\n\n\t================================================================
===================\n";
        cout << "\t|++++++++++++++++++++++++++++-'SEARCH 0R UPDATE BOOKS'-
+++++++++++++++++++++++++++++++++|\n\n";
        cout << "\t\t\t[1].SEARCH BOOKS{?}                              \n";
        cout << "\t\t\t[2].UPDATE BOOKS{()}                             \n";
        cout << "\t\t\t[3].BACK TO LIBRARIANS' MENU {<<-}               \n";
        cout <<
"\n\t================================================================
==================\n\n";

        //get user input
        cout << "\t\tChoose an option: " ;
        cin >> opt;



        switch(opt){
                case 1:{
                        system("cls");
                        searchBooks();
                        break;
                }

                case 2:{
                        system("cls");
                        updateBooks();
                        break;
                }

                case 3:{
                        system("cls");
                        librarianMenu();
                        break;
                }
                default:{
                        char opt = '0';
                        cout <<"\n\t\t\tINVALID INPUT!! Do you want to try again?(y/n): ";
                        cin >> opt;

                        if(opt=='y'||opt=='Y'){
                                system("cls");
                                searchUpdateBooks();
                        }
                        else{
                                system("cls");
                                librarianMenu();
                        }
                }
        }
}



void issueReturnedBooks(){



        string bookID, borrowerID;
        int bookFound = 0;
        int borrowerFound = 0;
        int bookLimitMatch = 0;
        int borrowingLimit = 0;
        int newBrrCount = 0;

        while(true){
                cout << "\n\n\t |#########################|-'ISSUE OR RETURN BOOK'-
|##########################|\n\n";
                cout << "\t\t\tBOOK ID\t\t: ";
                cin >> bookID;
                cout << "\t\t\tBORROWER ID\t: ";
                cin >> borrowerID;
                cout << "\n\t
|################################################################################|\n\n"
;

                transform(bookID.begin(),bookID.end(),bookID.begin(),::tolower);
                transform(borrowerID.begin(),borrowerID.end(),borrowerID.begin(),::tolower);

                fstream BookFile,BorrowFile;

                BookFile.open("books.dat",ios::in);
                BorrowFile.open("borrower.dat",ios::in);

                if(BookFile.is_open() && BorrowFile.is_open()){
                        string line;
                        while(getline(BookFile,line)){
                                string bookid;          //variables to store the details of the each books
                                istringstream iss(line);
                                getline(iss, bookid, '|');              //extract book details to variables

                                //convert bookid,title to lowercase
                                transform(bookid.begin(), bookid.end(), bookid.begin(), ::tolower);
                                cout << bookID << endl;
                                cout << bookid << endl;
                                getch();

                                if(bookID==bookid){

                                    bookFound++;
                                    break;
                                }
                        }
                        if(bookFound==0){
                                char opt = '0';
                                cout << "\t\tYOU CAN'T BORROW [BOOK DOES NOT EXIST OR
BOOK ALREADY ISSUED!!!]";
                                cout << "\n\t\tDO YOU WANT TO ISSUE ANOTHER BOOK?";
                                cin >> opt;
                                if(opt=='y'||opt=='Y'){
                                    system("cls");
                                    continue;
                                }

                                else if(opt=='n'||opt=='N'){
                                system("cls");
                                librarianMenu;
                                }

                                else{
                                cout << "\n\t\tINVALID INPUT!!\n";
                                cout << "\n\t\t_'Press any key to return librarian Menu'_";
                                getch();
                                system("cls");
                                librarianMenu();
                                }
                        }

                        //read the Maximum borrowlimit
                        fstream inMaxBorrowFile;
                        inMaxBorrowFile.open("borrowingLimit.dat", ios::in);

                        if(!inMaxBorrowFile){
                                cout << "\t\tFILE OPENED FAIL!!!\n";
                                Sleep(1000);
                                cout << "\n\t\t_'Press any key to return Admin Menu'_\n";
                                getch();
                                system("cls");
                                adminMenu();
                            }
                            inMaxBorrowFile >> borrowingLimit;



                            string line2;   //variable that store each line of the borrower file
                            //read each line of the borrower file
                            while(getline(BorrowFile,line2)){

                                //variables to store the details of the each borrower
                                string
borrowerid,name,phoneNo,email,address,joinedDate,brrCount,librarian;

                                istringstream iss(line2);
                                //extract borrower details to variables
                                getline(iss, borrowerid, '|');
                                getline(iss, name, '|');
                                getline(iss, phoneNo, '|');
                                getline(iss, email, '|');
                                getline(iss, address, '|');
                                getline(iss, joinedDate, '|');
                                getline(iss, brrCount, '|');
                                getline(iss, librarian, '|');



                                //convert borrower id to lowercase
                                transform(borrowerid.begin(), borrowerid.end(), borrowerid.begin(),
::tolower);

                                istringstream ss(brrCount);         // convert brrCount string value
into integer value
                                ss >> newBrrCount;
                                /*if(borrowerID==borrowerid, borrowingLimit > newBrrCount)
                                {
                                        borrowerFound++;
                                        bookLimitMatch++;
                                }*/



                                //check if the current borrower matches to the users' Target
                                if(borrowerID==borrowerid){
                                        borrowerFound++;
                                        if(newBrrCount < borrowingLimit){
                                                bookLimitMatch++;               //if book limit
matches(greater than or equal) the current count of books, (add+)
                                                break;
                                        }
                                }
                            }
                            if(borrowerFound==0){
                                char opt = '0';
                                cout << "\t\tBORROWER DOES NOT EXIST!!!\n";
                                cout << "\n\t\tDO YOU WANT TO TRY AGAIN(y/n)?";
                                cin >> opt;
                                if(opt=='y'||opt=='Y'){
                                        system("cls");
                                        continue;
                                }

                                else if(opt=='n'||opt=='N'){
                                        system("cls");
                                        librarianMenu;
                                }
```

```
                    else{
                            cout << "\n\t\tINVALID INPUT!!\n";
                            cout << "\n\t\t_'Press any key to return librarian Menu'_";
                            getch();
                            system("cls");
                            librarianMenu();
                    }
            }

            if(bookLimitMatch==0){
                    char opt = '0';
                    cout << "\t\tBORROWER EXCEDED THE MAXIMUM
BORROWING LIMIT!!!\n";
                    cout << "\n\t\tDO YOU WANT TO TRY AGAIN(y/n)?";
                    cin >> opt;
                    if(opt=='y'||opt=='Y'){
                            system("cls");
                            continue;
                    }

                    else if(opt=='n'||opt=='N'){
                            system("cls");
                            librarianMenu;
                    }

                    else{
                            cout << "\n\t\tINVALID INPUT!!\n";
                            cout << "\n\t\t_'Press any key to return librarian Menu'_";
                            getch();
                            system("cls");
                            librarianMenu();
                    }
            }
```

```
	}

	else{
		cout << "\n\t\tBOOK FILE OR BORROW FILE NOT FOUND!!";
	}

	BookFile.close();
	BorrowFile.close();

	//if both book and borrower exist check whether to issue or accept a book.
	if(bookFound==1 && borrowerFound==1 && bookLimitMatch==1){
		cout << "\n\t\tBOOK ID AND BORROWER ID VALID!! YOU CAN GO
AHEAD!!";
		Sleep(2500);
		system("cls");
		issueReturnedBooksMenu(borrowerID, bookID, newBrrCount);
	}
	else{
		char option = '0';
		cout << "\n\t\t BOOK ID OR BORROWER ID DOES NOT FOUND!!";
		cout << "\n\t\tDO YOU WANT TO TRY AGAIN?";
		cin  >> option;

		if(option=='y'||option=='Y'){
			system("cls");
			continue;
		}

		else if(option=='n'||option=='N'){
			system("cls");
			librarianMenu;
		}
```

```
                else{
                        cout << "\n\t\tINVALID INPUT!!\n";
                        cout << "\n\t\t_'Press any key to return librarian Menu'_";
                        getch();
                        system("cls");
                        librarianMenu();
                }
        }
    }
}

void issueReturnedBooksMenu(string borrowerID, string bookID, int newBrrCount){
    char opt = '0';
    do{
        cout <<
"\n\n\t=============================================================================================\n";
        cout << "\t|++++++++++++++++++++++-'ISSUE' OR 'ACCEPT RETURNED
BOOKS'"-+++++++++++++++++++++++|\n\n";
        cout << "\t\t\t[1].ISSUE BOOKS{+}                                    \n";
        cout << "\t\t\t[2].ACCEPT RETURNED BOOKS{-}
\n";
        cout << "\t\t\t[3].BACK TO LIBRARIANS' MENU {<<-}
\n";
        cout <<
"\n\t============================================================================================\n\n";

        //get user input
        cout << "\t\tChoose an option: ";
        cin >> opt;



        switch(opt){
            case '1':{
                system("cls");
                        issueBooks(borrowerID, bookID, newBrrCount);
                        break;
                }

                case '2':{
                        system("cls");
                        acceptReturnedBooks(borrowerID, bookID, newBrrCount);
                        break;
                }

                case '3':{
                        system("cls");
                        librarianMenu();
                        break;
                }

                default:{
                        char option = '0';
                        cout <<"\n\t\t\tINVALID INPUT! Do you want to try again?(y/n): ";
                        cin >> opt;

                        if(option=='y'||option=='Y'){
                                system("cls");
                                continue;
                        }

                        else if(option=='n'||option=='N'){
                        system("cls");
                        librarianMenu();
                        }

                        else{
                        cout << "\t\tINVALID INPUT!!\n";
```

```
                        cout << "\n\t\t_'Press any key to return librarian Menu'_";
                        getch();
                        system("cls");
                        librarianMenu();
                        }
                }
        }
    }while(opt!='3');
}


void issueBooks(string borrowerID, string bookID, int newBrrCount){


    string bookTitle;
    float fine = 0.0;

    //get current date
    time_t now = time(0);
    tm *ltm = localtime(&now);

    int year = 1900 + ltm->tm_year;
    int month = 1 + ltm->tm_mon;
    int day = ltm->tm_mday;

    tm issueDate = {};
    issueDate.tm_year = year - 1900;
    issueDate.tm_mon = month - 1;
    issueDate.tm_mday = day;

    //calculate the due date
    tm dueDate = issueDate;
    dueDate.tm_mday +=14;
```

```
	mktime(&dueDate);


	//open book.dat file to read
	fstream BookFile,TempFile1;
	BookFile.open("books.dat", ios::in);
	TempFile1.open("temp.dat", ios::out);


	//when borrowing a book displays 'unavailable' in "books.dat"
	if(BookFile.is_open() && TempFile1.is_open()){
		string line;
		while(getline(BookFile,line)){
			istringstream iss(line);
			string bookid, title, author, publisher, genre, price, availability,
addedDate,librarian;
		getline(iss,bookid,'|');
		getline(iss,title,'|');
		getline(iss,author,'|');
		getline(iss,publisher,'|');
		getline(iss,genre,'|');
		getline(iss,price,'|');
		getline(iss,availability,'|');
		getline(iss,addedDate,'|');
		getline(iss,librarian,'|');


		transform(bookid.begin(), bookid.end(), bookid.begin(), ::tolower);


		if(bookID == bookid){
			transform(bookid.begin(), bookid.end(), bookid.begin(), ::toupper);
			bookTitle = title;
			TempFile1 << bookid << "|" << title << "|" << author << "|" << publisher <<
"|" << genre << "|" << price << "|" << "UNAVAILABLE" << "|" << addedDate << "|" << librarian
<< "\n";
			}
			else{
```

```
                TempFile1 << line << "\n";
            }
        }
        BookFile.close();
        BookFile.flush();
        TempFile1.close();
        TempFile1.flush();

        remove("books.dat");
        rename("temp.dat", "books.dat");
    }
    else{
        cout << "\n\t\tBOOK FILE OR TEMP FILE DOES NOT FOUND!!!\n";
        Sleep(2000);
        cout << "Press any key to return Librarian Menu";
        getch();
        system("cls");
        librarianMenu();
    }

    fstream BorrowingRecordFile;
    BorrowingRecordFile.open("borrowingRec.dat",ios::app);

    if(!BorrowingRecordFile.is_open()){
        cout << "\n\t\tBORROWING RECORD FILE DOES NOT FOUND!!!\n";
        Sleep(2000);
        cout << "Press any key to return Librarian Menu";
        getch();
        system("cls");
        librarianMenu();
    }

    else{
```

```
		transform(borrowerID.begin(), borrowerID.end(), borrowerID.begin(),::toupper);
		transform(bookTitle.begin(), bookTitle.end(), bookTitle.begin(),::toupper);
		transform(bookID.begin(), bookID.end(), bookID.begin(),::toupper);

		BorrowingRecordFile << borrowerID << "|" << bookTitle << "|" << bookID << "|" 
<< year << "/" << month << "/" << day << "|" << (1900+dueDate.tm_year) << "/" << 
(1+dueDate.tm_mon) << "/" << (dueDate.tm_mday) << "\n";
		BorrowingRecordFile.close();
	}

	cout << "\t---------------------------------------------------------------------------\n";
	cout << "\t\t\tBORROWER ID\t\t: " << borrowerID << endl;
	cout << "\t\t\tBOOK ID\t\t\t: " << bookID << endl;
	cout << "\t\t\tBOOK TITLE\t\t: " << bookTitle << endl;
	cout << "\t\t\tISSUE DATE\t\t: " << year << "/" << month << "/" << day << endl;
	cout << "\t\t\tDUE DATE\t\t: " << (1900+dueDate.tm_year) << "/" << (1+dueDate.tm_mon) 
<< "/" << (dueDate.tm_mday) << "\n";
	cout << "\t---------------------------------------------------------------------------\n";

	//intialize the string
	string S = "\n\t\t\tBOOK ISSUED SUCCESSFULLY!!!";
	//Travers the given string S
	for(int i = 0; i < S[i]; i++){
		cout << S[i];
		Sleep(100);
	}

	//convert the borrowerID to lowercase
	transform(borrowerID.begin(), borrowerID.end(), borrowerID.begin(), ::tolower);

	//open original Borrower File for reading and TempFile for writing
	fstream BorrowerFile, TempFile2;
	BorrowerFile.open("borrower.dat",ios::in);
```

```
TempFile2.open("temp.dat",ios::out);

//check if the borrower file is successfully opened
if(!BorrowerFile.is_open() && !TempFile2.is_open()){
        cout << "\t\t\tFILE OPENINING ERROR!!!\n";
        cout << "\n\t\tPress any key to return Librarian Menu";
        getch();
        system("cls");
        librarianMenu();
}

string line;    //variable that store each line of the borrower file

//read each line of the borrower file
while(getline(BorrowerFile,line)){

        //variables to store the details of the each borrower
        string borrowerid,name,phoneNo,email,address,joinedDate,brrCount,librarian;

        istringstream iss(line);
        //extract borrower details to variables
        getline(iss, borrowerid, '|');
        getline(iss, name, '|');
        getline(iss, phoneNo, '|');
        getline(iss, email, '|');
        getline(iss, address, '|');
        getline(iss, joinedDate, '|');
        getline(iss, brrCount, '|');
        getline(iss, librarian, '|');

        //convert borrower id to lowercase
        transform(borrowerid.begin(), borrowerid.end(), borrowerid.begin(), ::tolower);
```

```
                newBrrCount++;


                //check if the current borrower matches to the users' Target
                if(borrowerID==borrowerid){
                        //convert borrower id to uppercase
                        transform(borrowerid.begin(), borrowerid.end(), borrowerid.begin(),
::toupper);


                        //write the updated details(newBorrow count) to the temporary file
                        TempFile2 << borrowerID << "|" << name << "|" << phoneNo << "|" << email
<< "|" << address << "|" << joinedDate << "|" << newBrrCount << "|" << librarian << "\n";
                }


                else{
                        TempFile2 << line << "\n";
                }
        }


        //close both files
        BorrowerFile.flush();
        BorrowerFile.close();


        TempFile2.flush();
        TempFile2.close();


        //delete BookFile and rename TempFile
        remove("borrower.dat");
        rename("temp.dat","borrower.dat");




        char option = '0';
        Sleep(1000);
        cout << "\n\t\t\tDo you want to borrow another book?(y/n): ";
```

```
	cin >> option;
	if(option=='y'||option=='Y'){
		system("cls");
		issueReturnedBooks();
	}

	else if(option=='n'||option=='N'){
		system("cls");
		librarianMenu();
	}

	else{
		cout << "\t\tINVALID INPUT!!\n";
		cout << "\n\t\t_'Press any key to return librarian Menu'_";
		getch();
		system("cls");
		librarianMenu();
	}
}


void acceptReturnedBooks(string borrowerID, string bookID, int newBrrCount){

	cout << borrowerID << endl;
	cout << bookID << endl;
	cout << newBrrCount;
	getch();

	//calculate the return date
	time_t returnDate = time(0);

	// Get the user input (return data)
	int year, month, day;
```

```
        cout << "\t\tEnter Return Date(yyyy/mm/dd): ";
        cin >> year >> month >> day;

        // Find the return the date
        tm returnDateCal = {};
        returnDateCal.tm_year = year - 1900;
        returnDateCal.tm_mon = month - 1;
        returnDateCal.tm_mday = day;

        // Get the fine rate from fineRateDisplay()
        float fineRate = displayFineRate();
        cout << fineRate;
        getch();




}



void addDeleteBorrowers(){
        int opt = 0;


        cout <<
"\n\n\t======================================================================
====================\n";
        cout << "\t|+++++++++++++++++++++++++++-'ADD OR DELETE BORROWERS'-
+++++++++++++++++++++++++++++++|\n\n";
        cout << "\t\t\t[1].ADD BORROWERS{+}                              \n";
        cout << "\t\t\t[2].DELETE BORROWERS{-}                            \n";
        cout << "\t\t\t[3].BACK TO LIBRARIANS' MENU {<<-}                  \n";
        cout <<
"\n\t========================================================================
===================\n\n";
```

```
//get user input
cout << "\t\tChoose an option: " ;
cin >> opt;



switch(opt){
        case 1:{
                system("cls");
                addBorrowers();
                break;
        }

        case 2:{
                system("cls");
                deleteBorrowers();
                break;
        }

        case 3:{
                system("cls");
                librarianMenu();
                break;
        }

        default:{
                char opt = '0';
                cout <<"\n\t\t\tINVALID INPUT!! Do you want to try again?(y/n): ";
                cin >> opt;

                if(opt=='y'||opt=='Y'){
                        system("cls");
                        addDeleteBorrowers();
                }
```

```
                    else if(opt=='n'||opt=='N'){
                            system("cls");
                            librarianMenu();
                    }


                    else{
                    cout << "INVALID INPUT!!\n";
                    cout << "\n\t\t_'Press any key to return Librarian Menu'_";
                    getch();
                    system("cls");
                    librarianMenu();
                    }
            }
        }
}



void searchUpdateBorrowers(){
            int opt = 0;


        cout << 
"\n\n\t==========================================================================
===================\n";
        cout << "\t|++++++++++++++++++++++++++++-'SEARCH OR UPDATE BORROWERS'-
++++++++++++++++++++++++++++|\n\n";
        cout << "\t\t\t[1].SEARCH BORROWERS{?}                                    \n";
        cout << "\t\t\t[2].UPDATE BORROWERS{()}                                   \n";
        cout << "\t\t\t[3].BACK TO LIBRARIANS' MENU {<<-}                         \n";
        cout << 
"\n\t===========================================================================
==================\n\n";


        //get user input
        cout << "\t\tChoose an option: " ;
        cin >> opt;
```

```
switch(opt){
        case 1:{
                system("cls");
                searchBorrowers();
                break;
        }

        case 2:{
                system("cls");
                updateBorrowers();
                break;
        }

        case 3:{
                system("cls");
                librarianMenu();
                break;
        }

        default:{
                char opt = '0';
                cout <<"\n\t\t\tINVALID INPUT!! Do you want to try again?(y/n): ";
                cin >> opt;

                if(opt=='y'||opt=='Y'){
                        system("cls");
                        searchUpdateBorrowers();
                }
                else if(opt=='n'||opt=='N'){
                        librarianMenu();
                }
```

```
				else{
					cout << "\t\t\tINVALID INPUT!!!\n";
					cout << "Press any key to return Librarian Menu";
					getch();
					system("cls");
					librarianMenu();
				}
			}
		}
}

void addBooks(){

		string isbn, title, author, price, publisher, genre;

		while(true){
			//Getting user input (NIC no) to check whether that book already exist in the current system.
			cout << "\n\n\t |#########################|-'ADD BOOK INFO'-|##########################|\n\n";
			cout << "\t\t\tBOOK ISBN: ";
			cin >> isbn;
			cout << "\n\t |###################################################################################|\n\n";

			if(isbn.length() == 10 || isbn.length() == 13 ){
				break;
			}

			else{
				char option = '0';

				cout << "\n\t\t\tINVALID ISBN!!\n";
```

```
                cout << "\n\t\t\tDo you want to try again?(y/n):  ";
                cin >> option;

                if(option=='y'||option=='Y'){
                        system("cls");
                        continue;
                }

                else if(option=='n'||option=='N'){
                        system("cls");
                        librarianMenu();
                }

                else{
                        cout << "\t\t\tINVALID INPUT!!!\n";
                        cout << "Press any key to return Librarian Menu";
                        getch();
                        system("cls");
                        librarianMenu();
                }
        }
}

//Create an Id for the book
string bookId = "LBK#" + isbn;                  //LBOOK#20035150144

//read the books.dat file and check whether that book already exist in the system
fstream BookFile;

BookFile.open("books.dat",ios::in);

if(BookFile.is_open()){
        string line;
```

```
		while(getline(BookFile, line)){
			istringstream iss(line);
			string searchBookID;
			getline(iss, searchBookID, ',');

			if(bookId==searchBookID){
				cout << "\t\t_'THIS BOOK ALREADY EXIST'_\n\n";
				BookFile.close();
				cout << "\n\t\t_'Press any key to return librarian Menu'_";
				getch();
				system("cls");
				librarianMenu();
			}
		}
	}

	else{
		cout << "\t\t__'FILE DOES NOT EXIST!!'__";
		Sleep(2000);
		system("cls");
		librarianMenu();
	}

	cout << "\t\t__'THIS BOOK DOES NOT EXIST!!'__\n";
	Sleep(1000);
	system("cls");



	//getting user inputs to add books(Title,Author,Price,Publisher,Genre)
	cout << "\n\n\t|#######################|-'ADD BOOKS'-
|#############################################|\n\n";
	cout << "\t\tTITLE\t\t: ";
	cin.ignore();
```

```
        getline(cin, title);
        cout << "\t\tAUTHOR\t\t: ";
        getline(cin, author);
        cout << "\t\tPUBLISHER\t: ";
        getline(cin, publisher);;
        cout << "\t\tGENRE\t\t: ";
        getline(cin, genre);;
        cout << "\t\tPRICE(LKR)\t: ";
        cin >> price;
        cout << 
"\n\t|####################################################################################
###|\n\n";

        //create object called BookFileWrite from fstream class.
        fstream BookFileWrite;

        BookFileWrite.open("books.dat",ios::app);

        if(!BookFileWrite){
                cout << "\t\tFILE CANNOT BE OPENED!!";
                cout << "\n\t\t_'Press any key to return librarian Menu'_";
                getch();
                system("cls");
                librarianMenu();
        }
        else{

                //get current date
                time_t now = time(0);
                tm *ltm = localtime(&now);

                int year = 1900 + ltm->tm_year;
                int month = 1 + ltm->tm_mon;
                int day = ltm->tm_mday;
```

```
			transform(title.begin(), title.end(), title.begin(),::toupper);
			transform(author.begin(), author.end(), author.begin(),::toupper);
			transform(publisher.begin(), publisher.end(), publisher.begin(),::toupper);
			transform(genre.begin(), genre.end(), genre.begin(),::toupper);

			BookFileWrite << bookId << "|" << title << "|" << author << "|" << publisher << "|"
<< genre << "|" << price << "|" << availability << "|" << year << "/" << month << "/" << day << "|"
<< librarianBookRecord << "\n";
			BookFileWrite.close();

			//intialize the string
			string S = "\t\t\tTHE BOOK SUCCESSFULLY ADDED!!\n";
			//Travers the given string S
			for(int i = 0; i < S[i]; i++){
				cout << S[i];
				Sleep(100);
			}

			Sleep(2000);
			system("cls");
			librarianMenu();
		}
}

void deleteBooks(){

		int deletedCount = 0;
		string eraseTarget;

		cout << "\n\n\t |######################|-'DELETE BOOKS'-
|#################################|\n\n";
		cout << "\t\tENTER ID/TITLE\t: ";
```

```
        cin.ignore();
        getline(cin, eraseTarget);
        cout << "\n\t
|##############################################################################|\n\n"
;

        //convert inputs to lowercase for case-insensitive comparison
        transform(eraseTarget.begin(), eraseTarget.end(), eraseTarget.begin(), ::tolower);

        fstream BookFile, TempFile;
        BookFile.open("books.dat", ios::in);
        TempFile.open("temp.dat", ios::out);

        if(!BookFile.is_open()){
                "\t\t\tFILE OPENINING ERROR!!!\n";
                cout << "\t\tPress any key to return Librarian Menu";
                getch();
                system("cls");
                librarianMenu();
        }

        string line;
        while(getline(BookFile, line)){
                string bookID, title;
                istringstream iss(line);
                getline(iss, bookID, '|');
                getline(iss, title, '|');

                //convert details to lowercase for case-insensitive comparison
                transform(bookID.begin(), bookID.end(), bookID.begin(), ::tolower);
                transform(title.begin(), title.end(), title.begin(), ::tolower);

                //check whether the file matches eraseTarget
                if(eraseTarget==bookID||eraseTarget==title){
```

```
                deletedCount++;
            }
            else{
                TempFile << line << "\n";     //write to the temporary file if it's not the target
book
            }
        }

        BookFile.close();
        TempFile.close();

        remove("books.dat");
        rename("temp.dat","books.dat");

        if(deletedCount>0){
            //intialize the string
            string S = "\t\t\tTHE BOOK DELETED SUCCESSFULLY !!\n";
            //Travers the given string S
            for(int i = 0; i < S[i]; i++){
                cout << S[i];
                Sleep(100);
            }

            Sleep(2000);
            cout << "\n\n\t\tPress any key to return Librarian Menu";
            getch();
            system("cls");
            librarianMenu();
        }

        else{
            cout << "\t\tBOOK NOT FOUND!!!\n";
            Sleep(2000);
```

```
                    cout << "\n\t\tPress any key to return Librarian Menu";
                    getch();
                    system("cls");
                    librarianMenu();
          }
}



void searchBooks(){
          string userInput = " ";
          cout << "\n\n\t |######################|-'SEARCH BOOKS'-
|##################################|\n\n";
          cout << "\tENTER
ID/TITLE/AUTHOR/PUBLISHER/GENRE/AVAILABILITY/ENTERED LIBRARIAN: ";
          cin.ignore();
          getline(cin,userInput);
          cout << "\n\t
|############################################################################|\n\n"
;

          transform(userInput.begin(),userInput.end(),userInput.begin(),::tolower);   //converting user
input to lowercase <algorithm>

          //read books.dat file and get the line by line

   fstream BookFile;
   BookFile.open("books.dat",ios::in);

   int matchCount = 0;           //This is a counter for found book.

   //search matching items for the userInput
   if(BookFile.is_open()){
          string line;

           string bookID, title, author, publisher, genre, price, availability, addedDate, librarian;
```

```
	while(getline(BookFile,line)){
		istringstream iss(line);
		getline(iss,bookID,'|');
		getline(iss,title,'|');
		getline(iss,author,'|');
		getline(iss,publisher,'|');
		getline(iss,genre,'|');
		getline(iss,price,'|');
		getline(iss,availability,'|');
		getline(iss,addedDate,'|');
		getline(iss,librarian,'|');

		//converting all the informations to lowercase
			transform(bookID.begin(),bookID.end(),bookID.begin(),::tolower);
			transform(title.begin(),title.end(),title.begin(),::tolower);
			transform(author.begin(),author.end(),author.begin(),::tolower);
			transform(publisher.begin(),publisher.end(),publisher.begin(),::tolower);
			transform(genre.begin(),genre.end(),genre.begin(),::tolower);
			transform(availability.begin(),availability.end(),availability.begin(),::tolower);
			transform(librarian.begin(),librarian.end(),librarian.begin(),::tolower);


	if(userInput==bookID||userInput==title||userInput==author||userInput==publisher||userInput=
=genre||userInput==availability||userInput==librarian){

				matchCount++;		//increment the counter when match is found
				cout << "\t-----------------------------------------------------------------------
-----\n";
				cout << "\t\t\tBOOK ID\t\t\t: " << bookID << endl;
				cout << "\t\t\tTITLE\t\t\t: " << title << endl;
				cout << "\t\t\tAUTHOR\t\t\t: " << author << endl;
				cout << "\t\t\tPUBLISHER\t\t: " << publisher << endl;
				cout << "\t\t\tGENRE\t\t\t: " << genre << endl;
```

```
                    cout << "\t\t\tPRICE\t\t\t: " << price << endl;
                    cout << "\t\t\tAVAILABILITY\t\t: " << availability << endl;
                    cout << "\t\t\tADDED DATE\t\t: " << addedDate << endl;
                    cout << "\t\t\tTHE BOOK ENTERED BY\t: " << librarian << endl;
                    cout << "\t---------------------------------------------------------------------
-----\n";
            }
        }

        if(matchCount==0){
            char opt = '0';
            cout << "\n\n\t\t\tTHE BOOK IS NOT FOUND!!!\n";
            cout << "\n\t\tDO YOU WANT TO SEARCH ANOTHER BOOK?(y/n): ";
            cin >> opt;

            if(opt=='y'||opt=='Y'){
                system("cls");
                searchBooks();
            }

            else if(opt=='n'||opt=='N'){
                system("cls");
                librarianMenu();
            }

            else{
                cout << "\n\n\t\t\tINVALID INPUT!!!\n";
                cout << "\n\t\tPress any key to return Librarian Menu";
                getch();
                system("cls");
                librarianMenu();
            }
        }
```

```
	}

	else{
		cout << "\t\t\tFILE IS NOT OPENED!!!\n";
		Sleep(1000);
		cout << "\n\t\t_'Press any key to return librarian Menu'_";
		getch();
		system("cls");
		librarianMenu();
	}

	BookFile.close();
	cout << "\n\t\t_'Press any key to return librarian Menu'_";
	getch();
	system("cls");
	librarianMenu();
}



void updateBooks(){

	string updateTarget;
	cout << "\n\n\t |########################|-'UPDATE BOOKS'-
|#########################################|\n\n";
	cout << "\t\tENTER THE BOOK ID/TITLE TO UPDATE\t: ";
	cin.ignore();
	getline(cin, updateTarget);
	cout << "\n\t
|################################################################################
###|\n\n";

	//convert the usser's input to lowercase
	transform(updateTarget.begin(), updateTarget.end(), updateTarget.begin(), ::tolower);
```

```
//open original BookFile for reading and TempFile for writing
fstream BookFile, TempFile;
BookFile.open("books.dat",ios::in);
TempFile.open("temp.dat",ios::out);

//check if the book file is successfully opened
if(!BookFile.is_open() && !TempFile.is_open()){
        cout << "\t\t\tFILE OPENINING ERROR!!!\n";
        cout << "\n\t\tPress any key to return Librarian Menu";
        getch();
        system("cls");
        librarianMenu();
}

int updatedCount = 0;
string line;     //variable that store each line of the book file

while(getline(BookFile,line)){

        //variables to store the details of the each books
        string bookID, title, author, publisher, genre, price, availability, addedDate, librarian;

        istringstream iss(line);
        //extract book details to variables
        getline(iss, bookID, '|');
        getline(iss, title, '|');
        getline(iss, author, '|');
        getline(iss, publisher, '|');
        getline(iss, genre, '|');
        getline(iss, price, '|');
        getline(iss, availability, '|');
        getline(iss, addedDate, '|');
```

```
                getline(iss, librarian, '|');

                //convert bookid,title to lowercase
                transform(bookID.begin(), bookID.end(), bookID.begin(), ::tolower);
                transform(title.begin(), title.end(), title.begin(), ::tolower);

                //check if the current book matches to the users' Target
                if(updateTarget==bookID||updateTarget==title){
                        updatedCount++;

                        int opt = 0;

                        while(true){
                                cout <<
"\n\t============================================================================
====================\n";
                                cout << "\t|+++++++++++++++++++++++++++++++-'UPDATE
BOOKS'-++++++++++++++++++++++++++++++++++++++++++|\n\n";
                                cout << "\t\t\t[1].TITLE                              \n";
                                cout << "\t\t\t[2].AUTHOR
\n";
                                cout << "\t\t\t[3].GENRE                              \n";
                                cout << "\t\t\t[4].AVAILABILITY
                                  \n";
                                cout << "\t\t\t[5].BACK TO (SEARCH OR UPDATE BOOKS)
MENU                              \n";
                                cout <<
"\n\t============================================================================
====================\n\n";

                                //get user input
                                cout << "\t\tChoose an option: " ;
                                cin >> opt;

                                if(opt < 1 || opt > 6){
                                        cout << "\n\n\t\t\tINVALID INPUT!!!\n";
                cout << "\n\t\tPress any key to re-enter";
                getch();
                system("cls");
                continue;
            }
            break;
        }

        if(opt == 1){
            //update title
            cout << "\n\t--------------------------------------------------------------------
----------------\n";
            cout << "\t\tCURRENT TITLE\t\t: " << title <<"\n\n";
            cout << "\n\t\tNEW TITLE\t\t: ";
            cin.ignore();
            getline(cin,title);
            cout << "\t--------------------------------------------------------------------
--------------\n";

        }
        else if(opt == 2){
            //update author
            cout << "\n\t--------------------------------------------------------------------
----------------\n";
            cout << "\t\tCURRENT AUTHOR\t\t: " << author <<"\n\n";
            cout << "\n\t\tNEW AUTHOR\t\t: ";
            cin.ignore();
            getline(cin,author);
            cout << "\t--------------------------------------------------------------------
--------------\n";

        }
        else if(opt == 3){
            //update genre
```

```
                        cout << "\n\t--------------------------------------------------------------------
----------------\n";
                        cout << "\t\tCURRENT GENRE\t\t: " << genre <<"\n\n";
                        cout << "\n\t\tNEW GENRE\t\t: ";
                        cin.ignore();
                        getline(cin,genre);
                        cout << "\t----------------------------------------------------------------------
--------------\n";
                }
                else if(opt == 4){
                        //update availability
                        cout << "\n\t--------------------------------------------------------------------
----------------\n";
                        cout << "\t\tCURRENT AVAILABILITY\t: " << availability <<"\n\n";
                        cout << "\n\t\tNEW AVAILABILITY\t: ";
                        cin.ignore();
                        getline(cin,availability);
                        cout << "\t----------------------------------------------------------------------
--------------\n";
                }
                else if(opt == 5){
                        system("cls");
                        searchUpdateBooks();
                }

                //convert to uppercase
                transform(bookID.begin(), bookID.end(), bookID.begin(), ::toupper);
                transform(title.begin(), title.end(), title.begin(), ::toupper);
                transform(author.begin(), author.end(), author.begin(), ::toupper);
                transform(genre.begin(), genre.end(), genre.begin(), ::toupper);
                transform(availability.begin(), availability.end(), availability.begin(),
::toupper);


                //write the updated details to the temporary file
```

```
                    TempFile << bookID << "|" << title << "|" << author << "|" << publisher <<
"|" << genre << "|" << price << "|" << availability << "|" << addedDate << "|" << librarian << "\n";

                }
                else{
                    TempFile << line << "\n";
                }


        }
        //close both files
        BookFile.flush();
        BookFile.close();

        TempFile.flush();
        TempFile.close();

        //delete BookFile and rename TempFile
        remove("books.dat");
        rename("temp.dat","books.dat");

        if(updatedCount > 0){

                char opt = '0';

                //intialize the string
                string S = "\n\t\t\tTHE BOOK UPDATED SUCCESSFULLY!!\n";
                //Travers the given string S
                for(int i = 0; i < S[i]; i++){
                    cout << S[i];
                    Sleep(100);
                }

                Sleep(1000);
                cout << "\n\t\t\tDo you want to (SEARCH/UPDATE) again?(y/n)\t: ";
                cin >> opt;
                if(opt = 'y' || opt == 'Y'){
                        system("cls");
                        searchUpdateBooks();
                }
                else{
                        system("cls");
                        librarianMenu();
                }
        }

        else{
                cout << "\t\t\tBOOK NOT FOUND!!\n";
                cout << "\n\t\tPress any key to (SEARCH/UPDATE) Menu again";
                getch();
                system("cls");
                searchUpdateBooks();
        }



}


void addBorrowers(){
        string nic, name, phoneNo,  email, address ;


        while(true){
                //Getting user input (NIC no) to check whether that borrower already exist in the
current system.
                cout << "\n\n\t |#######################|-'ADD BORROWER INFO'-
|########################|\n\n";
                cout << "\t\t\tNIC NUMBER: ";
                cin >> nic;
                cout << "\n\t
|##############################################################################|\n\n"
;
```

```
		if(nic.length() == 9 || nic.length() == 12 ){
			break;
		}

		else{
			char option = '0';

			cout << "\n\t\t\tINVALID NIC NUMBER!!\n";
			cout << "\n\t\t\tDo you want to try again?(y/n):  ";
			cin >> option;

			if(option=='y'||option=='Y'){
				system("cls");
				continue;
			}

			else{
				system("cls");
				librarianMenu();
			}
		}
	}

	//Create an Id for the Borrower
	string borrowerID = "LBOR#" + nic;		//LIBB#200351501449

	//read borrower File and see whether is that person already exist in the system
	fstream BorrowerFile;

	BorrowerFile.open("borrower.dat",ios::in);

	if(BorrowerFile.is_open()){
		string line;
		while(getline(BorrowerFile, line)){
			istringstream iss(line);
			string borrowerNo;
			getline(iss,borrowerNo,'|');

			if(borrowerID==borrowerNo){
				cout << "\t\t_'THIS BORROWER ALREADY EXIST'_\n\n";
				BorrowerFile.close();
				cout << "\n\t\t_'Press any key to return librarian Menu'_";
				getch();
				system("cls");
				librarianMenu();
			}
		}
	}

	else{
		cout << "\t\t__'FILE DOES NOT EXIST!!'__";
		Sleep(2000);
		system("cls");
		librarianMenu();
	}

	cout << "\t\t__'THIS BORROWER DOES NOT EXIST!!'__\n";
	Sleep(1000);
	system("cls");



	//getting user inputs to add borrower(Name, phoneNo, email, address)
	cout << "\n\n\t|########################|-'ADD BORROWERS'-
|###########################################|\n\n";
	cout << "\t\tNAME\t\t: ";
```

```
        cin.ignore();
        getline(cin, name);
        cout << "\t\tPHONE NUMBER\t: ";
        getline(cin, phoneNo);
        cout << "\t\tE MAIL\t\t: ";
        getline(cin, email);;
        cout << "\t\tADDRESS\t\t: ";
        getline(cin, address);;
        cout <<
"\n\t|###########################################################################
###|\n\n";


        //create object called BorrowerFileWrite from fstream class.
        fstream BorrowerFileWrite;


        BorrowerFileWrite.open("borrower.dat",ios::app);


        if(!BorrowerFileWrite){
                cout << "\t\tFile cannot be opened!!";
        }
        else{
                //get current date
                time_t now = time(0);
                tm *ltm = localtime(&now);


                int year = 1900 + ltm->tm_year;
                int month = 1 + ltm->tm_mon;
                int day = ltm->tm_mday;


                transform(name.begin(), name.end(), name.begin(),::toupper);
                transform(email.begin(), email.end(), email.begin(),::toupper);
                transform(address.begin(), address.end(), address.begin(),::toupper);
```

```
                    BorrowerFileWrite << borrowerID << "|" << name << "|" << phoneNo << "|" <<
email << "|" << address << "|" << year << "/" << month << "/" << day << "|" << 0 << "|" <<
librarianBookRecord << "\n";
                    BorrowerFileWrite.close();


                    //intialize the string
                    string S = "\t\t\tTHE BORROWER SUCCESSFULLY ADDED!!\n";
                    //Travers the given string S
                    for(int i = 0; i < S[i]; i++){
                              cout << S[i];
                              Sleep(100);
                    }


                    Sleep(2000);
                    system("cls");
                    librarianMenu();
          }


}




void deleteBorrowers(){
          int deletedCount = 0;
          string eraseTarget;


          cout << "\n\n\t |########################|-'DELETE BORROWERS'-
|#################################|\n\n";
          cout << "\t\tENTER ID/NAME\t: ";
          cin.ignore();
          getline(cin, eraseTarget);
          cout << "\n\t
|###################################################################################|\n\n"
;


          //convert inputs to lowercase for case-insensitive comparison
```

```
	transform(eraseTarget.begin(), eraseTarget.end(), eraseTarget.begin(), ::tolower);

	fstream BorrowerFile, TempFile;
	BorrowerFile.open("borrower.dat", ios::in);
	TempFile.open("temp.dat", ios::out);

	if(!BorrowerFile.is_open()){
		"\t\t\tFILE OPENINING ERROR!!!\n";
		cout << "\t\tPress any key to return Librarian Menu";
		getch();
		system("cls");
		librarianMenu();
	}

	string line;
	while(getline(BorrowerFile, line)){
		string borrowerID, name;
		istringstream iss(line);
		getline(iss, borrowerID, '|');
		getline(iss, name, '|');

		//convert details to lowercase for case-insensitive comparison
		transform(borrowerID.begin(), borrowerID.end(), borrowerID.begin(), ::tolower);
		transform(name.begin(), name.end(), name.begin(), ::tolower);

		//check whether the file matches eraseTarget
		if(eraseTarget==borrowerID||eraseTarget==name){
			deletedCount++;
		}
		else{
			TempFile << line << "\n";	//write to the temporary file if it's not the target
borrower
		}
	}

	BorrowerFile.close();
	TempFile.close();

	remove("borrower.dat");
	rename("temp.dat","borrower.dat");

	if(deletedCount>0){
		//intialize the string
		string S = "\t\t\tTHE BORROWER DELETED SUCCESSFULLY !!\n";
		//Travers the given string S
		for(int i = 0; i < S[i]; i++){
			cout << S[i];
			Sleep(100);
		}

		Sleep(2000);
		cout << "\n\n\t\tPress any key to return Librarian Menu";
		getch();
		system("cls");
		librarianMenu();
	}

	else{
		cout << "\t\tBORROWER NOT FOUND!!!\n";
		Sleep(2000);
		cout << "\n\t\tPress any key to return Librarian Menu";
		getch();
		system("cls");
		librarianMenu();
	}
}
```

```
void searchBorrowers(){
            string userInput;
            cout << "\n\n\t |#######################|-'SEARCH BORROWERS'-
|##############################|\n\n";
            cout << "\tENTER BORROWER ID/NAME/PHONE NUMBER/EMAIL/ENTERED
LIBRARIANS' ID: ";
            cin.ignore();
            getline(cin,userInput);
            cout << "\n\t
|############################################################################|\n\n"
;

            transform(userInput.begin(),userInput.end(),userInput.begin(),::tolower);    //converting user
input to lowercase <algorithm>

            //read borrower.dat file and get the line by line

    fstream BorrowerFile;
    BorrowerFile.open("borrower.dat",ios::in);

    int matchCount = 0;            //This is a counter for found borrower.

    //search matching items for the userInput
    if(BorrowerFile.is_open()){
            string line;
            string borrowerID,name,phoneNo,email,address,joinedDate,brrCount,librarian;
            while(getline(BorrowerFile,line)){
                    istringstream iss(line);
                    getline(iss,borrowerID,'|');
                    getline(iss,name,'|');
                    getline(iss,phoneNo,'|');
                    getline(iss,email,'|');
                    getline(iss,address,'|');
```

```
                getline(iss,joinedDate,'|');
                getline(iss, brrCount, '|');
                getline(iss,librarian,'|');

                //converting all the informations to lowercase

        transform(borrowerID.begin(),borrowerID.end(),borrowerID.begin(),::tolower);
                        transform(name.begin(),name.end(),name.begin(),::tolower);
                        transform(email.begin(),email.end(),email.begin(),::tolower);
                        transform(address.begin(),address.end(),address.begin(),::tolower);
                        transform(librarian.begin(),librarian.end(),librarian.begin(),::tolower);


        if(userInput==borrowerID||userInput==name||userInput==phoneNo||userInput==email||userIn
put==address||userInput==librarian){
                                matchCount++;       //increment the counter when match is found
                                cout << "\t-----------------------------------------------------------------------
-----\n";
                                cout << "\t\t\tBORROWER ID\t\t\t: " << borrowerID << endl;
                                cout << "\t\t\tNAME\t\t\t\t: " << name << endl;
                                cout << "\t\t\tPHONE NUMBER\t\t\t: " << phoneNo << endl;
                                cout << "\t\t\tE-MAIL ADDRESS\t\t\t: " << email << endl;
                                cout << "\t\t\tADDRESS\t\t\t\t: " << address << endl;
                                cout << "\t\t\tJOINED DATE\t\t\t: " << joinedDate << endl;
                                cout << "\t\t\tBORROWED COUNT\t\t\t: " << brrCount << endl;
                                cout << "\t\t\tTHE BORROWER ENTERED BY\t\t: " << librarian <<
endl;
                                cout << "\t-----------------------------------------------------------------------
-----\n";
                        }
                }

                if(matchCount==0){
                        char opt = '0';
                        cout << "\n\n\t\t\tTHIS BORROWER IS NOT FOUND!!!\n\n";
                        cout << "\n\t\tDO YOU WANT TO SEARCH ANOTHER
BORROWER(y/n)?";
                        cin >> opt;

                        if(opt=='y'||opt=='Y'){
                                system("cls");
                                searchBorrowers();
                        }

                        else if(opt=='n'||opt=='N'){
                                system("cls");
                                librarianMenu();
                        }

                        else{
                                cout << "\n\t\t\tINVALID INPUT!!!\n";
                                cout << "\n\t\tPress any key to return Librarian Menu";
                                getch();
                                system("cls");
                                librarianMenu();
                        }
                }

        }

        else{
                cout << "\t\t\tFILE IS NOT OPENED!!!\n";
                Sleep(1000);
                cout << "\n\t\t_'Press any key to return librarian Menu'_";
                getch();
                system("cls");
                librarianMenu();
        }
```

```
	BorrowerFile.close();
	cout << "\n\t\t_'Press any key to return librarian Menu'_";
	getch();
	system("cls");
	librarianMenu();
}




void updateBorrowers(){
	string updateTarget;
	cout << "\n\n\t |###############################|-'UPDATE BORROWERS'-
|################################|\n\n";
	cout << "\t\tENTER THE BORROWER ID/NAME TO UPDATE\t: ";
	cin.ignore();
	getline(cin, updateTarget);
	cout << "\n\t
|####################################################################################
###|\n\n";

	//convert the usser's input to lowercase
	transform(updateTarget.begin(), updateTarget.end(), updateTarget.begin(), ::tolower);

	//open original Borrower File for reading and TempFile for writing
	fstream BorrowerFile, TempFile;
	BorrowerFile.open("borrower.dat",ios::in);
	TempFile.open("temp.dat",ios::out);

	//check if the borrower file is successfully opened
	if(!BorrowerFile.is_open() && !TempFile.is_open()){
		cout << "\t\t\tFILE OPENINING ERROR!!!\n";
		cout << "\n\t\tPress any key to return Librarian Menu";
		getch();
```

```
        system("cls");
        librarianMenu();
}

int updatedCount = 0;
string line;    //variable that store each line of the borrower file

//read each line of the borrower file
while(getline(BorrowerFile,line)){

        //variables to store the details of the each borrower
        string borrowerID,name,phoneNo,email,address,joinedDate,brrCount,librarian;

        istringstream iss(line);
        //extract borrower details to variables
        getline(iss, borrowerID, '|');
        getline(iss, name, '|');
        getline(iss, phoneNo, '|');
        getline(iss, email, '|');
        getline(iss, address, '|');
        getline(iss, joinedDate, '|');
        getline(iss, brrCount, '|');
        getline(iss, librarian, '|');

        //convert borrower id,name to lowercase
        transform(borrowerID.begin(), borrowerID.end(), borrowerID.begin(), ::tolower);
        transform(name.begin(), name.end(), name.begin(), ::tolower);

        //check if the current borrower matches to the users' Target
        if(updateTarget==borrowerID||updateTarget==name){
                updatedCount++;
                char choice = '0';
                int opt = 0;

                while(true){
                        cout <<
"\n\t==================================================================
==================\n";
                        cout << "\t|++++++++++++++++++++++++++++++-'UPDATE
BORROWERS'-++++++++++++++++++++++++++++++++++++|\n\n";
                        cout << "\t\t\t[1].NAME                                        \n";
                        cout << "\t\t\t[2].PHONE NUMBER
\n";
                        cout << "\t\t\t[3].E-MAIL ADDRESS
\n";
                        cout << "\t\t\t[4].ADDRESS
                         ";
                        cout << "\t\t\t[5].BACK TO (SEARCH OR UPDATE BORROWER)
MENU                        \n";
                        cout <<
"\n\t==================================================================
==================\n\n";

                        //get user input
                        cout << "\t\tChoose an option: " ;
                        cin >> opt;

                        if(opt < 1 || opt > 6){
                                cout << "\n\n\t\t\tINVALID INPUT!!!\n";
                                cout << "\n\t\tPress any key to re-enter";
                                getch();
                                system("cls");
                                continue;
                        }
                        break;
                }

                if(opt == 1){
```

```
                //update name
                cout << "\n\t------------------------------------------------------------------
-----------------\n";
                cout << "\t\tCURRENT NAME\t\t: " << name <<"\n\n";
                cout << "\n\t\tNEW NAME\t\t:";
                cin.ignore();
                getline(cin,name);
                cout << "\t-------------------------------------------------------------------
---------------\n";

            }
            else if(opt == 2){
                //update phone number
                cout << "\n\t------------------------------------------------------------------
-----------------\n";
                cout << "\t\tCURRENT PHONE NUMBER \t\t: " << phoneNo
<<"\n\n";
                cout << "\n\t\tNEW PHONE NUMBER \t\t:";
                cin.ignore();
                getline(cin,phoneNo);
                cout << "\t-------------------------------------------------------------------
---------------\n";

            }
            else if(opt == 3){
                //update email
                cout << "\n\t------------------------------------------------------------------
-----------------\n";
                cout << "\t\tCURRENT E-MAIL ADDRESS\t\t: " << email <<"\n\n";
                cout << "\n\t\tNEW E-MAIL ADDRESS\t\t:";
                cin.ignore();
                getline(cin,email);
                cout << "\t-------------------------------------------------------------------
---------------\n";
            }
            else if(opt == 4){
                //update address
                cout << "\n\t--------------------------------------------------------------------
-----------------\n";
                cout << "\t\tCURRENT ADDRESS\t: " << address <<"\n\n";
                cout << "\n\t\tNEW ADDRESS\t:";
                cin.ignore();
                getline(cin,address);
                cout << "\t--------------------------------------------------------------------
---------------\n";
            }
            else if(opt == 5){
                system("cls");
                searchUpdateBorrowers();
            }

            //convert to uppercase
            transform(borrowerID.begin(), borrowerID.end(), borrowerID.begin(),
::toupper);
            transform(name.begin(), name.end(), name.begin(), ::toupper);
            transform(email.begin(), email.end(), email.begin(), ::toupper);
            transform(address.begin(), address.end(), address.begin(), ::toupper);

            //write the updated details to the temporary file
            TempFile << borrowerID << "|" << name << "|" << phoneNo << "|" << email
<< "|" << address << "|" << joinedDate << "|" << brrCount << "|" << librarian << "\n";


            }


            else{
            TempFile << line << "\n";
            }
        }

        //close both files
        BorrowerFile.flush();
BorrowerFile.close();

TempFile.flush();
TempFile.close();

//delete BookFile and rename TempFile
remove("borrower.dat");
rename("temp.dat","borrower.dat");

if(updatedCount > 0){

        char opt = '0';

        //intialize the string
        string S = "\n\t\t\tTHE BORROWER UPDATED SUCCESSFULLY!!\n";
        //Travers the given string S
        for(int i = 0; i < S[i]; i++){
                cout << S[i];
                Sleep(100);
        }

        cout << "\n\t\t\tDo you want to (SEARCH/UPDATE) again?(y/n)\t: ";
        if(opt = 'y' || opt == 'Y'){
                system("cls");
                searchUpdateBorrowers();
        }
        else{
                system("cls");
                librarianMenu();
        }
}

else{
        cout << "\t\t\tBORROWER NOT FOUND!!\n";
        cout << "\n\t\tPress any key to (SEARCH/UPDATE) Menu again";
        getch();
        system("cls");
        searchUpdateBorrowers(); }
}
```

9. Reference

https://www.w3resource.com/cpp-exercises/

https://www.w3schools.com

https://www.geeksforgeeks.org/cpp-stl-tutorial/

Thank you !