



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

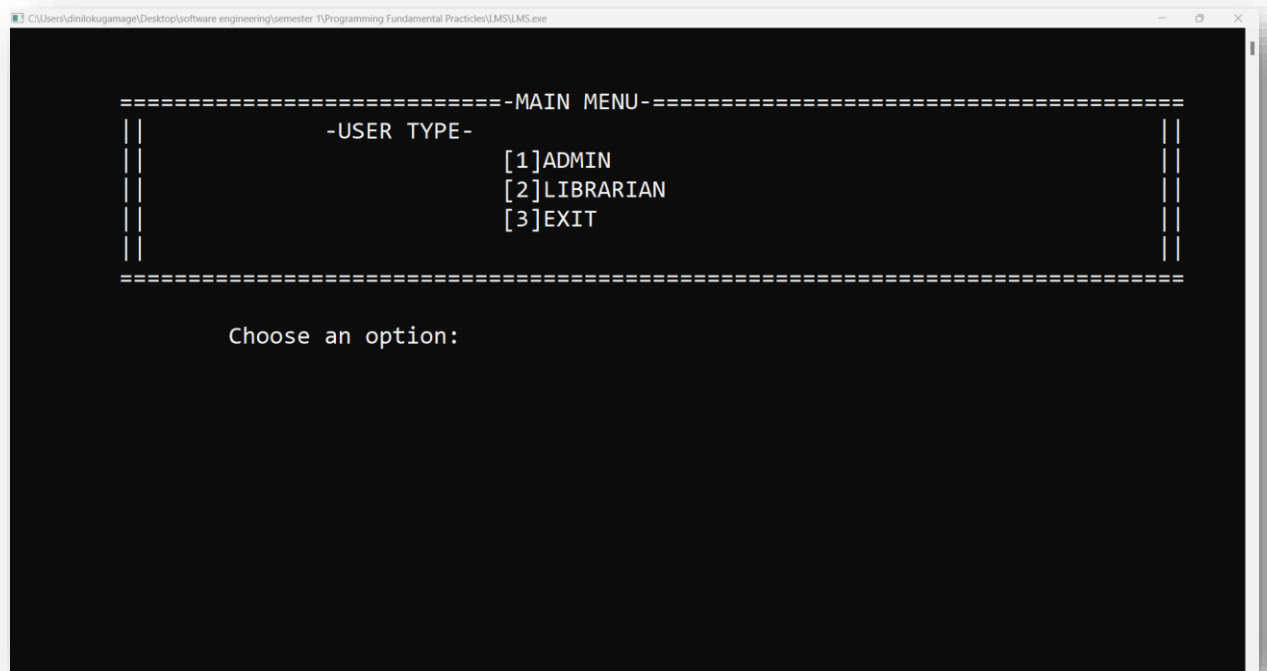
| | |
|---|------------|
| 1. Introduction | 2 |
| 2. Provide Evidence for the system implementation..... | 3 |
| 3. Test plan..... | 13 |
| 4. Test Cases..... | 14 |
| 5. Data validation and Error handling..... | 27 |
| 6. Additional features..... | 33 |
| 7. User Documentation | 36 |
| 8. Reference..... | 124 |
| 9. Code Annex (Evidence of code implementation) | 41 |

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.

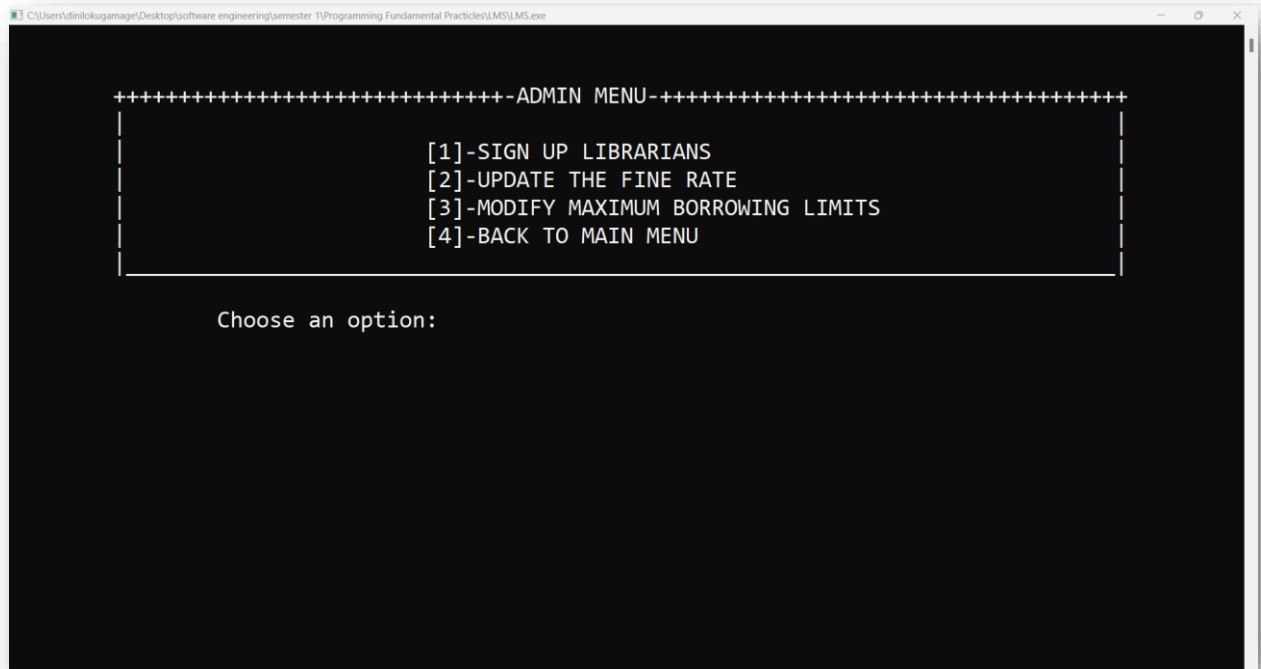
(1) Main menu



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

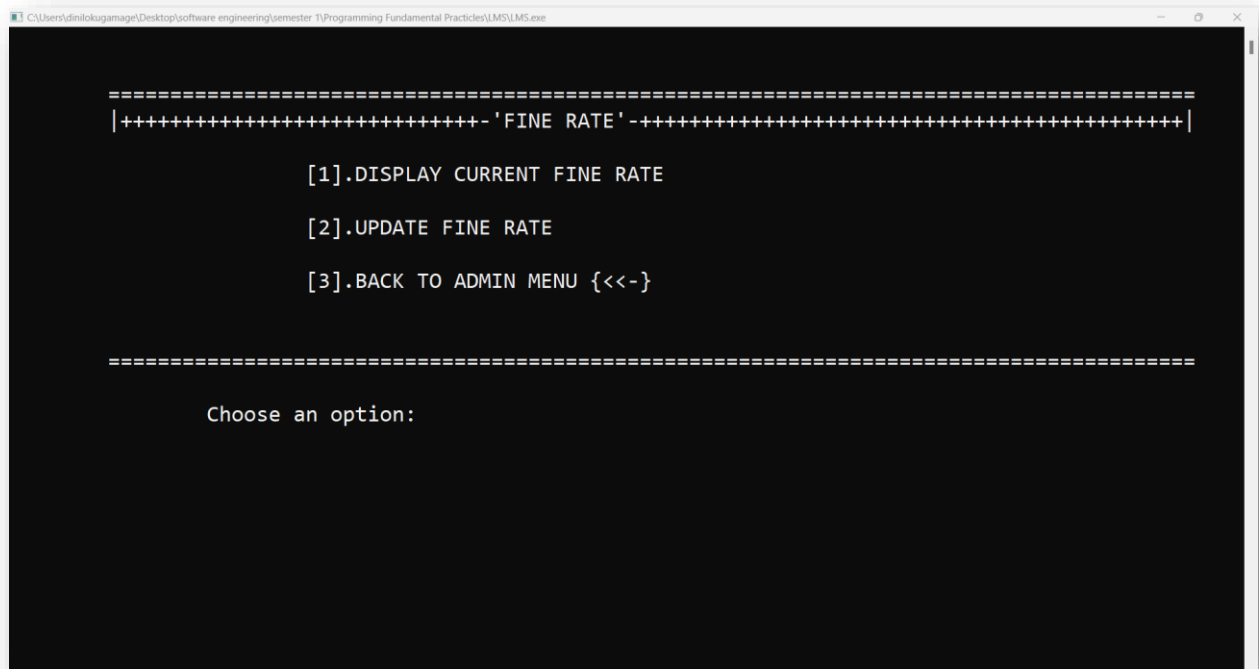


```
CS\Users\lilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practices\MS\MS.exe
+++++ADMIN MENU+++++
[1]-SIGN UP LIBRARIANS
[2]-UPDATE THE FINE RATE
[3]-MODIFY MAXIMUM BORROWING LIMITS
[4]-BACK TO MAIN MENU

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.

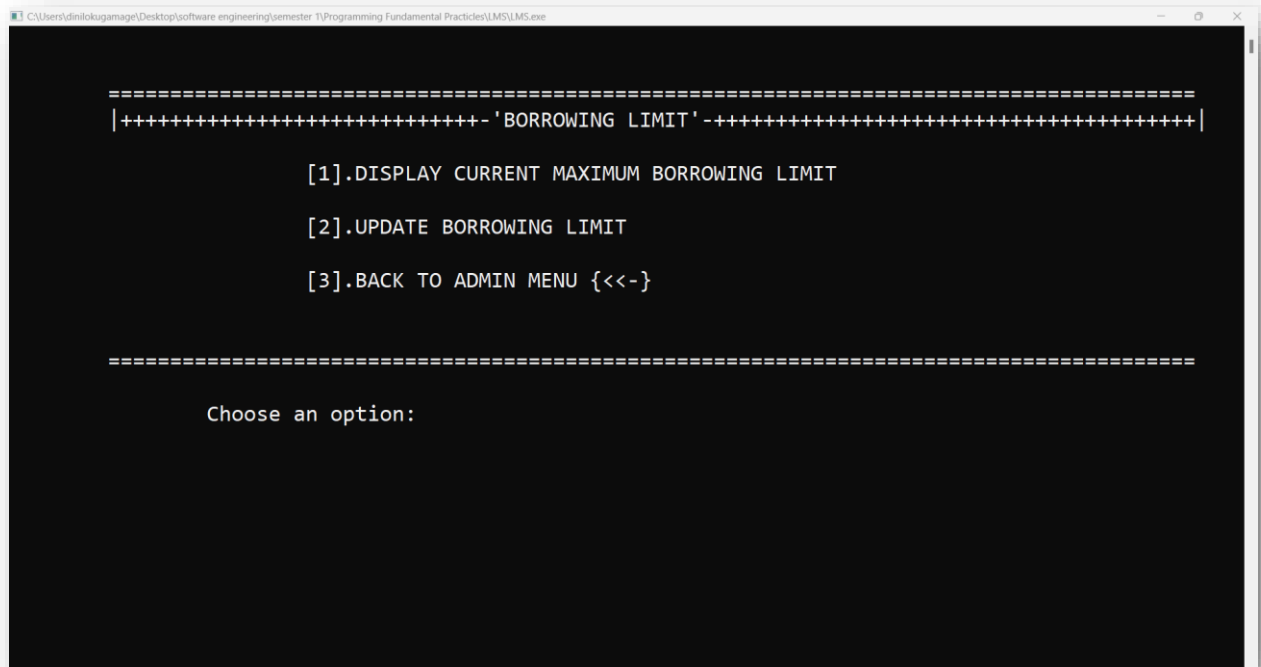
(3) Fine rate menu



```
=====
|+++++++'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) Borrowing limit menu

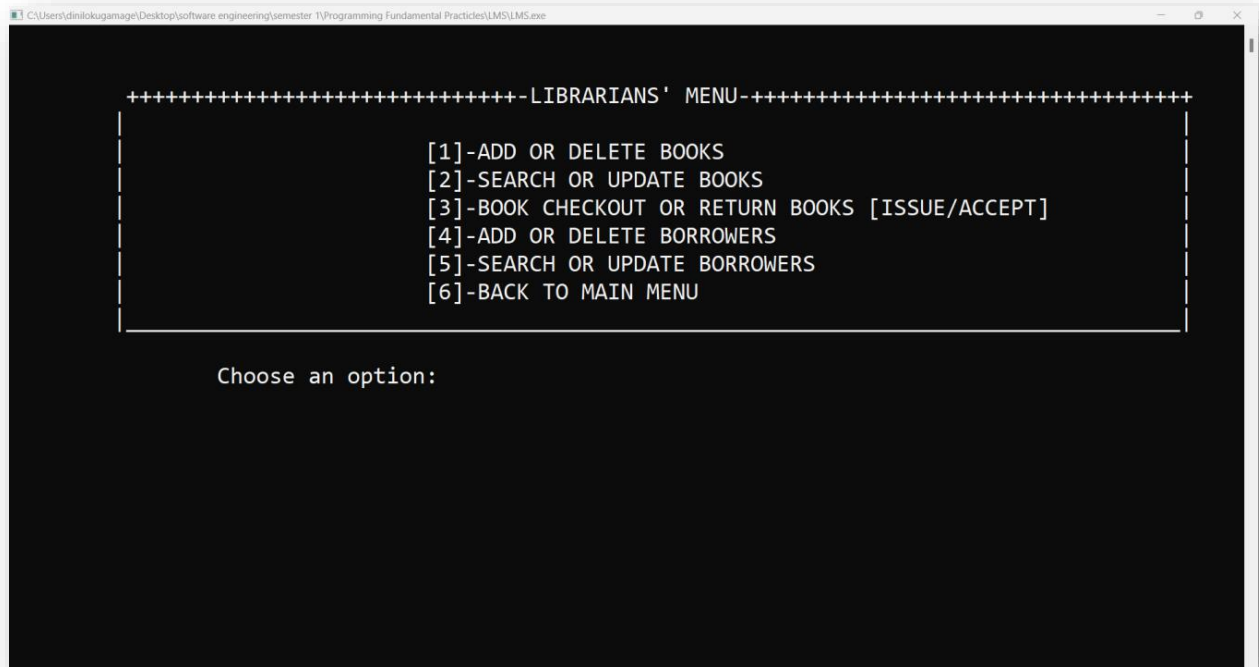


```
C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practices\LM5\LM5.exe

=====
|++++++- 'BORROWING LIMIT' -++++++|
|
|      [1].DISPLAY CURRENT MAXIMUM BORROWING LIMIT
|      [2].UPDATE BORROWING LIMIT
|      [3].BACK TO ADMIN MENU {<<-}
|
|=====
|
|      Choose an option:
|
```

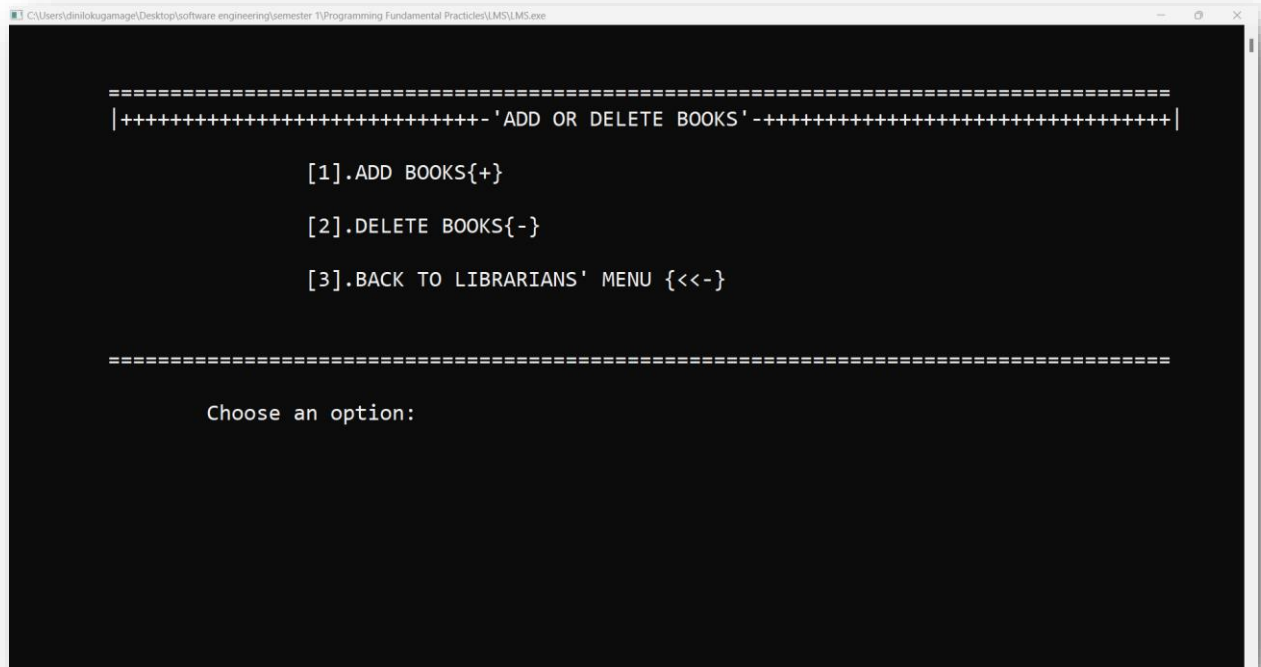
Using option 1, Librarians can quickly view the Current maximum borrowing limit. If changes are required,(option 2) the update feature allows for easy adjustments, in maximum borrowing limit.

(5) Librarian menu



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) Add or delete book menu



```

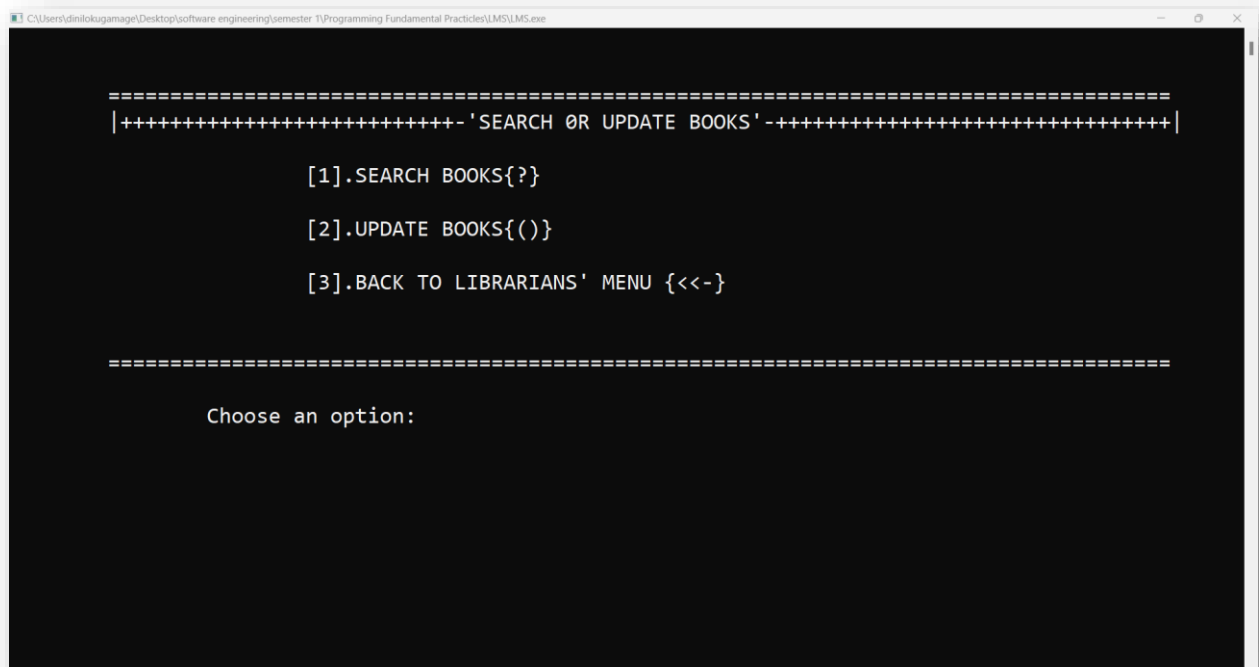
C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practices\JMS\JMS.exe

=====
|+++++++'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) Search or update books menu

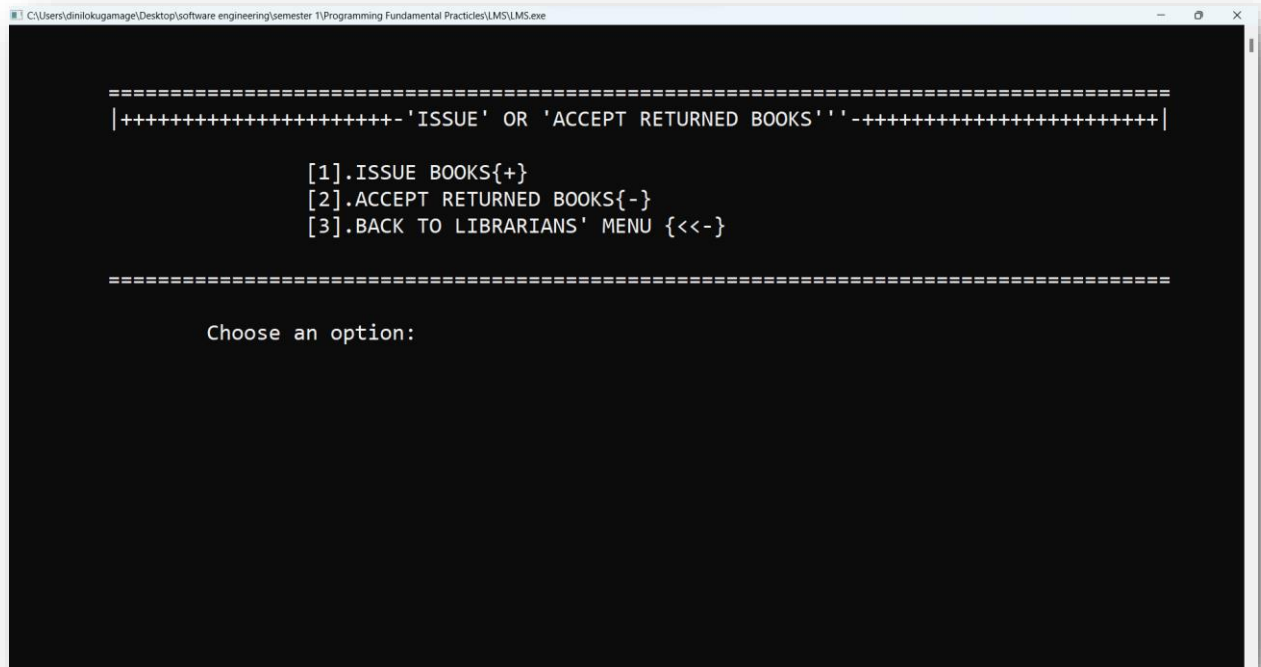


```
C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practices\LMS\LMS.exe

=====
|++++++- 'SEARCH OR 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) Issue or accept return books menu



```

C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practices\UMS\UMS.exe

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

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

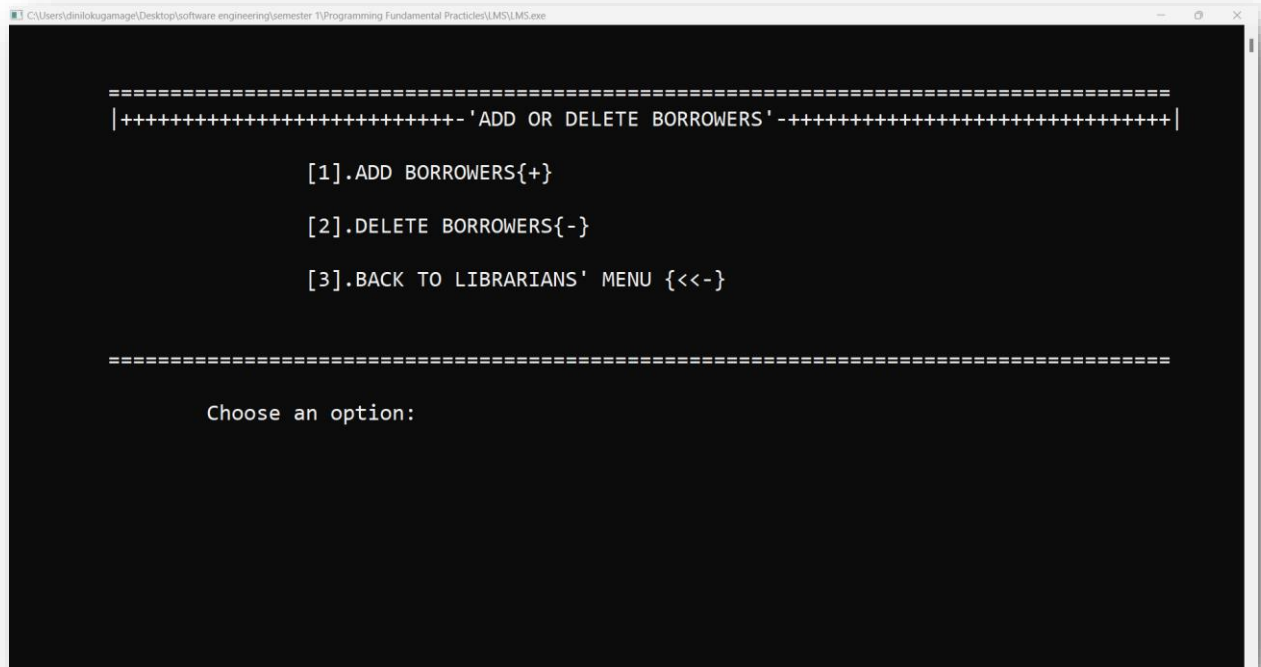
=====

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



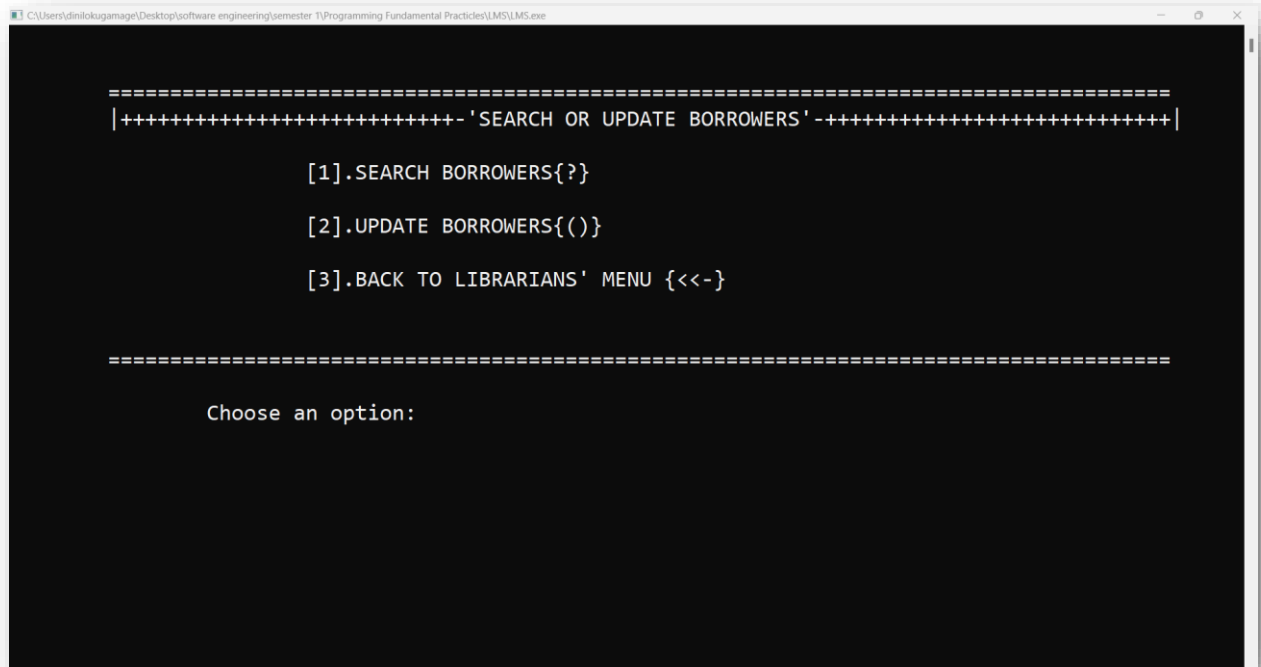
```
=====
|+++++'ADD OR DELETE BORROWERS'-----|
[1].ADD BORROWERS{+}
[2].DELETE BORROWERS{-}
[3].BACK TO LIBRARIANS' MENU {<<-}

=====

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 Practices\1MS\1MS.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 | <ul style="list-style-type: none">• 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 | <ul style="list-style-type: none">• 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 | <ul style="list-style-type: none">• Test planning• Test design• Test development• Test execution• Test evaluation |
| Test Deliverables | <ul style="list-style-type: none">• 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 \Rightarrow 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#1 233456789 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: LBOOK20 0351/ 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#1 234567891 , title – mal mama | Show “THE BOOK UPDATE D SUCCESS FULLY.” | Show “THE BOOK UPDATED SUCCESSFULLY.” | Pass |
| 02 | Enter invalid book id or title. | 07 | Book ID - LBOK#20 0351, 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 EXCEEDED THE MAXIMUM BORROWING LIMIT." | Show "BORROWER EXCEEDED 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 BORROWER ALREADY EXIST.” | Pass |
| 04 | Generate a borrower ID, which does not exist in the system. | 10 | Generated librarian id (New NIC) | Show “THIS BORROWER DOES 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 invalid 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:

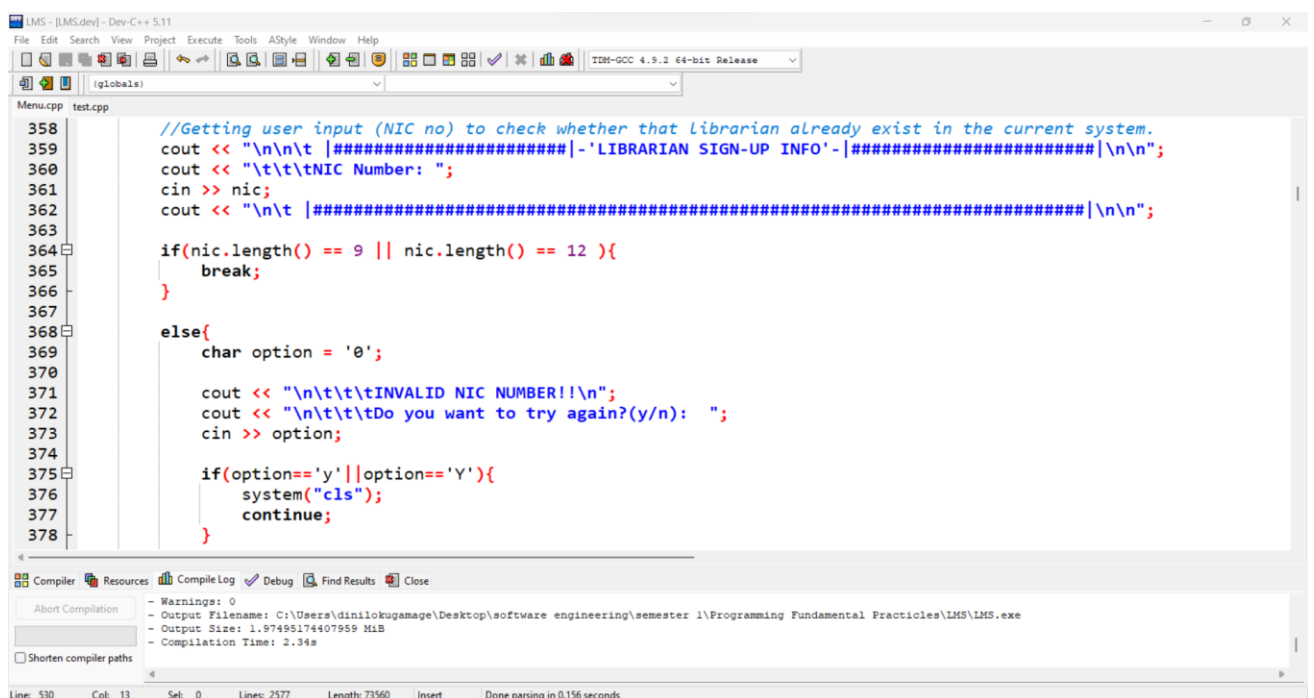
During the registration process for librarians/borrowers, the National Identity Card (NIC) number is used to create a unique librarian ID/borrower ID. The system validates that the entered NIC has either 9 or 12 characters, in line with standard NIC formats.

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.



```
LMS - [LMS.dev] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Menu.cpp test.cpp
358 //Getting user input (NIC no) to check whether that Librarian already exist in the current system.
359 cout << "\n\n\t |#####|-'LIBRARIAN SIGN-UP INFO'-|#####|\n\n";
360 cout << "\t\t\tNIC Number: ";
361 cin >> nic;
362 cout << "\n\t |#####|\n\n";
363
364 if(nic.length() == 9 || nic.length() == 12 ){
365     break;
366 }
367
368 else{
369     char option = '0';
370
371     cout << "\n\t\t\tINVALID NIC NUMBER!!\n";
372     cout << "\n\t\t\tDo you want to try again?(y/n): ";
373     cin >> option;
374
375     if(option=='y'||option=='Y'){
376         system("cls");
377         continue;
378     }
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
279
```

2. Case Sensitivity and Data Matching:

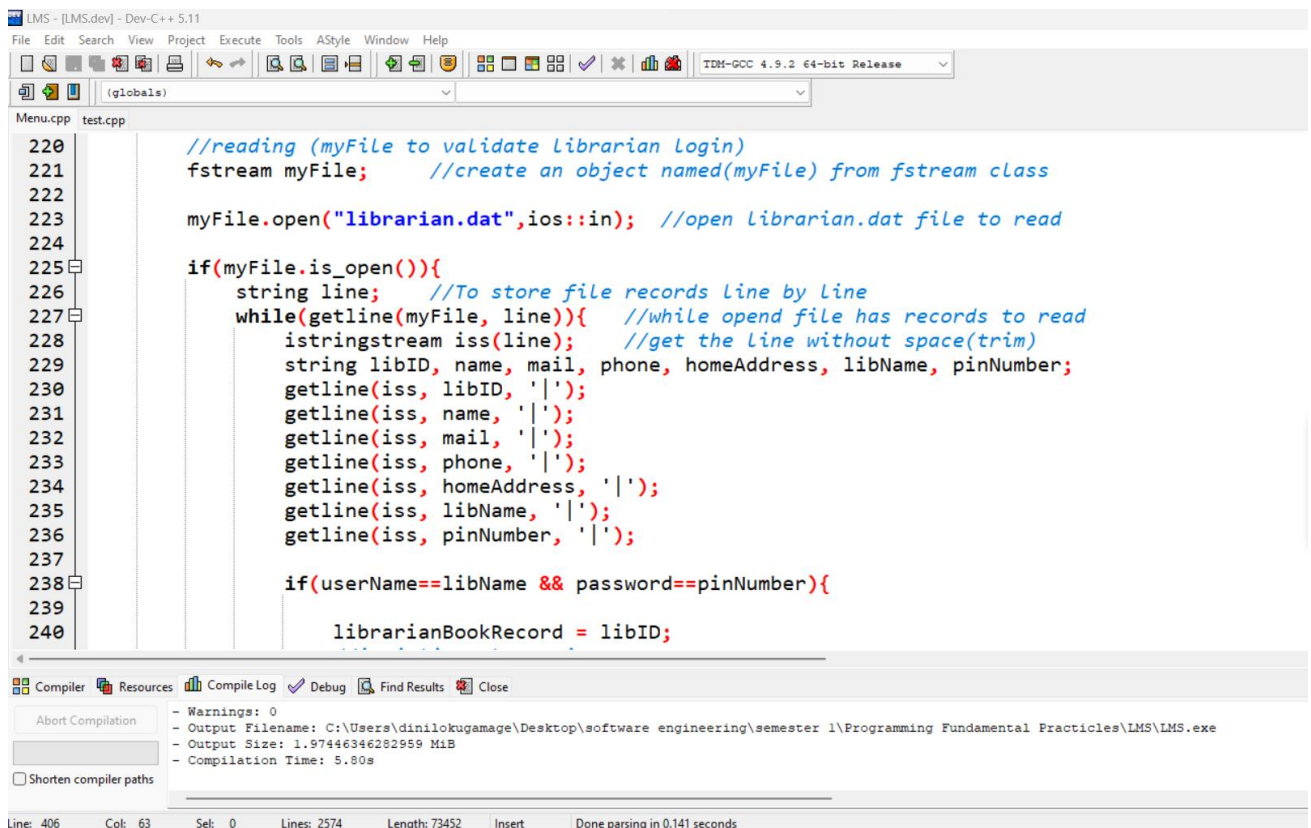
a. Security validation

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
(globals)
Menu.cpp test.cpp

220 //reading (myFile to validate Librarian Login)
221 fstream myFile; //create an object named(myFile) from fstream class
222
223 myFile.open("librarian.dat",ios::in); //open librarian.dat file to read
224
225 if(myFile.is_open()){
226     string line; //To store file records Line by Line
227     while(getline(myFile, line)){ //while open file has records to read
228         istringstream iss(line); //get the line without space(trim)
229         string libID, name, mail, phone, homeAddress, libName, pinNumber;
230         getline(iss, libID, '|');
231         getline(iss, name, '|');
232         getline(iss, mail, '|');
233         getline(iss, phone, '|');
234         getline(iss, homeAddress, '|');
235         getline(iss, libName, '|');
236         getline(iss, pinNumber, '|');
237
238         if(userName==libName && password==pinNumber){
239
240             librarianBookRecord = libID;
241         }
242     }
243 }

Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
- 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
Shorten compiler paths

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.

```

400
401     myFile.open("librarian.dat",ios::in);    //open Librarian.dat file to read
402
403     if(myFile.is_open()){
404         string line;        //To store file records line by line
405         while(getline(myFile, line)){ //while open file has records to read
406             istringstream ss(line);    //get the line without space(trim)
407             string libID;
408             getline(ss, libID, '|');
409
410             if(id==libID){
411                 cout << "\t\t 'LIBRARIAN ALREADY EXIST' \n\n";
412                 myFile.close();
413                 cout << "\n\t\t 'Press any key to return Admin Menu' _";
414                 getch();
415                 system("cls");
416                 adminMenu();
417             }
418         }
419     }
420

```

Compiler Output:

```

- 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

```

1747     while(getline(BookFile, line)){
1748         string bookID, title;
1749         istringstream iss(line);
1750         getline(iss, bookID, '|');
1751         getline(iss, title, '|');
1752
1753         //convert details to lowercase for case-insensitive comparison
1754         transform(bookID.begin(), bookID.end(), bookID.begin(), ::tolower);
1755         transform(title.begin(), title.end(), title.begin(), ::tolower);
1756
1757         //check whether the file matches eraseTarget
1758         if(eraseTarget==bookID||eraseTarget==title){
1759             deletedCount++;
1760         }
1761         else{
1762             TempFile << line << "\n"; //write to the temporary file if it's not the target book
1763         }
1764     }
1765
1766     BookFile.close();
1767     TempFile.close();

```

Compiler Output:

```

- 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

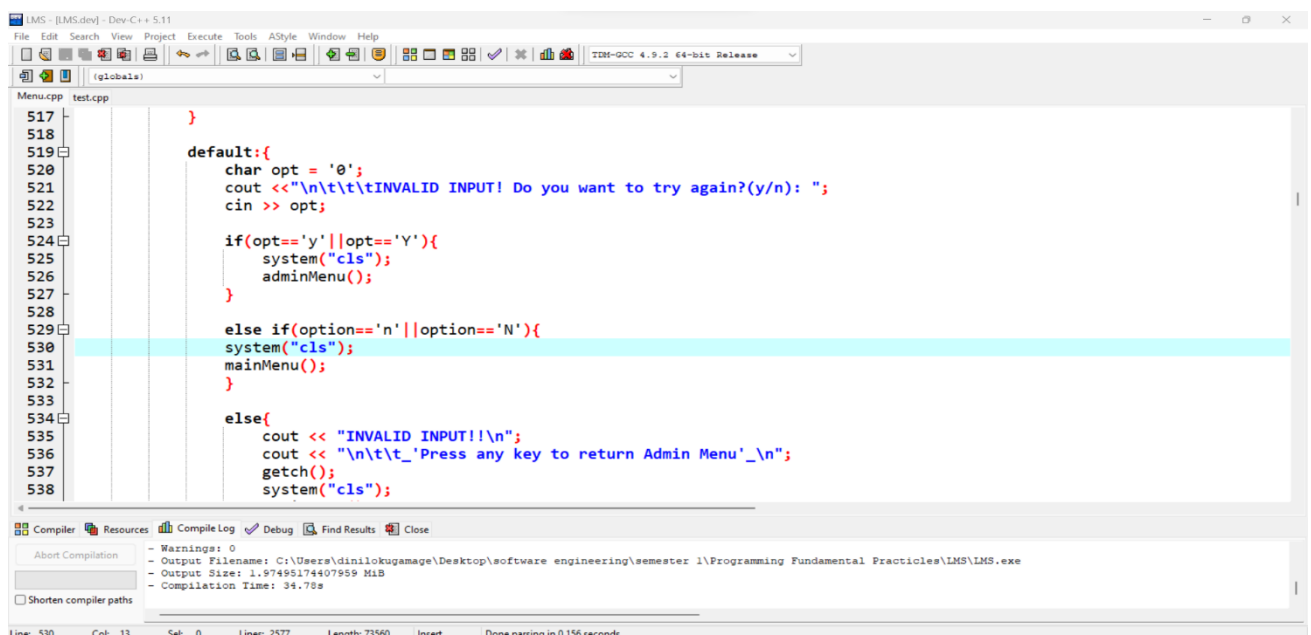
-Delete books

(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.



The screenshot shows a C++ IDE with a file named 'Menu.cpp'. The code implements a menu system with error handling. It uses a 'default' case in a switch statement to handle invalid input. When an invalid input is detected, it prints a message: "INVALID INPUT! Do you want to try again?(y/n): ". If the user enters 'y' or 'Y', it calls 'adminMenu()'. If the user enters 'n' or 'N', it calls 'mainMenu()'. For any other input, it prints "INVALID INPUT!!\n" and "Press any key to return Admin Menu'\n", then uses 'getch()' to wait for a key press before clearing the screen with 'system("cls")'.

```
517 }
518
519 default:{
520     char opt = '0';
521     cout << "\n\t\t\t\t\tINVALID INPUT! Do you want to try again?(y/n): ";
522     cin >> opt;
523
524     if(opt=='y' || opt=='Y'){
525         system("cls");
526         adminMenu();
527     }
528
529     else if(option=='n' || option=='N'){
530         system("cls");
531         mainMenu();
532     }
533
534     else{
535         cout << "INVALID INPUT!!\n";
536         cout << "\n\t\t\t\t\tPress any key to return Admin Menu'\n";
537         getch();
538         system("cls");
539     }
```

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 as 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.


```

721 void modifyBorrowingLimit(){
722     fstream outMaxBorrowFile;
723
724     outMaxBorrowFile.open("borrowingLimit.dat",ios::out);
725
726     if(!outMaxBorrowFile){
727         cout << "FILE OPENED FAIL!!!\n";
728         Sleep(1000);
729         cout << "\n\t\t\t\t\t'Press any key to return Admin Menu'\n";
730         getch();
731         system("cls");
732         adminMenu();
733     }
734
735     else{
736         int borrowingLimit = 0;
737
738         while(true){
739             cout << "\n\t\t\t\t\tEnter new borrowing limit(Books): ";
740             cin >> borrowingLimit;
741             if(borrowingLimit < 0){

```

Compiler Resources Compile Log Debug Find Results Close

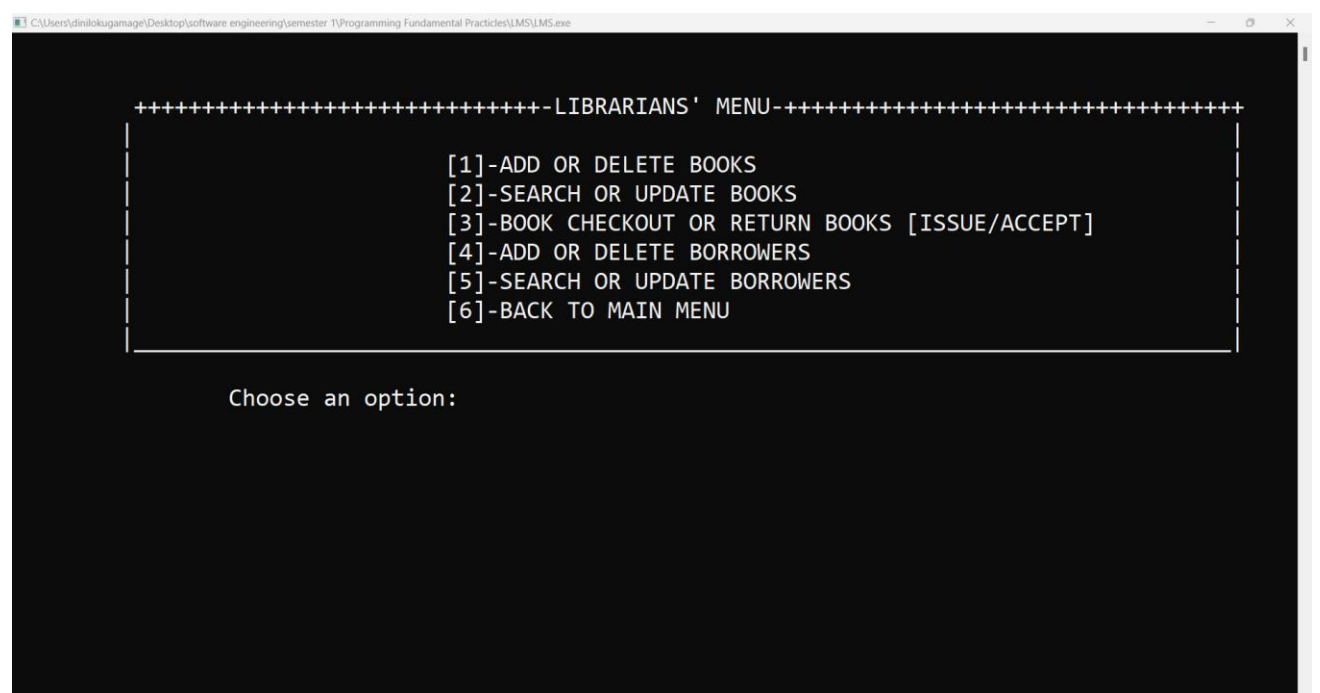
Warnings: 0
 Output Filename: C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practices\LMS\LMS.exe
 Output Size: 1.97495174407959 MiB
 Compilation Time: 34.78s

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

Durable System Navigation:

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

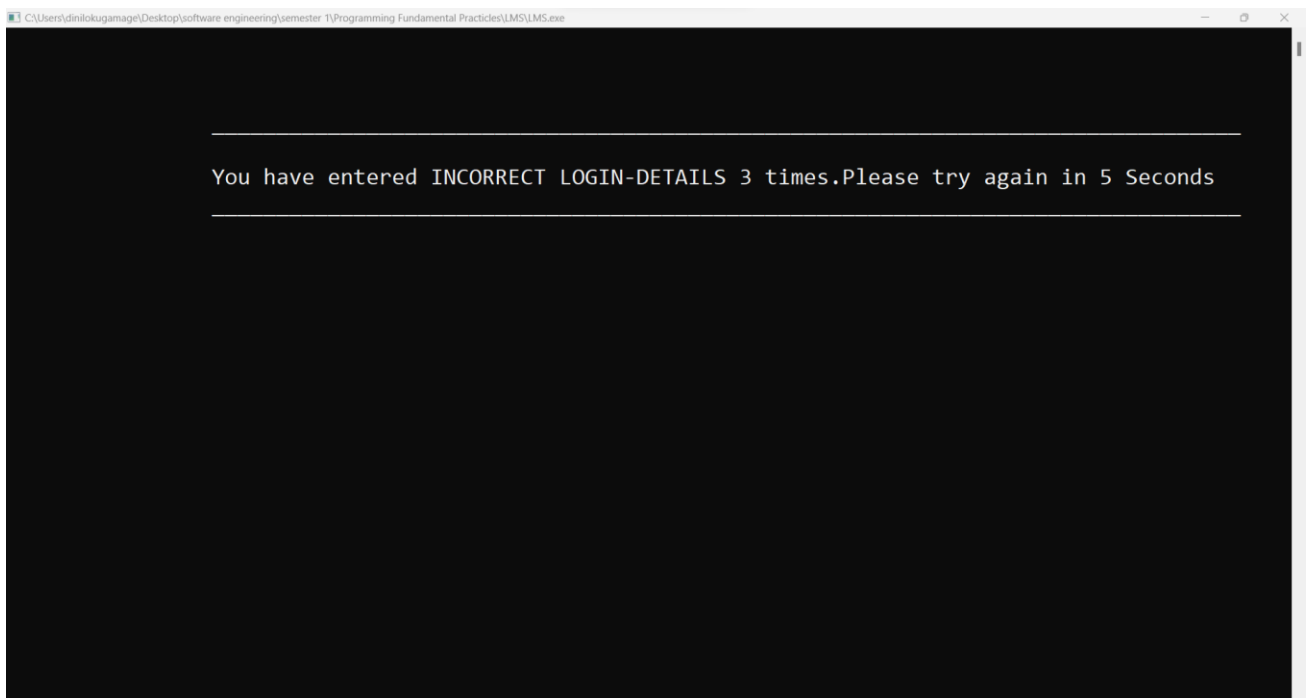
Retries and System Flow: To prevent users from being left stranded after an error, the system frequently gives users the choice to attempt again or to return to the prior state. This strategy makes sure that users are always aware of their position inside the system and what to do next.ex: every menu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



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.



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.

```

=====
|++++++-'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.

```

Menu.cpp test.cpp
1801
1802 void searchBooks(){
1803     string userInput = " ";
1804     cout << "\n\n\t |#####-'SEARCH BOOKS'#####\n\n";
1805     cout << "\tENTER ID/TITLE/AUTHOR/PUBLISHER/GENRE/AVAILABILITY/ENTERED LIBRARIAN: ";
1806     cin.ignore();
1807     getline(cin,userInput);
1808     cout << "\n\t |#####\n\n";
1809
1810     transform(userInput.begin(),userInput.end(),userInput.begin(),::tolower); //converting user input to lowercase <a
1811
1812     //read books.dat file and get the line by line
1813
1814     fstream BookFile;
1815     BookFile.open("books.dat",ios::in);
1816
1817     int matchCount = 0; //This is a counter for found book.
1818
1819     //search matching items for the userInput
1820     if(BookFile.is_open()){
1821         string line;

```

Compiler Resources Compile Log Debug Find Results Close

About Compilation

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

Shorten compiler paths

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

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

-----
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.

The screenshot shows a C++ IDE with a file named `test.cpp` open. The code is a menu-driven program for a library management system. It includes headers for `iostream`, `string`, `vector`, and `fstream`. It defines a `Book` struct and a `Librarian` struct. The `main` function contains a menu with options to add book, add borrower, issue book, return book, and search. The `addBook` function is partially visible, showing the creation of a `Book` object and its addition to a `vector`. The `addBorrower` function is also partially visible, showing the creation of a `Librarian` object and its addition to a `vector`. The `issueBook` function is shown, which checks if a book is available and issues it to a borrower. The `returnBook` function is also shown, which updates the book's status to 'AVAILABLE'. The `search` function is shown, which searches for a book by title and displays its details. The compiler output window at the bottom shows the compilation of `test.cpp` using `g++` 4.9.2, with no warnings and a compilation time of 0.16s.

```
1270
1271
1272
1273 er);
1274
1275
1276 oupper);
1277
1278 << publisher << "|" << genre << "|" << price << "|" << "UNAVAILABLE" << "|" << addedDate << "|" << librarian << "\n";
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
```

Compiler Output:

```
- Warnings: 0
- Output Filename: C:\Users\dinilokugamage\Desktop\software engineering\semester 1\Programming Fundamental Practices\LMS\LMS.exe
- Output Size: 1.9746346282959 MiB
- Compilation Time: 0.16s
```

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
```


[illegible]


```

        cout << "\t |%%%%%%%%%%-'ADMIN LOGIN'-
|%%%%%%%%%\n\n";

        cout << "\t\t [*]ENTER USERNAME: ";

        cin >> userName;

        cout << "\t\t [*]ENTER PASSWORD: " ;

        cin >> password;

        cout << "\n\t
|%%%%%%%%%%
%%%%%%%%%\n\n";

```

```

if(userName==adminUserName && password==adminPassword){

```

```

    //intialize the string

```

```

    string S = "\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\tINCORRECT USERNAME OR PASSWORD!!\n\n";

```

```

    cout << "\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\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 opened 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\tINCORRECT USERNAME OR PASSWORD!!\n\n";
cout << "\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 << "\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\tINVALID NIC NUMBER!!\n";
cout << "\n\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 openend 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\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=====
=====
\t";

        cout << "\t++++++-'FINE RATE'-
+++++\n\n";

        cout << "\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\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\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\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;
```

```
            }
```

```
        }
```

```
file
```

```
        outMaxBorrowFile << borrowingLimit ;    //write maximum borrowing limit in the
```

```
        outMaxBorrowFile.close();
```

```
//initialize the string
string S = "\n\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\t\t\t\t|\n";
        cout << "\t\t\t\t[1]-ADD OR DELETE BOOKS\t\t\t\t\t|\n";
        cout << "\t\t\t\t[2]-SEARCH OR UPDATE BOOKS\t\t\t\t\t|\n";
        cout << "\t\t\t\t[3]-BOOK CHECKOUT OR RETURN BOOKS\t\t\t\t\t|\n";
        [ISSUE/ACCEPT] |\n";
        cout << "\t\t\t\t[4]-ADD OR DELETE BORROWERS\t\t\t\t\t|\n";
        cout << "\t\t\t\t[5]-SEARCH OR UPDATE BORROWERS\t\t\t\t\t|\n";
        cout << "\t\t\t\t[6]-BACK TO MAIN MENU\t\t\t\t\t|\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[1].ADD BOOKS{+}                \n";
    cout << "\t\t[2].DELETE BOOKS{-}                \n";
    cout << "\t\t[3].BACK TO LIBRARIANS' MENU {<<-}    \n";

    cout <<
    "\n\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 OR 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\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\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\tBOOK ID\t\t: ";

```



```

cin >> bookID;
cout << "\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 << "\n\tYOU CAN'T BORROW [BOOK DOES NOT EXIST OR
BOOK ALREADY ISSUED!!!]";
            cout << "\n\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\tINVALID INPUT!!\n";
                cout << "\n\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 << "\n\tFILE OPENED FAIL!!!\n";
    Sleep(1000);
    cout << "\n\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\tBOOK 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\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\tBORROWER ID\t\t: " << borrowerID << endl;
    cout << "\t\tBOOK ID\t\t\t: " << bookID << endl;
    cout << "\t\tBOOK TITLE\t\t: " << bookTitle << endl;
    cout << "\t\tISSUE DATE\t\t: " << year << "/" << month << "/" << day << endl;
    cout << "\t\tDUE DATE\t\t\t: " << (1900+dueDate.tm_year) << "/" << (1+dueDate.tm_mon)
<< "/" << (dueDate.tm_mday) << "\n";
    cout << "\t-----\n";

    //intialize the string
    string S = "\n\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 OPENING 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[1].ADD BORROWERS{+}                \\n";
    cout << "\t\t[2].DELETE BORROWERS{-}                \\n";
    cout << "\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\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[1].SEARCH BORROWERS{?}                \n";
        cout << "\t\t[2].UPDATE BORROWERS{()}                \n";
        cout << "\t\t[3].BACK TO LIBRARIANS' MENU {<<-}        \n";

        cout <<
        "\n\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\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 << "\n\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();

```



```

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

string line;
while(getline(BookFile, line)){
    string bookID, title;
    istream 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\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\t: " << publisher << endl;
        cout << "t\t\tGENRE\t\t\t: " << genre << endl;

```

```

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

if(matchCount==0){
    char opt = '0';
    cout << "\n\n\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\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 OPENING 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[1].TITLE                \\n";
        cout << "\t\t[2].AUTHOR
\\n";

        cout << "\t\t[3].GENRE                \\n";
        cout << "\t\t[4].AVAILABILITY
\\n";

        cout << "\t\t[5].BACK TO (SEARCH OR UPDATE BOOKS)
\\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\\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\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\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\t-----\n";

}

else if(opt == 4){
    //update availability
    cout << "\n\t-----\n";

    cout << "\t\tCURRENT AVAILABILITY\t\t: " << availability << "\n\n";
    cout << "\n\t\tNEW AVAILABILITY\t\t: ";
    cin.ignore();
    getline(cin, availability);
    cout << "\t\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 OPENING 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\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||userInput==address||userInput==librarian){
    matchCount++; //increment the counter when match is found
    cout << "\t-----\n";

    cout << "\t\tBORROWER ID\t\t: " << borrowerID << endl;
    cout << "\t\tNAME\t\t\t: " << name << endl;
    cout << "\t\tPHONE NUMBER\t\t: " << phoneNo << endl;
    cout << "\t\tE-MAIL ADDRESS\t\t: " << email << endl;
    cout << "\t\tADDRESS\t\t\t: " << address << endl;
    cout << "\t\tJOINED DATE\t\t: " << joinedDate << endl;
    cout << "\t\tBORROWED COUNT\t\t: " << brrCount << endl;
    cout << "\t\tTHE BORROWER ENTERED BY\t: " << librarian <<
endl;

    cout << "\t-----\n";

    }

}

if(matchCount==0){
    char opt = '0';
    cout << "\n\n\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\tINVALID INPUT!!!\n";
            cout << "\n\t\tPress any key to return Librarian Menu";
            getch();
            system("cls");
            librarianMenu();
        }
    }

    else{
        cout << "\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\t_Press 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[1].NAME                                \\n";
    cout << "\t\t[2].PHONE NUMBER
\\n";

    cout << "\t\t[3].E-MAIL ADDRESS
\\n";

    cout << "\t\t[4].ADDRESS
";

    cout << "\t\t[5].BACK TO (SEARCH OR UPDATE BORROWER)
\\n";
MENU
    cout <<
"\n\t=====
=====\\n\\n";

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

    if(opt < 1 || opt > 6){
        cout << "\\n\\n\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\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\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\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\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\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\tBORROWER NOT FOUND!!\n";  
cout << "\n\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 !