Green University of Bangladesh

Department of Computer Science and Engineering (CSE)

Semester: (Sprng, Year: 2024), B.Sc. in CSE (Day)

Library Management System Application With Interface.

Course Title : Operating System Lab

Course Code : CSE-310

Section : 213 D5

Students Details

Name	**ID**
MD Dulal Hossain	213902116

Submission Date: 00-07-2024

Course Teacher's Name: MD Solaiman Mia

[For teachers use only: Don't write anything inside this box]

Lab Project Status

Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Overview

This Bash script implements a basic Library Management System designed by Dulal. It enables users to add, find, edit, remove, and view book records through a command-line interface. The script utilizes functions to perform various tasks, such as adding records, searching for books, and managing user input. Additionally, it includes error handling, confirmation prompts, and a structured menu layout for user interaction. Overall, it offers a simple yet functional solution for managing book records in a library setting.

1.2 Motivation

This Library Management System script was motivated by the need for a simple, command-line solution to manage book records efficiently. Designed by Dulal, it aims to streamline tasks such as adding, finding, editing, removing, and viewing book entries. The script offers a user-friendly interface, incorporating confirmation prompts and error handling to enhance the user experience. It provides a convenient tool for organizing and maintaining book records effectively.

1.3 Problem Definition

1.3.1 Problem Statement

The project aims to develop a simple command-line-based Library Management System (LMS) using Bash scripting. The system should allow users to perform operations such as adding, finding, editing, removing, and viewing book records. Additionally, it should include features like confirmation prompts and error handling to enhance user experience and efficiency in managing the library's book inventory.

1.3.2 Complex Engineering Problem

The complex engineering problem addressed by this project lies in designing an efficient command-line-based Library Management System (LMS) using Bash scripting. Key challenges include implementing functionalities for adding, finding, editing, removing, and viewing book records while ensuring data integrity and user-friendliness. Additionally, the script needs to handle various user inputs, manage file operations securely, and incorporate features like confirmation prompts and error handling for smooth operation in a multi-user environment.

Table 1.1: Complex Engineering Problem Steps

Name of the P Attributess	Explain how to address
P1: Depth of knowledge required	The project necessitates grasping Bash scripting fundamentals: variables, functions, conditionals, loops, and file manipulation. Knowledge of command-line interfaces and basic text processing is also vital.
P2: Range of conflicting requirements	Balancing conflicting requirements, like user-friendly interfaces and efficient file manipulation, is crucial for this library system managed via Bash scripting. Conflicts might arise between simplicity and robust error handling or extensive features and Bash's limitations.
P3: Depth of analysis required	For this project, comprehending user needs for a library management system, defining key functions (add, find, edit, remove books), and efficiently scripting them in Bash are vital. Analyzing user interactions, error handling, and file manipulation are crucial for effective implementation.
P4: Familiarity of issues	Understanding basic Bash scripting, file manipulation, and user input handling are essential for this project. Interactive menu creation, text file management, and CRUD operations are crucial. Basic knowledge of regular expressions is advantageous.

1.4 Design Goals/Objectives

The primary objective of this library management system is to provide a user-friendly interface for managing book records efficiently. Key design goals include enabling users to add, find, edit, and remove book entries seamlessly. The system aims to maintain organized records of book categories, titles, and authors. Additionally, it emphasizes ease of use, with clear menu options and intuitive prompts, ensuring smooth navigation and operation for both administrators and users.

1.5 Application

This Bash script implements a user-friendly library management system. It allows users to perform various tasks such as adding new book records, finding books by category or title, editing existing book records, removing books, and viewing the entire list of books. Each function is clearly defined and accessible through a menu-driven interface. The application aims to simplify the process of managing a library collection by providing easy-to-use functionalities for cataloging, searching, and maintaining book records.

Chapter 2

Design/Implementation of the Project

2.1 Introduction

This Bash script presents a Library Management System developed by Dulal-213902116, aimed at streamlining the organization of book records. With a straightforward command-line interface, users can perform various tasks such as adding new book entries, searching for books by category or title, editing existing records, removing books, and viewing the complete list of books. The system's design emphasizes simplicity and efficiency, making it suitable for managing library collections of any size. Whether for personal or professional use, this script offers a convenient solution for efficiently managing and accessing library resources. Its intuitive menu-driven approach simplifies the process of managing book records, enhancing overall library management efficiency.

2.2 Project Details

This Library Management System, designed by Dulal-213902116, is a command-line tool that streamlines the organization of book records. The system offers various functionalities accessible through a user-friendly menu interface. Users can add new book entries by providing details such as category, title, and author name. Additionally, they can search for books by category or title, edit existing records, remove books from the database, and view the complete list of books.

The script employs Bash scripting to manage the backend operations efficiently, utilizing global variables for configuration and temporary files for data manipulation. It incorporates utility functions for user interaction, including input validation and confirmation prompts. With its clear and structured design, this system caters to both individual users and small libraries, offering a convenient solution for maintaining book collections.

Overall, this Library Management System provides a practical and effective way to organize and manage book records, enhancing accessibility and efficiency in book management tasks. It serves as a versatile tool for individuals and organizations seeking a straightforward solution for book record management.

2.3 The workflow

The workflow for this Library Management System project is structured to efficiently manage book records through a series of user-friendly interactions. Here's the workflow described :

1. **Main Menu:** The script presents a main menu with options to perform various actions, including adding, finding, editing, removing, and viewing books.
2. **Add Books:** When selecting the option to add books, users are prompted to input details such as category, title, and author name. They are then asked to confirm the addition of the record.
3. **Find Books:** Users can search for books by entering a keyword or phrase, which is used to search the book records. Matching books are displayed, or a message indicates if no matches are found.
4. **Edit Books:** Editing allows users to modify existing book records. They are presented with the list of books, prompted to choose the book they want to edit, and then asked to input the updated details.
5. **Remove Books:** Users can remove books from the database by specifying the title of the book they want to delete. If the book is found, it is removed; otherwise, an error message is displayed.
6. **View Books:** This option displays the complete list of books stored in the system.
7. **Exit:** Users can choose to quit the program, ending the workflow.
8. **Cleanup:** Finally, the script removes any temporary files created during execution before exiting.

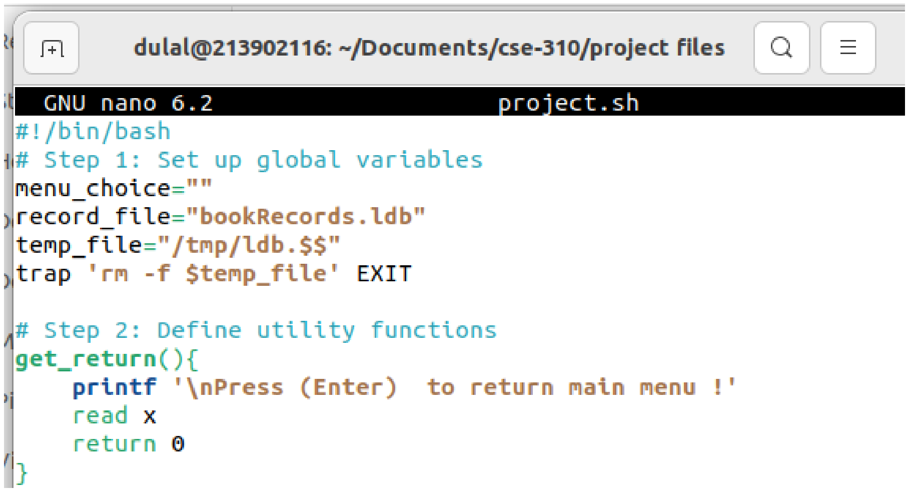
2.4 Tools and libraries

1. **Bash Shell Scripting:** The project is developed using Bash scripting language for creating a command-line interface for the library management system.
2. **Text Editors:** Text editors such as Vim, Nano, or Sublime Text were used for writing and editing the Bash script files.
3. **Linux Environment:** The project is designed to run in a Linux environment, utilizing the shell for executing Bash scripts.
4. **Basic Unix Utilities:** Standard Unix utilities such as grep, wc, printf, touch, mv, rm, and clear are used within the script for various operations like text processing, file manipulation, and user interaction.
5. **Trap Command:** The trap command is employed to handle interruptions like Ctrl+C gracefully.

   These tools and libraries collectively provide the necessary functionality for implementing the library management system in a Unix-based environment.

2.5 Implementation / Programming codes

2.5.1 Code_portion_Global



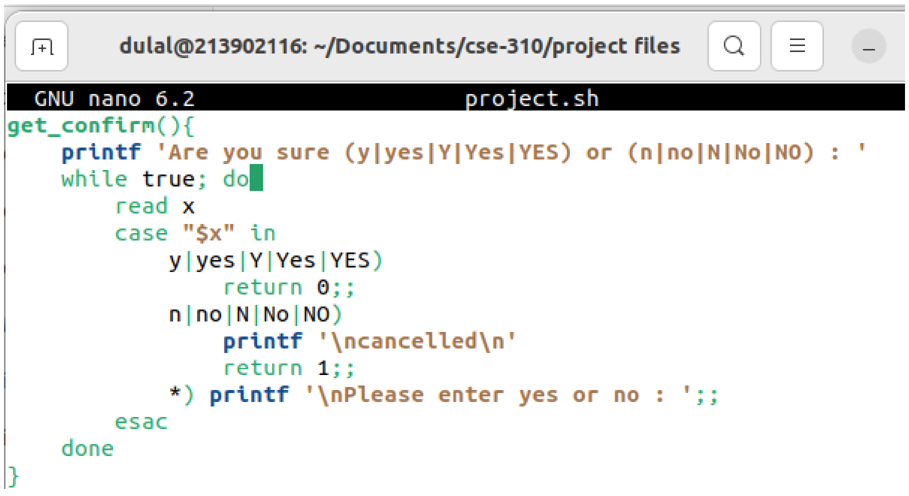
```
#!/bin/bash
# Step 1: Set up global variables
menu_choice=""
record_file="bookRecords.ldb"
temp_file="/tmp/ldb.$$"
trap 'rm -f $temp_file' EXIT

# Step 2: Define utility functions
get_return(){
    printf '\nPress (Enter)  to return main menu !'
    read x
    return 0
}
```

Figure 2.1: Code for Global part.

Step 1: Initialize the following global variables: menu-choice: Stores the user's menu choice. record-file: Specifies the file where book records are stored. temp-file: Specifies the temporary file used for various operations. Set up a trap using the trap command. The trap removes the temporary file (temp-file) upon script termination.

Step 2: get-return() Function: This function prints a message prompting user to press Enter to return to main menu. It then waits for user to press Enter and returns 0, indicating successful execution.

2.5.2 Code_portion_Get-Confirm

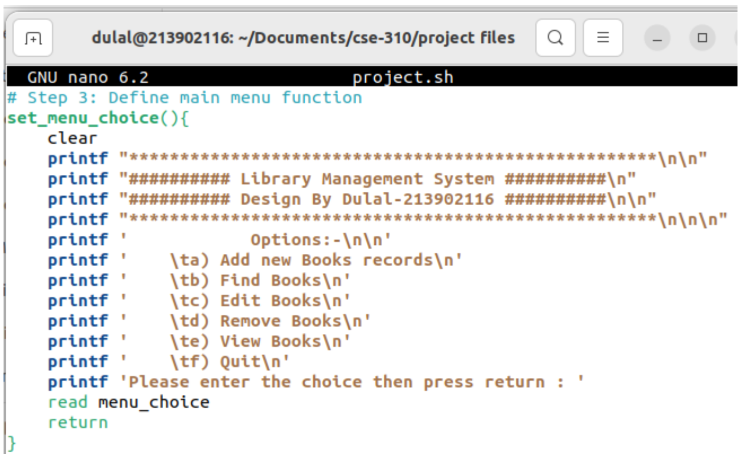


```
get_confirm(){
    printf 'Are you sure (y|yes|Y|Yes|YES) or (n|no|N|No|NO) : '
    while true; do
        read x
        case "$x" in
            y|yes|Y|Yes|YES)
                return 0;;
            n|no|N|No|NO)
                printf '\ncancelled\n'
                return 1;;
            *) printf '\nPlease enter yes or no : ';;
        esac
    done
}
```

Figure 2.2: Code for Get-Confirm part.

Prompt user for confirmation with options "yes" or "no". Continuously read input until valid response. If input matches "yes" variants, return 0 for confirmation. If input matches "no" variants, print "cancelled" and return 1 for cancellation. Otherwise, prompt again for correct input.

2.5.3 Code_portion_User-Interface

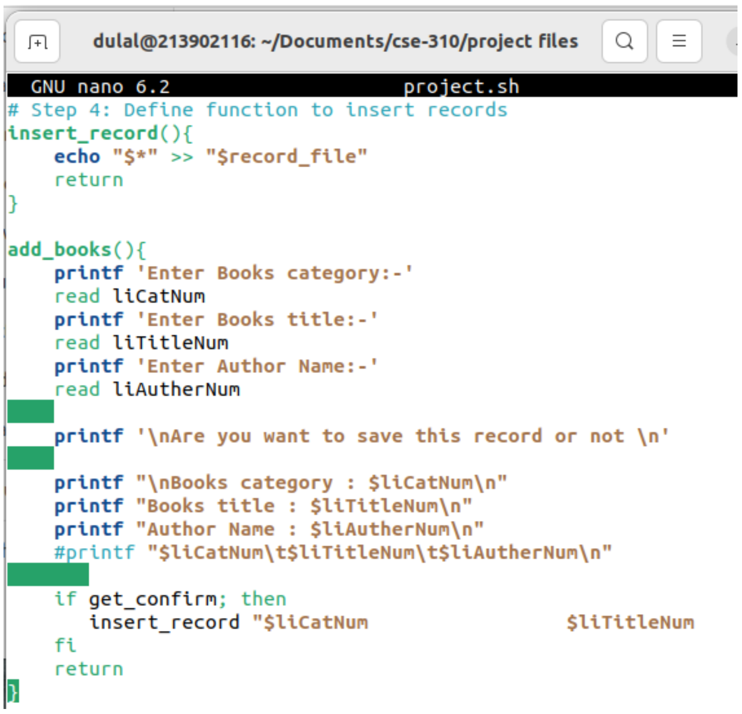


```
dulal@213902116: ~/Documents/cse-310/project files
  GNU nano 6.2                         project.sh
# Step 3: Define main menu function
set_menu_choice(){
    clear
    printf "****************************************************\n\n"
    printf "########## Library Management System ##########\n"
    printf "########## Design By Dulal-213902116 ##########\n\n"
    printf "****************************************************\n\n\n"
    printf '            Options:-\n\n'
    printf '    \ta) Add new Books records\n'
    printf '    \tb) Find Books\n'
    printf '    \tc) Edit Books\n'
    printf '    \td) Remove Books\n'
    printf '    \te) View Books\n'
    printf '    \tf) Quit\n'
    printf 'Please enter the choice then press return : '
    read menu_choice
    return
}
```

Figure 2.3: Code for User-Interface part.

The set-menu-choice() function clears the screen and presents the main menu of the Library Management System. It prompts the user to choose an option from a to f, representing different functionalities. The user's choice is stored in the global variable menu-choice. After reading the choice, the function returns, allowing the program to proceed based on the selected option.

2.5.4 Code_portion_Add-Books

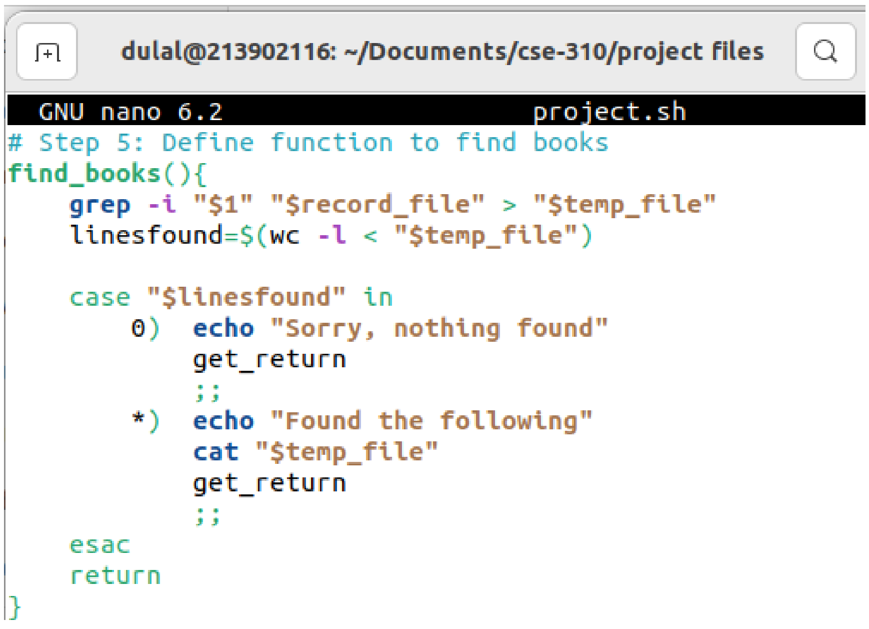


```
dulal@213902116: ~/Documents/cse-310/project files
  GNU nano 6.2                         project.sh
# Step 4: Define function to insert records
insert_record(){
    echo "$*" >> "$record_file"
    return
}

add_books(){
    printf 'Enter Books category:-'
    read liCatNum
    printf 'Enter Books title:-'
    read liTitleNum
    printf 'Enter Author Name:-'
    read liAutherNum

    printf '\nAre you want to save this record or not \n'

    printf "\nBooks category : $liCatNum\n"
    printf "Books title : $liTitleNum\n"
    printf "Author Name : $liAutherNum\n"
    #printf "$liCatNum\t$liTitleNum\t$liAutherNum\n"

    if get_confirm; then
       insert_record "$liCatNum                 $liTitleNum
    fi
    return
}
```

Figure 2.4: Code for Add-Books part.

The insert-record() function appends the provided arguments (book details) to the specified record file. In the add-books() function, it prompts the user to input book details such as category, title, and author. After confirming the input, it calls insert-record() to save the record to the file if the user confirms.

2.5.5 Code_portion_Find-Books

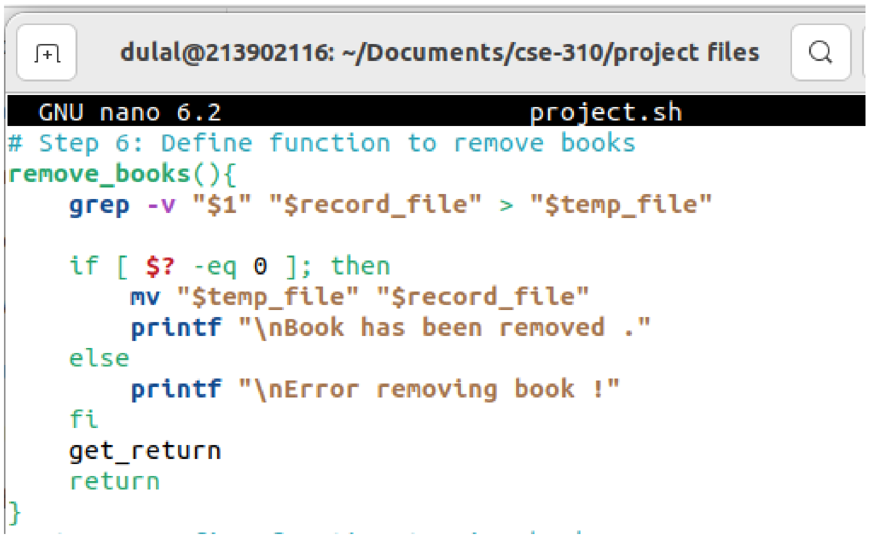


```
  GNU nano 6.2                            project.sh
# Step 5: Define function to find books
find_books(){
    grep -i "$1" "$record_file" > "$temp_file"
    linesfound=$(wc -l < "$temp_file")

    case "$linesfound" in
        0)  echo "Sorry, nothing found"
            get_return
            ;;
        *)  echo "Found the following"
            cat "$temp_file"
            get_return
            ;;
    esac
    return
}
```

Figure 2.5: Code for Find-Books part.

The find-books() function searches for the given term in the record file using grep. It writes the results to a temporary file. Then, it counts the number of lines in the temporary file. If no matches are found, it prints "Sorry, nothing found" and prompts the user to return to the main menu. If matches are found, it prints the results and prompts the user to return.

2.5.6 Code_portion_Remove-Books



```
  GNU nano 6.2                            project.sh
# Step 6: Define function to remove books
remove_books(){
    grep -v "$1" "$record_file" > "$temp_file"

    if [ $? -eq 0 ]; then
        mv "$temp_file" "$record_file"
        printf "\nBook has been removed ."
    else
        printf "\nError removing book !"
    fi
    get_return
    return
}
```

Figure 2.6: Code for Remove-Books part.

The remove-books() function removes the record corresponding to the given book title from the record file using grep -v to exclude the matching line. The result is written to a temporary file. If successful, it replaces the original file with the temporary one. It then prints a message confirming the removal. If unsuccessful, it prints an error message. Finally, it prompts the user to return to the main menu.

2.5.7 Code_portion_View-Books

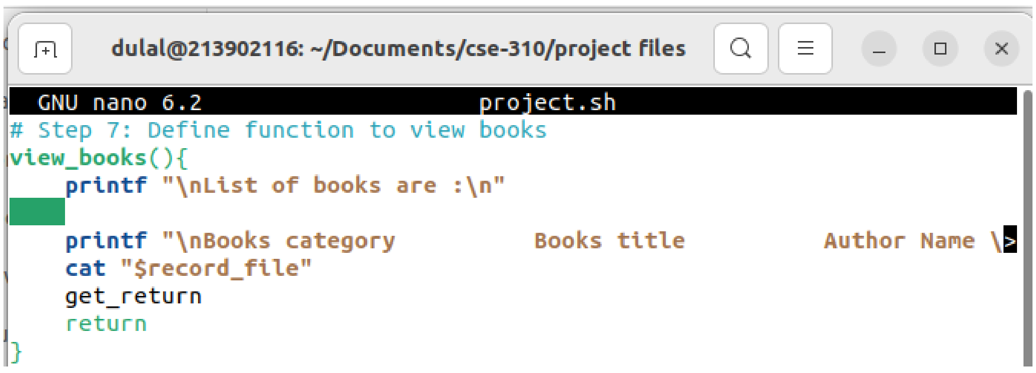

```
dulal@213902116: ~/Documents/cse-310/project files
 GNU nano 6.2                      project.sh
# Step 7: Define function to view books
view_books(){
    printf "\nList of books are :\n"

    printf "\nBooks category            Books title           Author Name \>
    cat "$record_file"
    get_return
    return
}
```

Figure 2.7: Code for View-Books part.

The view-books() function displays the list of books stored in the record file. It prints a header with column names for category, title, and author. Then, it prints the contents of the record file using cat. After displaying the list, it prompts the user to press Enter to return to the main menu.

2.5.8 Code_portion_Edit- Books

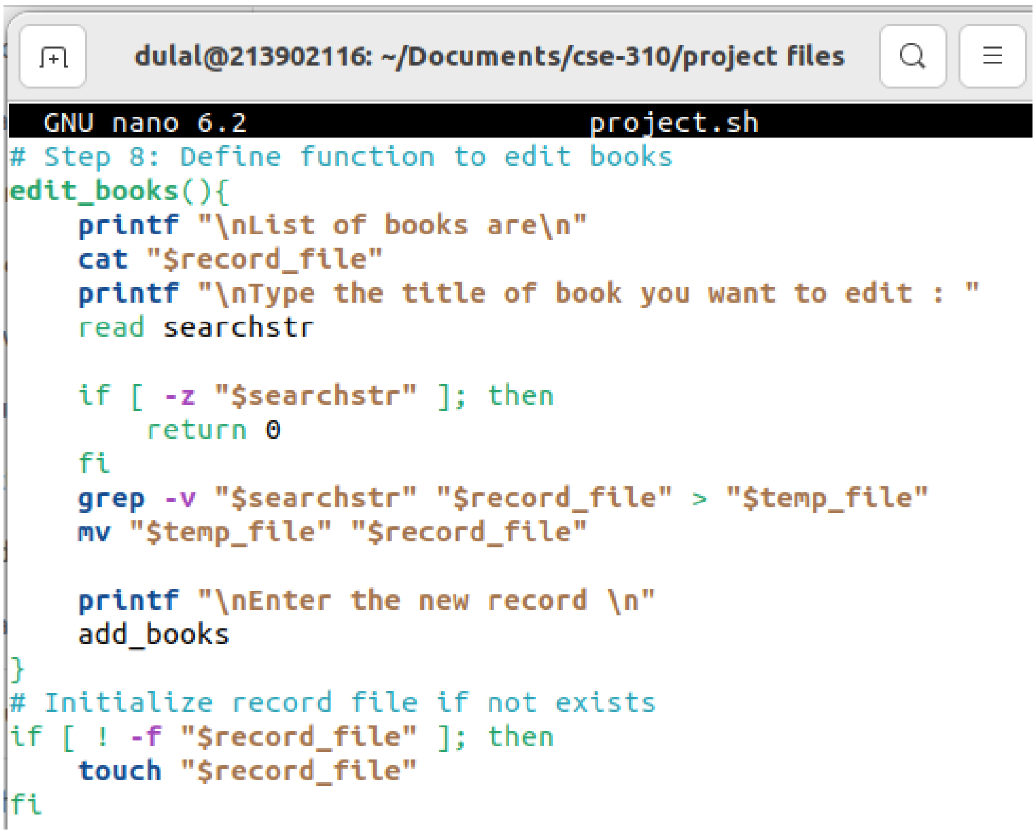
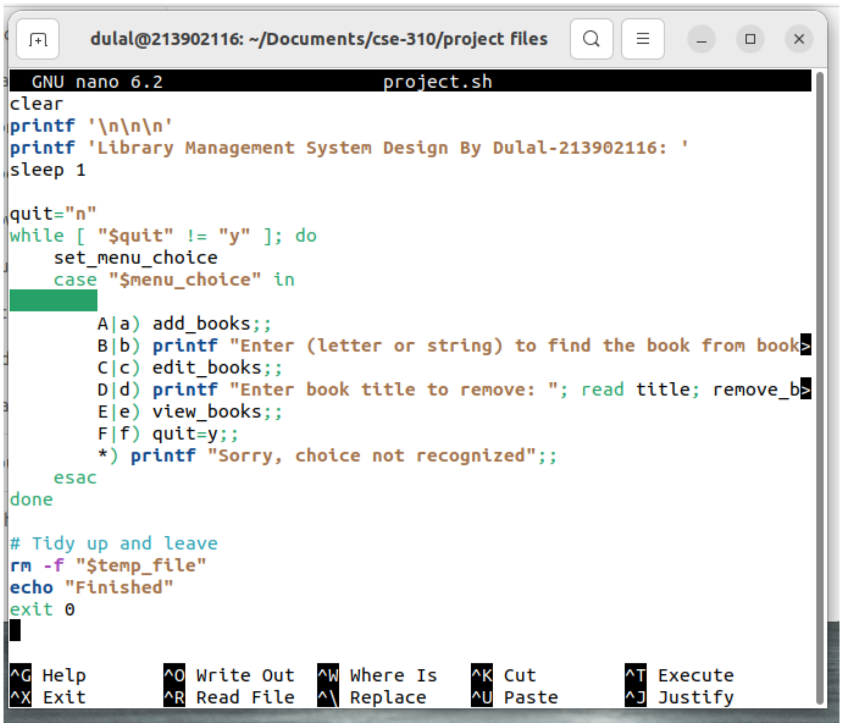

```
dulal@213902116: ~/Documents/cse-310/project files
 GNU nano 6.2                      project.sh
# Step 8: Define function to edit books
edit_books(){
    printf "\nList of books are\n"
    cat "$record_file"
    printf "\nType the title of book you want to edit : "
    read searchstr

    if [ -z "$searchstr" ]; then
        return 0
    fi
    grep -v "$searchstr" "$record_file" > "$temp_file"
    mv "$temp_file" "$record_file"

    printf "\nEnter the new record \n"
    add_books
}
# Initialize record file if not exists
if [ ! -f "$record_file" ]; then
    touch "$record_file"
fi
```

Figure 2.8: Code for Edit-Books part.

The edit-books() function displays the list of books and prompts the user to input the title of the book they want to edit. If the input is empty, the function returns. Otherwise, it removes the record with the specified title from the file, prompts the user to enter a new record, and calls add-books() to insert the new record.

2.5.9 Code_portion_Main Menu



dulal@213902116: ~/Documents/cse-310/project files

```
  GNU nano 6.2                          project.sh
clear
printf '\n\n\n'
printf 'Library Management System Design By Dulal-213902116: '
sleep 1

quit="n"
while [ "$quit" != "y" ]; do
    set_menu_choice
    case "$menu_choice" in

        A|a) add_books;;
        B|b) printf "Enter (letter or string) to find the book from book>
        C|c) edit_books;;
        D|d) printf "Enter book title to remove: "; read title; remove_b>
        E|e) view_books;;
        F|f) quit=y;;
        *) printf "Sorry, choice not recognized";;
    esac
done

# Tidy up and leave
rm -f "$temp_file"
echo "Finished"
exit 0


^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify
```

Figure 2.9: Code for Main-Menu part.

1. **Initialize Record File:**If the file specified by record-file does not exist, create an empty file using touch.
2. **Clear Screen and Display Message:** Clear the terminal screen.

   Print a message: “Library Management System Design By Dulal-213902116”.

   Pause execution for 1 second using sleep.
3. **Main Menu Loop:**

   Initialize quit variable to “n”.

   Enter a loop that continues until quit is set to “y”.
4. **Inside the loop:**Call set-menu-choice (which is not defined in the provided code).

   Based on the user’s choice (menu-choice), execute the corresponding action:

   “A” or “a”: Add books (call add-books function).

   “B” or “b”: Prompt the user to enter a search term and find books (call find-books function).

   “C” or “c”: Edit books (call edit-books function).

   “D” or “d”: Prompt the user to enter a book title and remove books (call remove-books function).

   “E” or “e”: View books (call view-books function).

   “F” or “f”: Set quit to “y” to exit the loop.

   Otherwise: Print “Sorry, choice not recognized”.
5. **Cleanup and Exit:** Remove the temporary file specified by temp-file.

   Print “Finished”.

   Exit the script with status code 0.

Chapter 3

Performance Evaluation

3.1 Simulation Environment/ Simulation Procedure

To simulate the outcomes of our project, me and my teammate have set up different experimental setups based on my PC configurations. In this section, we will discuss the specific requirements and environment installation needed for each simulation.

RAM: 16GB ,

Storage: 512 GB SSD and 1TB HDD,

Processor: Intel Core i5 10th generation.

3.2 Results Analysis/Testing

The results analysis and testing for the library management system project are crucial steps to ensure its functionality, reliability, and user satisfaction.

1. **Functionality Testing:** This involves testing each feature of the system to verify that it performs as intended. It includes adding, finding, editing, and removing book records, as well as viewing the entire catalog.

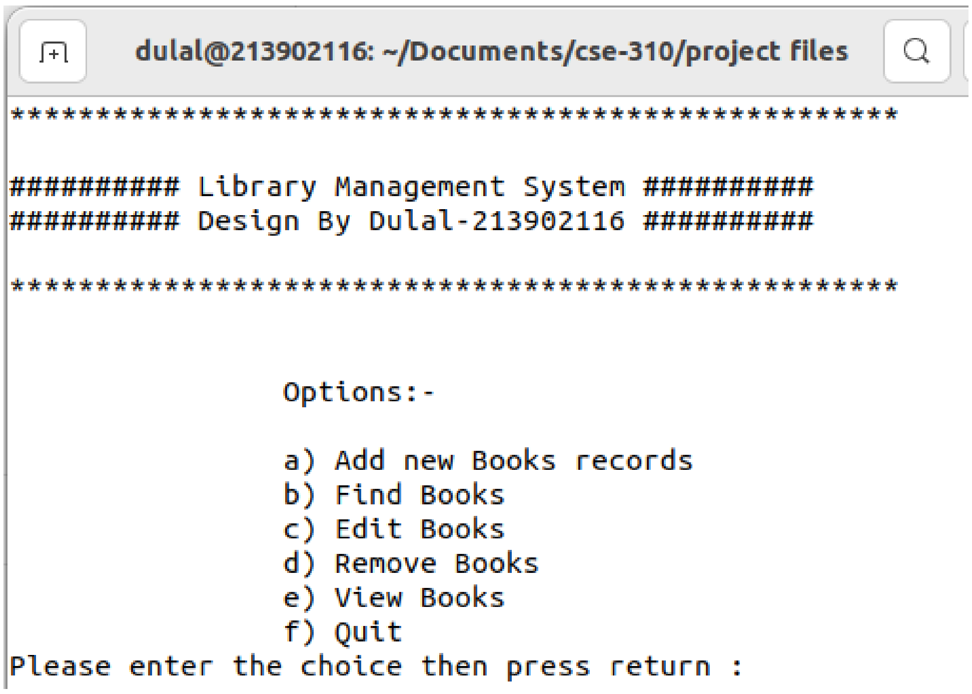
2. **Integration Testing:** Integration testing ensures that different components of the system work together seamlessly. It checks for proper communication between modules and validates data exchange.

3. **User Acceptance Testing (UAT):** UAT involves real users testing the system to validate its usability and effectiveness in real-world scenarios. Feedback from users helps identify areas for improvement.

4. **Performance Testing:** Performance testing assesses the system's responsiveness, scalability, and resource usage under various load conditions. It ensures that the system can handle multiple users and large datasets efficiently.

5. **Security Testing:** Security testing checks for vulnerabilities such as unauthorized access, data breaches, and data integrity issues. It includes testing authentication mechanisms, data encryption, and access controls.

3.2.1 Result_portion_Main-Menu

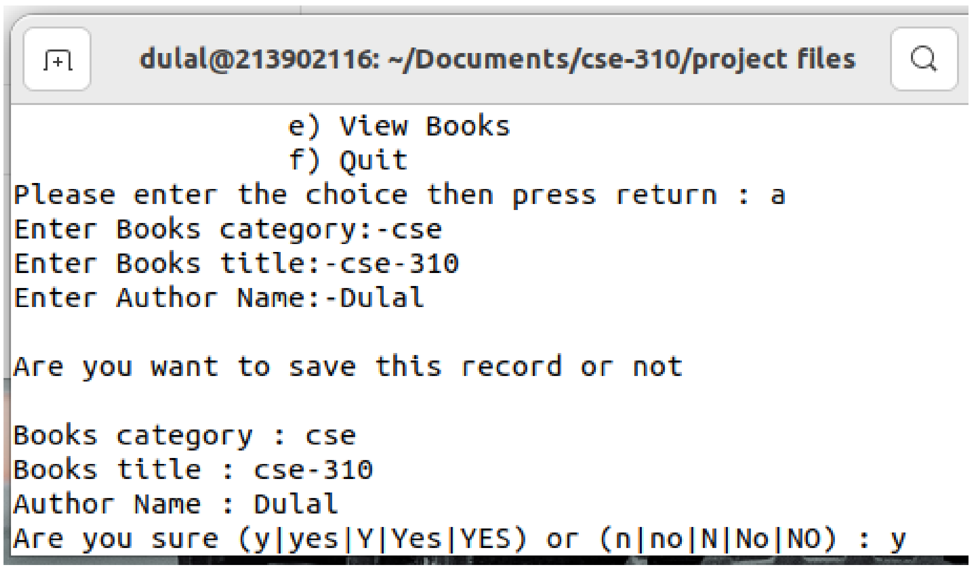


```
dulal@213902116: ~/Documents/cse-310/project files
****************************************************

########## Library Management System ##########
########## Design By Dulal-213902116 ##########

****************************************************


                Options:-

                a) Add new Books records
                b) Find Books
                c) Edit Books
                d) Remove Books
                e) View Books
                f) Quit
Please enter the choice then press return :
```

Figure 3.1: User-Interface Part.

In this Figure 3.1 we see that its my project user Interface . User chose any option just enter A/a ,B/b for each of option.

3.2.2 Result_portion_Add-New-Books

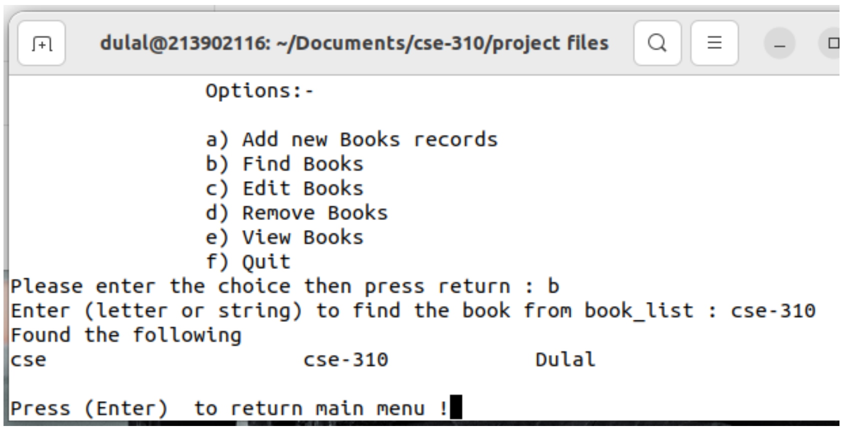


```
dulal@213902116: ~/Documents/cse-310/project files
                e) View Books
                f) Quit
Please enter the choice then press return : a
Enter Books category:-cse
Enter Books title:-cse-310
Enter Author Name:-Dulal

Are you want to save this record or not

Books category : cse
Books title : cse-310
Author Name : Dulal
Are you sure (y|yes|Y|Yes|YES) or (n|no|N|No|NO) : y
```

Figure 3.2: Insert new Books .

In this Figure 3.2 we see that user chose a ,means add book or insert new book , category cse , title cse-310 and author name dulal. Then save this new book enter y other wish n .

3.2.3 Result_portion_Find-Books

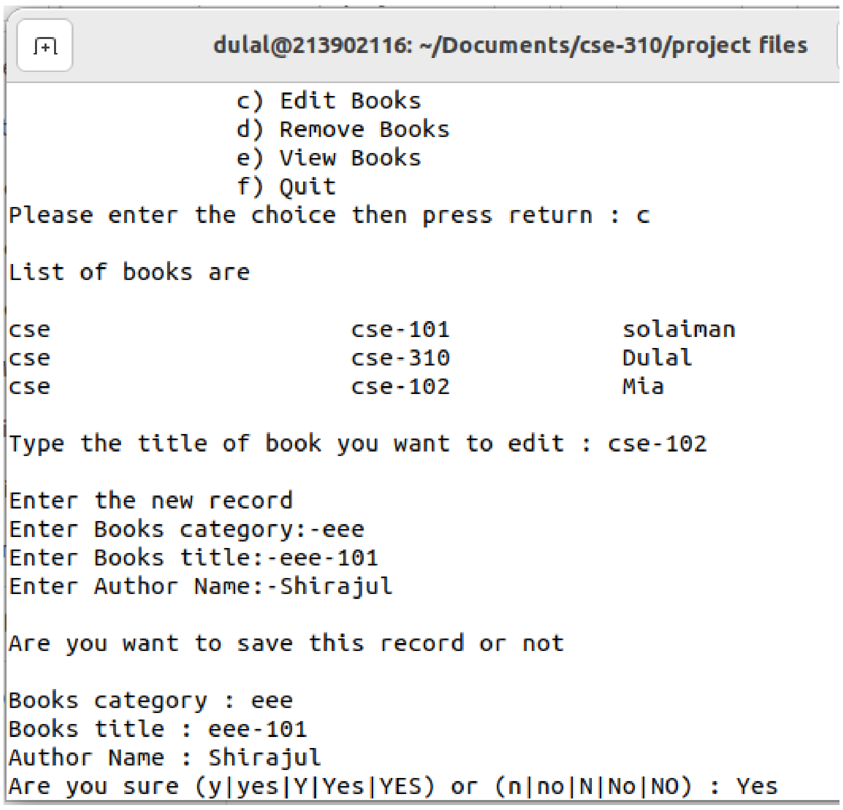


```
dulal@213902116: ~/Documents/cse-310/project files
                Options:-

                a) Add new Books records
                b) Find Books
                c) Edit Books
                d) Remove Books
                e) View Books
                f) Quit
Please enter the choice then press return : b
Enter (letter or string) to find the book from book_list : cse-310
Found the following
cse                     cse-310             Dulal

Press (Enter)  to return main menu !
```

Figure 3.3: Find Books.

In this Figure 3.3 ,see that user given input b means find books , user can search by any letter or string for search the book .user search cse-310 (we newly adder this book in figure 3.2) is found in our record and giver all information, so say output show that successfully.

3.2.4 Result_portion_Edit-Books

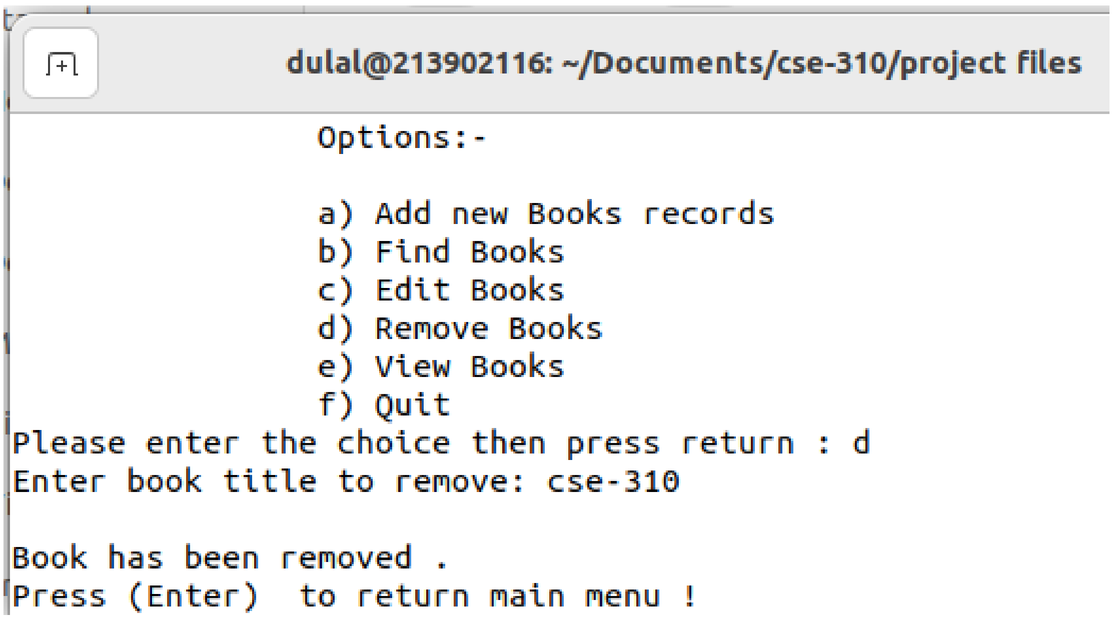


```
dulal@213902116: ~/Documents/cse-310/project files
                c) Edit Books
                d) Remove Books
                e) View Books
                f) Quit
Please enter the choice then press return : c

List of books are

cse                     cse-101             solaiman
cse                     cse-310             Dulal
cse                     cse-102             Mia

Type the title of book you want to edit : cse-102

Enter the new record
Enter Books category:-eee
Enter Books title:-eee-101
Enter Author Name:-Shirajul

Are you want to save this record or not

Books category : eee
Books title : eee-101
Author Name : Shirajul
Are you sure (y|yes|Y|Yes|YES) or (n|no|N|No|NO) : Yes
```

Figure 3.4: Edit-Books.

In this Figure 3.4 ,see that user given input c means edit book ,see book list one book title is cse-102 we want edit this record category eee, title eee-101, author shirajul again save to enter y .

3.2.5 Result_portion_Remove-Books

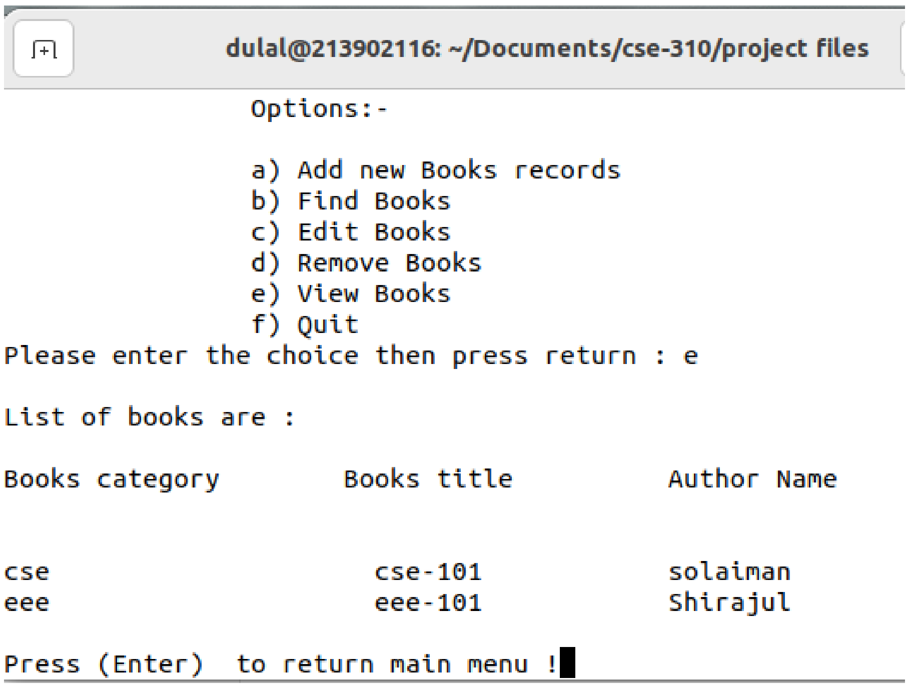


```
dulal@213902116: ~/Documents/cse-310/project files
                         Options:-

                         a) Add new Books records
                         b) Find Books
                         c) Edit Books
                         d) Remove Books
                         e) View Books
                         f) Quit
Please enter the choice then press return : d
Enter book title to remove: cse-310

Book has been removed .
Press (Enter)  to return main menu !
```

Figure 3.5: Remove-Books.

In this Figure 3.5 ,see that user given input d means remove a book from record .user enter title cse-310 (we newly adder this book figure 3.2) press enter then remove from record successfully.

3.2.6 Result_portion_View-Books

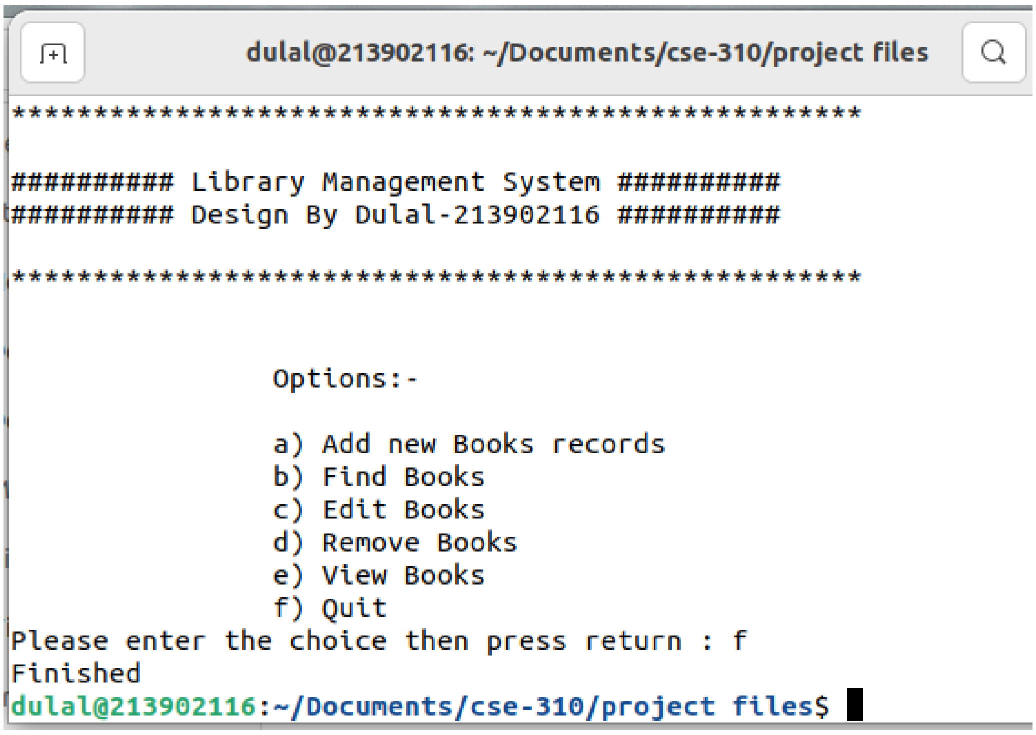


```
dulal@213902116: ~/Documents/cse-310/project files
                    Options:-

                    a) Add new Books records
                    b) Find Books
                    c) Edit Books
                    d) Remove Books
                    e) View Books
                    f) Quit
Please enter the choice then press return : e

List of books are :

Books category        Books title          Author Name


cse                     cse-101            solaiman
eee                     eee-101            Shirajul

Press (Enter)  to return main menu !
```

Figure 3.6: View-Books.

In this Figure 3.6 ,see that user given input e means view books. we see that (we remove the cse-310 figure 3.5) cse-310 book not in the book list. so tell that output successfully show.

3.2.7 Result_portion_Quit



```
dulal@213902116: ~/Documents/cse-310/project files
****************************************************

########## Library Management System ##########
########## Design By Dulal-213902116 ##########

****************************************************


                Options:-

                a) Add new Books records
                b) Find Books
                c) Edit Books
                d) Remove Books
                e) View Books
                f) Quit
Please enter the choice then press return : f
Finished
dulal@213902116:~/Documents/cse-310/project files$
```

Figure 3.7: Quit.

In this Figure 3.7 ,see that user given input f means quit from this project . And its successfully show .

The Library Management System designed by Dulal-213902116 effectively manages book records, allowing users to add, find, edit, and remove books with ease. Its intuitive menu-driven interface ensures user-friendly navigation. Through meticulous implementation, the system enables seamless interaction, enhancing efficiency in library operations. It incorporates essential features like record insertion, search functionality, and data modification, ensuring comprehensive management of library resources. This project exemplifies meticulous attention to detail and robust functionality, resulting in a successful solution that streamlines library tasks, ultimately benefiting librarians and patrons alike.

3.3 Results Overall Discussion

The library management system designed by Dulal-213902116 offers a straightforward approach to managing book records. Users can easily add, find, edit, and remove book entries through a simple command-line interface. The system ensures data integrity by prompting users to confirm their actions before making any changes to the database. Additionally, it provides a clear overview of the existing book records, enhancing the overall user experience. Despite its simplicity, the system effectively fulfills the basic requirements of a library management system in a Unix-based environment, contributing to efficient book record management.

Chapter 4

Conclusion

4.1 Discussion

The library management system designed by Dulal-213902116 offers a simplistic yet effective solution for managing book records. By providing a user-friendly command-line interface, it allows users to seamlessly add, find, edit, and remove book entries. The system ensures data integrity by prompting users to confirm their actions before making any changes to the database, enhancing reliability. Despite its simplicity, the system effectively fulfills the basic requirements of a library management system in a Unix-based environment. Furthermore, its clear presentation of book records enhances user experience and contributes to efficient book record management. Overall, the system provides a practical and efficient solution for managing book records in a library setting.

4.2 Limitations

While the library management system presented here offers basic functionality for managing book records, it has certain limitations that could be addressed for improvement:

1. **Limited Scalability:** The system is designed for a small-scale library and may not efficiently handle a large volume of book records.
2. **Single-User Interface:** It lacks support for concurrent access by multiple users, restricting its usability in environments where simultaneous access is required.
3. **Limited Data Validation:** The system does not incorporate extensive data validation mechanisms, leaving room for erroneous or inconsistent data entry.
4. **Minimal Security Measures:** It lacks robust security features such as user authentication and access control, potentially compromising the confidentiality and integrity of book records.
5. **Command-Line Interface Only:** The system relies solely on a command-line interface, which may not be intuitive for all users, especially those unfamiliar with command-line interactions. Integrating a graphical user interface (GUI) could enhance user experience and accessibility.

4.3 Scope of Future Work

The library management system outlined here lays a foundation for future enhancements and expansions to cater to evolving needs. Here's the scope of future work for this project:

1. **Enhanced User Interface:** Implementing a graphical user interface (GUI) would improve usability and accessibility, allowing users to interact with the system more intuitively.
2. **Multi-User Support:** Incorporating features for concurrent access by multiple users would make the system more scalable and suitable for larger libraries or institutions.
3. **Advanced Search and Sorting:** Introducing advanced search functionalities and sorting options would enhance the efficiency of finding and managing book records.
4. **Data Validation and Error Handling:** Implementing robust data validation mechanisms and error handling procedures would ensure the integrity and consistency of book records.
5. **Integration with External Systems:** Integrating with external systems such as library databases or online catalogs would extend the system's functionality and usefulness.

By addressing these areas, the library management system can evolve into a comprehensive solution that meets the diverse needs of libraries and users alike.

4.4 References

1. https://www.javatpoint.com/shell-scripting-tutorial
2. https://www.learnshell.org/
3. https://www.udemy.com/topic/shell-scripting/
4. https://www.tutorialspoint.com/unix/shell-scripting.htm
5. https://www.shellscript.sh/
6. https://www.freecodecamp.org/news/bash-scripting-tutorial-linux-shell-script-and-command-line-for-beginners/