B.M.S COLLEGE OF ENGINEERING BENGALURU
Autonomous Institute, Affiliated to VTU

UNIX SHELL PROGRAMMING
Report on

Library Management

Submitted in partial fulfillment of the requirements for AAT

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

Signature of the Candidates

Chirag C S (1BM22CS079)

Deekshith B (1BM22CS082)

Neha C (1BM22CS074)

Ganashree C M (1BM22CS097)

Department of Computer Science and Engineering
B.M.S College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019

2023-2024

B.M.S COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, **Chirag C S (1BM22CS079) Deekshith B (1BM22CS082) Neha C (1BM22CS074) Ganashree C M (1BM22CS097)** students of 3rd Semester, B.E, Department of Computer Science and Engineering, BMS College of Engineering, Bangalore, hereby declare that, this AAT Project entitled "Library Management" has been carried out in Department of CSE, BMS College of Engineering, Bangalore during the academic semester December - March 2024. We also declare that to the best of our knowledge and belief, the Project report is not from part of any other report by any other students.

Signature of the Candidates

Chirag C S (1BM22CS079)

Deekshith B (1BM22CS082)

Neha C (1BM22CS074)

Ganashree C M (1BM22CS097)

BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the AAT Project titled "**Library Management"** has been carried out by **Chirag C S (1BM22CS079) Deekshith B (1BM22CS082) Neha C (1BM22CS074) Ganashree C M (1BM22CS097**) during the academic year 2023-2024.

Signature of the Faculty in Charge Signature of Head of Department

Signature of Examiner with date

1. Internal Examiner _____
2. External Examiner _____

Abstract

In this project, we present a comprehensive UNIX shell script developed to streamline the **management of libraries**, offering a simplified yet efficient solution for organizing and accessing library resources. By amalgamating a variety of UNIX commands and scripting techniques, the system provides users with an intuitive graphical interface, facilitating seamless interaction with library functionalities. Users are empowered to register as either authors or regular users, with tailored access rights corresponding to their roles. Authors are equipped with the capability to enrich the library's collection by adding new books, while regular users benefit from functionalities such as browsing the catalogue, conducting author-based searches, and executing borrowing and returning operations effortlessly.

Stringent authentication measures are implemented to ensure secure access to the system and safeguard sensitive information. Furthermore, the script incorporates advanced features including dynamic book count calculations based on authors and comprehensive tabulation of total library holdings, thus furnishing librarians with invaluable insights into resource distribution and usage patterns. This project serves as a testament to the versatility and effectiveness of shell scripting in orchestrating complex workflows, offering a robust solution for efficient library resource management within UNIX environments.

Table of Contents

1. Introduction

1.1 Challenges

Developing a robust library management system in Bash presented several challenges, including:

- Limited functionality: Bash scripting is not as powerful as higher-level programming languages, which posed limitations on the complexity of the system.

- User interface: Creating a user-friendly interface within the constraints of a command-line environment required careful design and implementation.

-Security: Managing user authentication securely without access to advanced encryption techniques were a significant concern.

1.2 Motivation For The Work

- The motivation behind this project stems from the need for a lightweight and easily deployable library management system.

- Bash scripting offers a convenient way to automate tasks and create simple applications without the need for extensive dependencies or setup.

- Leveraging the inherent capabilities of Bash scripting allows for seamless integration with existing Unix utilities and commands, making it an ideal choice for system administrators and developers familiar with Unix environments.

- Furthermore, the lightweight nature of Bash scripts facilitates rapid prototyping and iteration, enabling quick adjustments to meet evolving requirements.

- Ultimately, this project serves as a testament to the adaptability and effectiveness of Bash scripting in addressing real-world challenges in software development and automation.

2. Methodology and Framework

2.1 System Requirements

The system requires:

- Bash shell environment.

- Standard Unix utilities and commands.

- `dialog` utility for creating a text-based user interface.

- Access to basic file manipulation commands such as `grep`, `cut`, and `sort`.

- Availability of resources for storing user login credentials and book information, typically achieved through text files.

2.1 Algorithms , Techniques

- User authentication: Simple username/password authentication is implemented using text files to store login credentials securely.

- File handling: The system utilizes basic file manipulation techniques to store and retrieve book and user data.

- Data validation: Input validation techniques are employed to ensure the integrity and correctness of user-provided information, minimizing errors and enhancing system reliability.

- Error handling: Robust error handling mechanisms are implemented to gracefully handle unexpected situations and provide informative feedback to users, ensuring a smooth user experience.

3.Implementation

3.1 List of Main UNIX Commands

1. **zenity**: Used for creating graphical user interfaces in shell scripts.
2. **grep**: Used to search for specific patterns within files.
3. **cut**: Used to extract specific columns from a file.
4. **sort**: Used to sort the content of a file.
5. **uniq**: Used to filter out duplicate lines in a file.
6. **wc**: Used to count the number of lines in a file.
7. **date**: Used to get the current date.
8. **sed**: A stream editor used to perform basic text transformations on an input stream.
9. **exit**: A shell built-in command used to exit the current shell with a specified exit status.
10. **case**: Used for conditional branching based on pattern matching.
11. **while**: Used to repeatedly execute a set of commands as long as a specified condition is true.
12. **if**: Used for conditional execution of commands based on the evaluation of a condition.

3.2 Uses and its Syntax

1. **zenity:**
    - Uses: Used for creating graphical user interfaces in shell scripts.
    - Syntax: zenity [options]
2. **grep:**
    - Uses: Used to search for specific patterns within files.
    - Syntax: grep [options] [pattern] [file]
3. **cut:**
    - Uses: Used to extract specific columns from a file.
    - Syntax: cut [options] [file]
4. **sort:**
    - Uses: Used to sort the content of a file.
    - Syntax: sort [options] [file]
5. **uniq:**
    - Uses: Used to filter out duplicate lines in a file.
    - Syntax: uniq [options] [file]
6. **wc:**
    - Uses: Used to count the number of lines in a file.
    - Syntax: wc [options] [file]
7. **date:**
    - Uses: Used to get the current date.
    - Syntax: date [options] [+format]
8. **sed:**
    - Uses: A stream editor used to perform basic text transformations on an input stream.
    - Syntax: sed [options] [script] [file]
9. **exit:**
    - Uses: A shell built-in command used to exit the current shell with a specified exit status.
    - Syntax: exit [status]

10. case:

- Uses: Used for conditional branching based on pattern matching.
- Syntax: case [value] in pattern1) commands;; pattern2) commands;; esac

11. while:

- Uses: Used to repeatedly execute a set of commands as long as a specified condition is true.
- Syntax: while [condition]; do commands; done

12. if:

- Uses: Used for conditional execution of commands based on the evaluation of a condition.
- Syntax: if [condition]; then commands; fi

3.3 Source code:

```
#!/bin/bash

LIBRARY_FILE="library.csv"
AUTHOR_LOGIN_FILE="author_login.txt"
USER_LOGIN_FILE="user_login.txt"
BORROWED_BOOKS_FILE="borrowed_books.txt"

# Function to register a new author
register_author() {
    new_author_username=$(zenity --entry --title "Register New Author" --text "Enter new author username:" --width=400 --height=500)
    new_author_password=$(zenity --password --title "Register New Author" --text "Enter new author password:" --width=400 --height=500)

    echo "$new_author_username:$new_author_password" >> "$AUTHOR_LOGIN_FILE"
    zenity --info --text "New author registered successfully! 🎉" --width=400 --height=500
}

# Function to register a new user
register_user() {
    new_user_username=$(zenity --entry --title "Register New User" --text "Enter new user username:" --width=400 --height=500)
    new_user_password=$(zenity --password --title "Register New User" --text "Enter new user password:" --width=400 --height=500)

    echo "$new_user_username:$new_user_password" >> "$USER_LOGIN_FILE"
    zenity --info --text "New user registered successfully! 🎉" --width=400 --height=500
}
```

```
# Function to authenticate author using zenity
authenticate_author() {
   author_username=$(zenity --entry --title "Author Login" --text "Enter author username:" --width=400 --height=500)
   author_password=$(zenity --password --title "Author Login" --text "Enter author password:" --width=400 --height=500)

   if grep -q "^$author_username:$author_password$" "$AUTHOR_LOGIN_FILE"; then
      zenity --info --text "Author authentication successful! 🎉" --width=400 --height=500
   else
      zenity --error --text "Author authentication failed. Exiting... ✖" --width=400 --height=500
      exit 1
   fi
}

# Function to authenticate user using zenity
authenticate_user()                                                                    {
1,1           Top# Function to authenticate user using zenity
authenticate_user() {
   user_username=$(zenity --entry --title "User Login" --text "Enter user username:" --width=400 --height=500)
   user_password=$(zenity --password --title "User Login" --text "Enter user password:" --width=400 --height=500)

   if grep -q "^$user_username:$user_password$" "$USER_LOGIN_FILE"; then
      zenity --info --text "User authentication successful! 🎉" --width=400 --height=500
   else
      zenity --error --text "User authentication failed. Exiting... ✖" --width=400 --height=500
      exit 1
```

```
    fi
}

# Function to determine login type
select_login_type() {
    login_type=$(zenity --list --title "Login Type" --column "Select login type:" "Author" "User" "Register New User/Author" --width=400 --height=500)

    case $login_type in
        "Author") authenticate_author ;;
        "User") authenticate_user ;;
        "Register New User/Author") register_new_user_author ;;
        *) zenity --error --text "Invalid selection. Exiting... ✗" --width=400 --height=500; exit 1 ;;
    esac
}

# Function to register a new user or author
register_new_user_author() {
    registration_type=$(zenity --list --title "Registration Type" --column "Select registration type:" "Register New Author" "Register New User" --width=400 --height=500)

    case $registration_type in
        "Register New Author") register_author ;;
        "Register New User") register_user ;;
        *) zenity --error --text "Invalid selection. Exiting... ✗" --width=400 --height=500; exit 1 ;;
    esac
}

# Function to display the main menu
```

```
display_menu() {
    choice=$(zenity --list --title "Library Management System" --text "Choose an option:" \
    # Function to display the main menu
display_menu() {
    choice=$(zenity --list --title "Library Management System" --text "Choose an option:" \
        --column="Option" --column="Description" \
        "Add Book" "Add a new book to the library" \
        "View All Books" "View all books in the library" \
        "Search Books by Author" "Search books by author" \
        "Display All Authors" "Display all unique authors" \
        "Borrow Book" "Borrow a book from the library" \
        "Return Book" "Return a borrowed book" \
        "Count Books by Author" "Count the number of books by a specific author" \
        "Display Total Books" "Display the total number of books in the library" \
        "Edit Book" "Edit book information" \
        "Delete Book" "Delete a book from the library" \
        "Display Borrowed Books" "View books borrowed by the user" \
        "Exit" "Exit the program" --width=600 --height=500)

    case $choice in
        "Add Book") add_book ;;
        "View All Books") view_books ;;
        "Search Books by Author") search_books_by_author ;;
        "Display All Authors") display_authors ;;
        "Borrow Book") borrow_book ;;
        "Return Book") return_book ;;
        "Count Books by Author") count_books_by_author ;;
        "Display Total Books") display_total_books ;;
        "Edit Book") edit_book ;;
```

```
        "Delete Book") delete_book ;;
        "Display Borrowed Books") display_borrowed_books ;;
        "Exit") zenity --info --text "Exiting... 🚪" --width=400 --height=500; exit 0 ;;
        *) zenity --error --text "Invalid choice. Please try again. ❌" --width=400 --height=500
;;
    esac
}
# Function to display borrowed books
display_borrowed_books() {
    if [ -s "$BORROWED_BOOKS_FILE" ]; then
        borrowed_books_info=$(cat "$BORROWED_BOOKS_FILE")
        zenity --text-info --title "Borrowed Books" --filename "$BORROWED_BOOKS_FILE"
--width=600 --height=500
    else
        zenity --info --text "No books have been borrowed yet. 📚" --width=400 --height=500}

# Function to delete a book
delete_book() {
    book_title=$(zenity --entry --title "Delete Book" --text "Enter the title of the book to delete:"
--width=400 --height=500)

    # Check if the book exists in the library
    if grep -q "^$book_title" "$LIBRARY_FILE"; then
        # Remove the book from the library
        sed -i "/^$book_title,/d" "$LIBRARY_FILE"
        zenity --info --text "Book deleted successfully! 📚" --width=400 --height=500
    else
        zenity --error --text "Book not found in the library. Please check the title and try again.
❌" --width=400 --height=500
    fi
```

```
}

# Function to edit book information

edit_book() {

    book_title=$(zenity --entry --title "Edit Book" --text "Enter the title of the book to edit:" --width=400 --height=500)


    # Check if the book exists in the library

    if grep -q "^$book_title" "$LIBRARY_FILE"; then

        # Prompt for new information

        new_title=$(zenity --entry --title "Edit Book" --text "Enter new title:" --width=400 --height=500)

        new_author=$(zenity --entry --title "Edit Book" --text "Enter new author:" --width=400 --height=500)

        new_quantity=$(zenity --entry --title "Edit Book" --text "Enter new quantity:" --width=400 --height=500)


        # Update the book information

        sed -i "/^$book_title,/ s/^.*$/$new_title,$new_author,$new_quantity/" "$LIBRARY_FILE"

        zenity --info --text "Book information updated successfully! 📚" --width=400 --height=500

    else

        zenity --error --text "Book not found in the library. Please check the title and try again. ❌" --width=400 --height=500

    fi

}



# Function to add a book (only accessible to authors)

add_book() {

    # Check if the user is an author

    if ! grep -q "^$author_username" "$AUTHOR_LOGIN_FILE"; then
```

```
        zenity --error --text "Only authors can add books. Please log in as an author. ❌" --width=400 --height=500
    title=$(zenity --entry --title "Add Book" --text "Enter book title:" --width=400 --height=500)
    quantity=$(zenity --entry --title "Add Book" --text "Enter the quantity of books to add:" --width=400 --height=500)

    # Validate if quantity is a positive integer
    if ! [[ "$quantity" =~ ^[1-9][0-9]*$ ]]; then
        zenity --error --text "Invalid quantity. Please enter a positive integer for the number of books. ❌" --width=400 --height=500
        return
    fi

    # Add the book entries to the library file
    for ((i=1; i<=$quantity; i++)); do
        echo "$title,$author_username" >> "$LIBRARY_FILE"
    done

    zenity --info --text "Books added successfully! 📚\nTotal Books Added: $quantity" --width=400 --height=500
}

# Function to view all books
view_books() {
    zenity --text-info --title "View All Books" --filename "$LIBRARY_FILE" --width=600 --height=500
}

# Function to search for books by author
search_books_by_author() {
```

```
    author=$(zenity --entry --title "Search Books by Author" --text "Enter author:" --width=400 --height=500)
    result=$(grep -i "$author" "$LIBRARY_FILE" | cut -d ',' -f 1)
    if [ -n "$result" ]; then
        zenity --info --title "Books by $author" --text "$(echo "$result" | tr '\n' ' ')" --width=600 --height=500
    else
        zenity --info --text "No books found for author $author. ✗" --width=400 --height=500
    fi
}

# Function to display unique authors
display_authors() {
    authors=$(cut -d ',' -f 2 "$LIBRARY_FILE" | sort | uniq)
    zenity --info --title "Unique Authors" --text "$(echo "$authors" | tr '\n' ' ')" --width=600 --height=500
}# Function to count the number of books by a specific author
count_books_by_author() {
    author_to_count=$(zenity --entry --title "Count Books by Author" --text "Enter author's name:" --width=400 --height=500)
    count=$(grep -ic "$author_to_count" "$LIBRARY_FILE")

    if [ "$count" -gt 0 ]; then
        zenity --info --title "Books by $author_to_count" --text "Number of books: $count" --width=400 --height=500
    else
        zenity --info --text "No books found for author $author_to_count. ✗" --width=400 --height=500
    fi
}
```

```
# Function to display the total number of books in the library with tables
display_total_books() {
    total_books=$(wc -l < "$LIBRARY_FILE")
    zenity --info --title "Total Number of Books in the Library" --text "Total Books: $total_books" --width=400 --height=500
}

# Function to borrow a book
borrow_book() {
    # Check if the user is not an author
    if grep -q "^$user_username" "$AUTHOR_LOGIN_FILE"; then
        zenity --error --text "Authors cannot borrow books. Please log in as a user. ✖" --width=400 --height=500
        return
    fi

    book_title=$(zenity --entry --title "Borrow Book" --text "Enter the title of the book you want to borrow:" --width=400 --height=500)
    # Check if the book exists in the library
    if grep -q "^$book_title" "$LIBRARY_FILE"; then
        # Get current date
        current_date=$(date +"%Y-%m-%d")
        # Get the author's name
        author_name=$(grep "^$book_title," "$LIBRARY_FILE" | cut -d ',' -f 2)
        # Store borrowed book information with user and author
        echo "$book_title,$user_username,$author_name,$current_date" >> "$BORROWED_BOOKS_FILE"
        zenity --info --text "Book borrowed successfully! 🕮\nDate of borrowing: $current_date" --width=400 --height=500
        # Remove the borrowed book from the library
        sed -i "/^$book_title/d" "$LIBRARY_FILE"
```

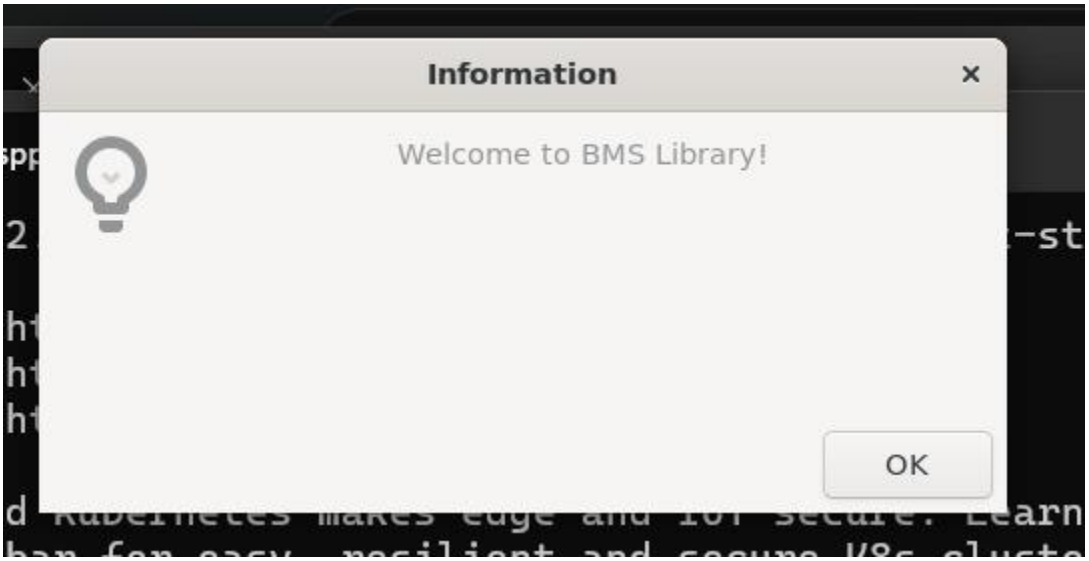
```
    else
        zenity --error --text "Book not found in the library. Please check the title and try again. ❌" --width=400 --height=500
# Function to return a borrowed book
# Function to return a borrowed book with possible fine calculation
return_book() {
    book_to_return=$(zenity --entry --title "Return Book" --text "Enter the title of the book to return:" --width=400 --height=500)

    # Check if the book is borrowed
    if grep -q "^$book_to_return" "$BORROWED_BOOKS_FILE"; then
        borrowed_info=$(grep "^$book_to_return" "$BORROWED_BOOKS_FILE")
        return_date=$(echo "$borrowed_info" | cut -d ',' -f 4)

        current_date=$(date +"%Y-%m-%d")
        days_diff=$(( ($(date -d "$current_date" +%s) - $(date -d "$return_date" +%s)) / (60*60*24) ))

        if [ "$days_diff" -gt 15 ]; then
            fine_amount=$(bc <<< "scale=2; $FINE_RATE * ($days_diff - 15)")
            zenity --info --text "Book returned successfully! 📚\nFine imposed: $fine_amount" --width=400 --height=500
        else
            zenity --info --text "Book returned successfully! 📚" --width=400 --height=500
        fi

        # Remove the book from borrowed list
        sed -i "/^$book_to_return/d" "$BORROWED_BOOKS_FILE"

        # Add the book back to the library with the author's name
        author_name=$(grep -i "^$book_to_return" "$LIBRARY_FILE" | cut -d ',' -f 2)
```

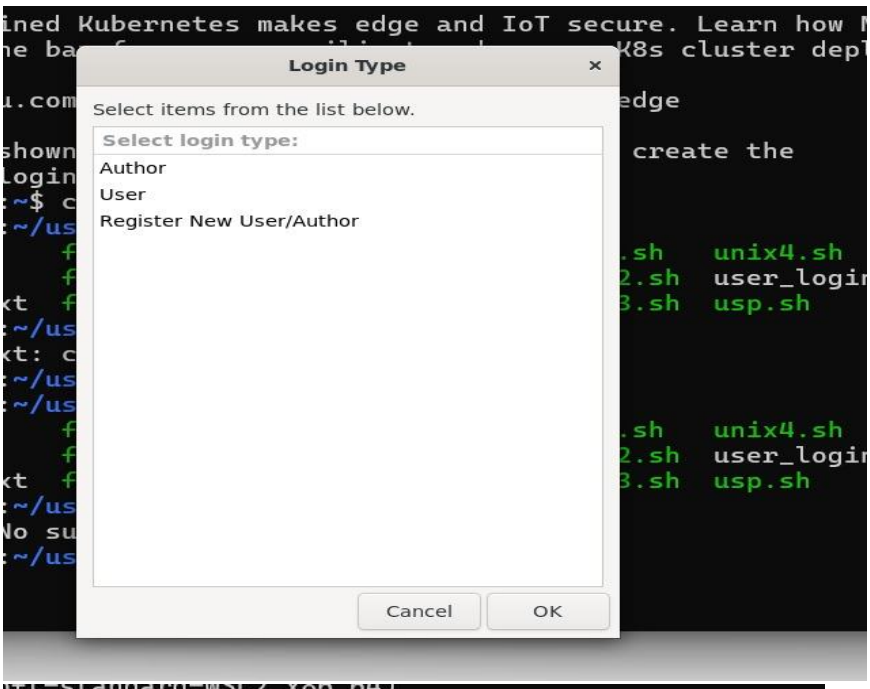
```
        echo "$book_to_return,$author_name" >> "$LIBRARY_FILE"
    else
        zenity --error --text "Book not found in borrowed books. Please check the title and try again. ✗" --width=400 --height=500
    fi
}


# Main program
select_login_type

while true; do
    display_menu
done
```

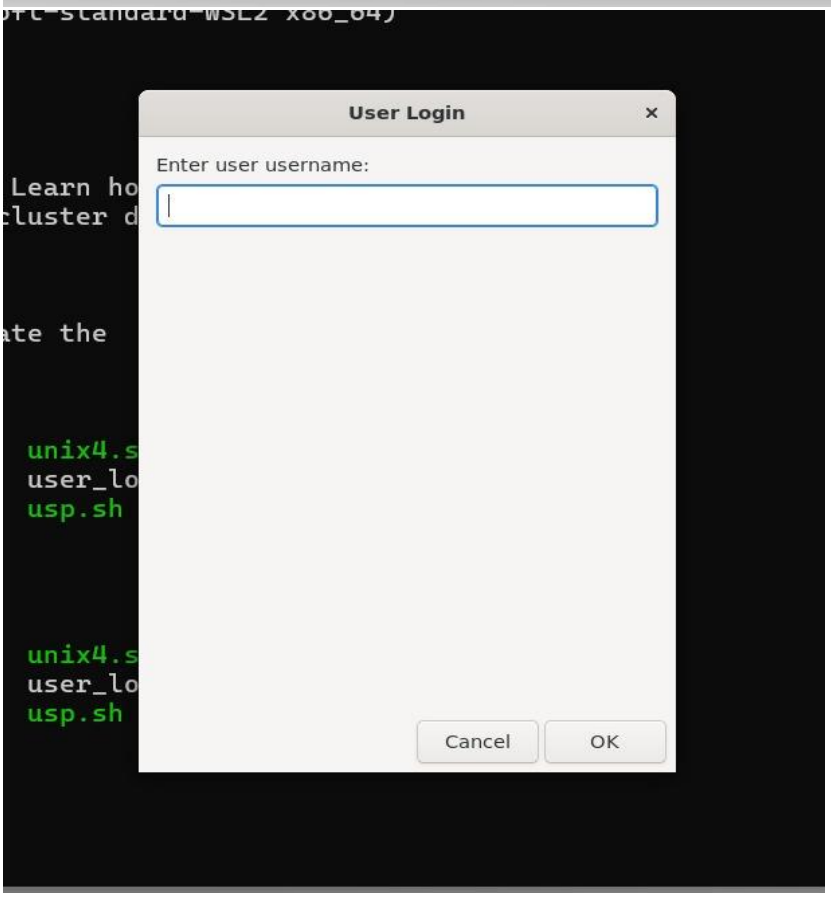
4.Result and Analysis

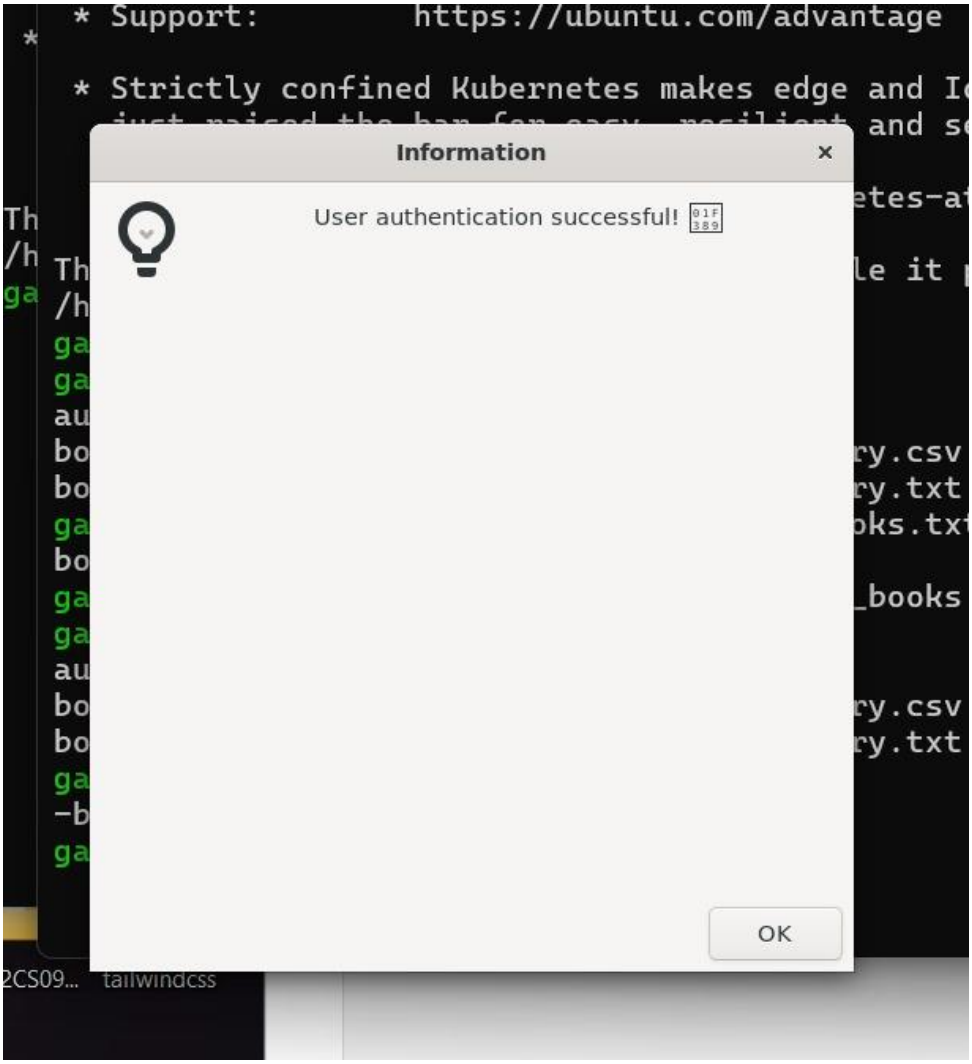
The provided shell script implements a basic library management system utilizing various UNIX commands and functionalities. Below is an analysis of the key features and functionalities provided by the script:

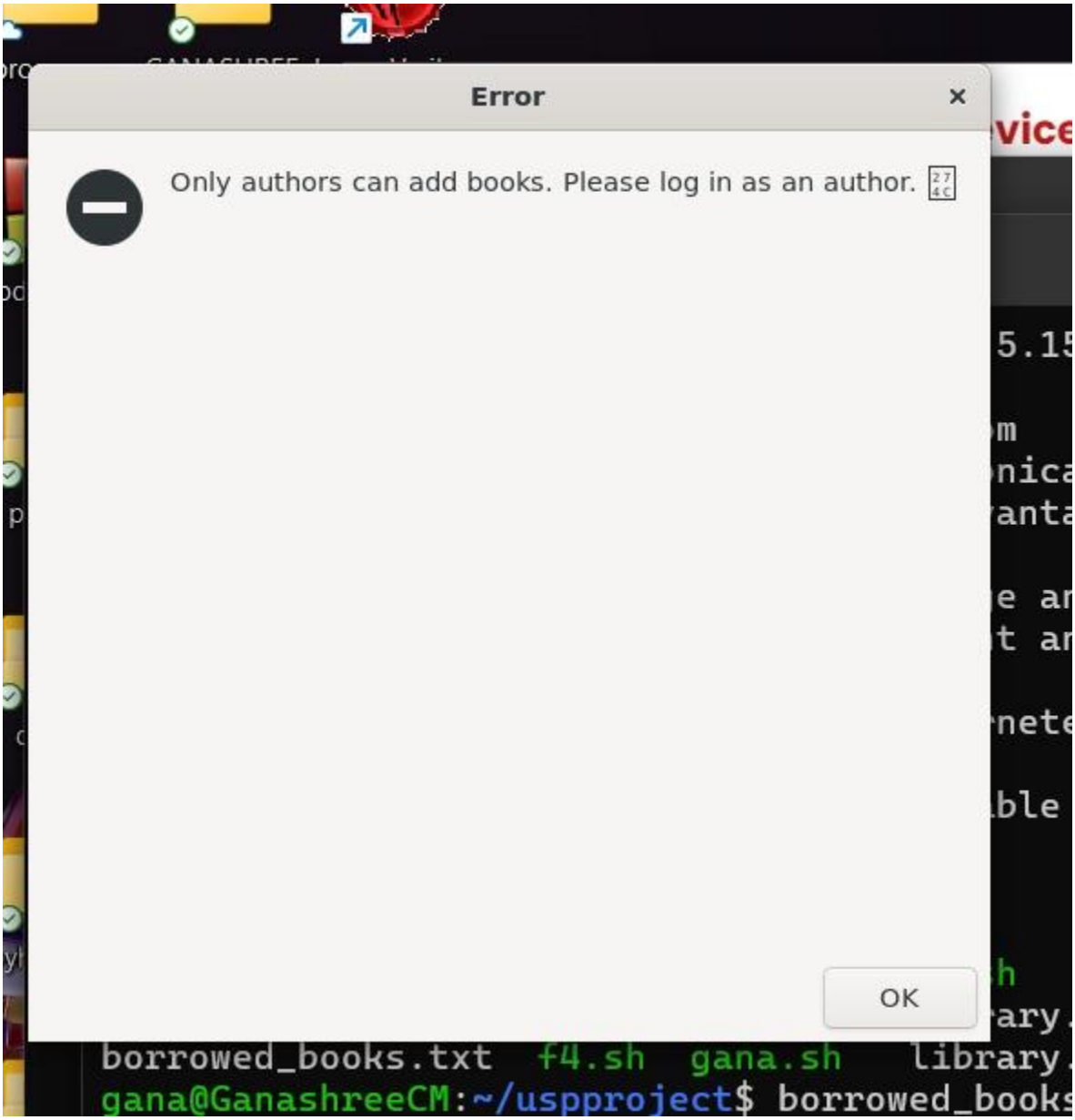
1. **User Registration and Authentication:**
    - The script allows users to register as authors or regular users by providing a username and password.
    - Authentication mechanisms are implemented for both authors and users to verify login credentials against stored records.
2. **Adding Books to the Library:**
    - Authors can add new books to the library by providing the title and author name through a graphical user interface.
    - The script ensures that only authors are allowed to add books to maintain data integrity.
3. **Viewing and Searching Books:**
    - Users can view all books available in the library or search for books by author.
    - Book search functionality provides users with a convenient way to locate specific books of interest.
4. **Displaying Unique Authors:**
    - The script offers functionality to display a list of unique authors present in the library.
5. **Borrowing and Returning Books:**
    - Users can borrow books from the library, and the system records the borrowing date.
    - Returned books are marked as available again in the library.
6. **Counting Books by Author:**
    - Users can retrieve the count of books authored by a specific author.
7. **Displaying Total Books in the Library:**
    - The script provides an option to display the total number of books available in the library.

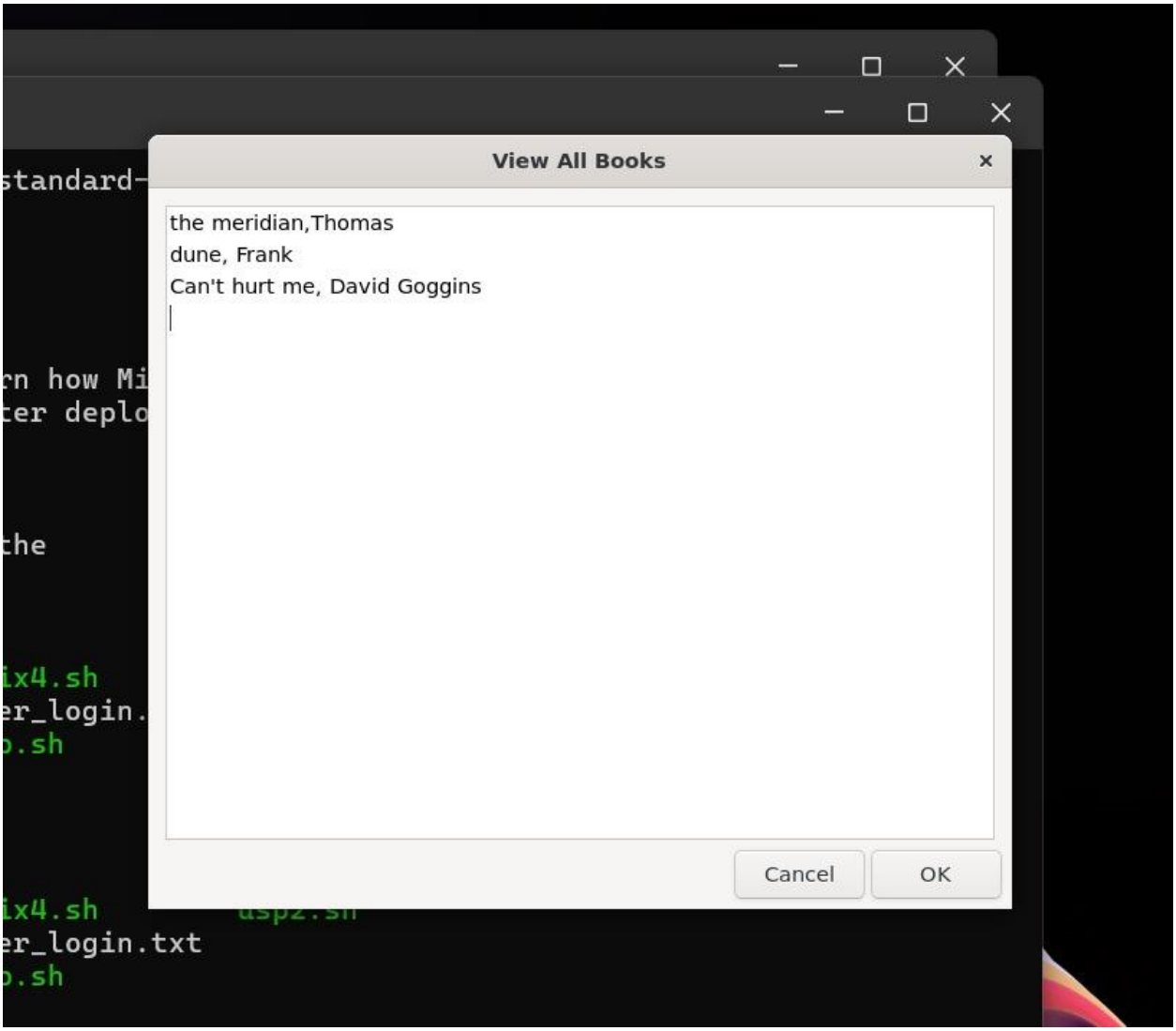

Information
Welcome to BMS Library!
OK

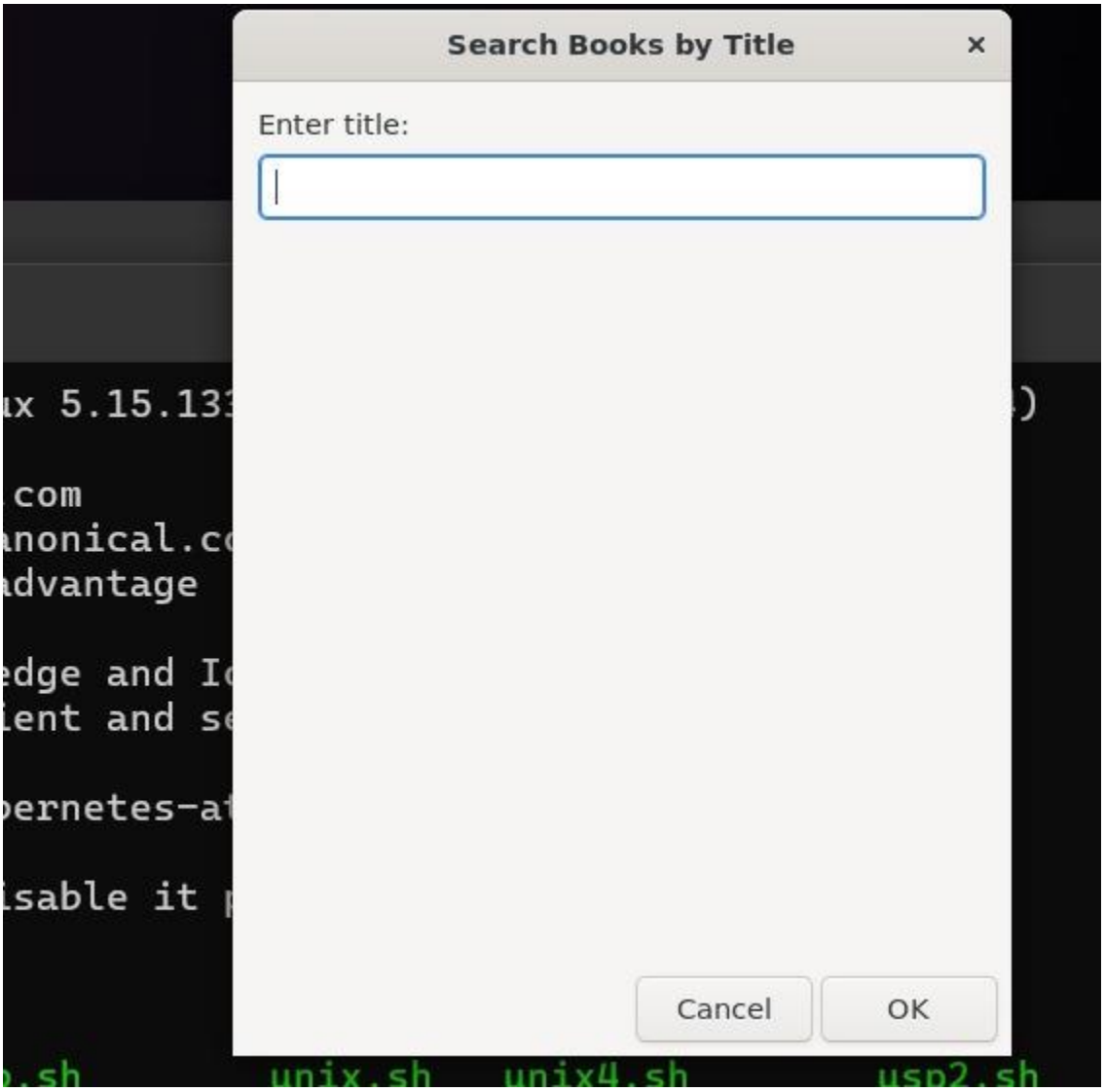

Login Type
Select items from the list below.
Select login type:
Author
User
Register New User/Author
Cancel
OK

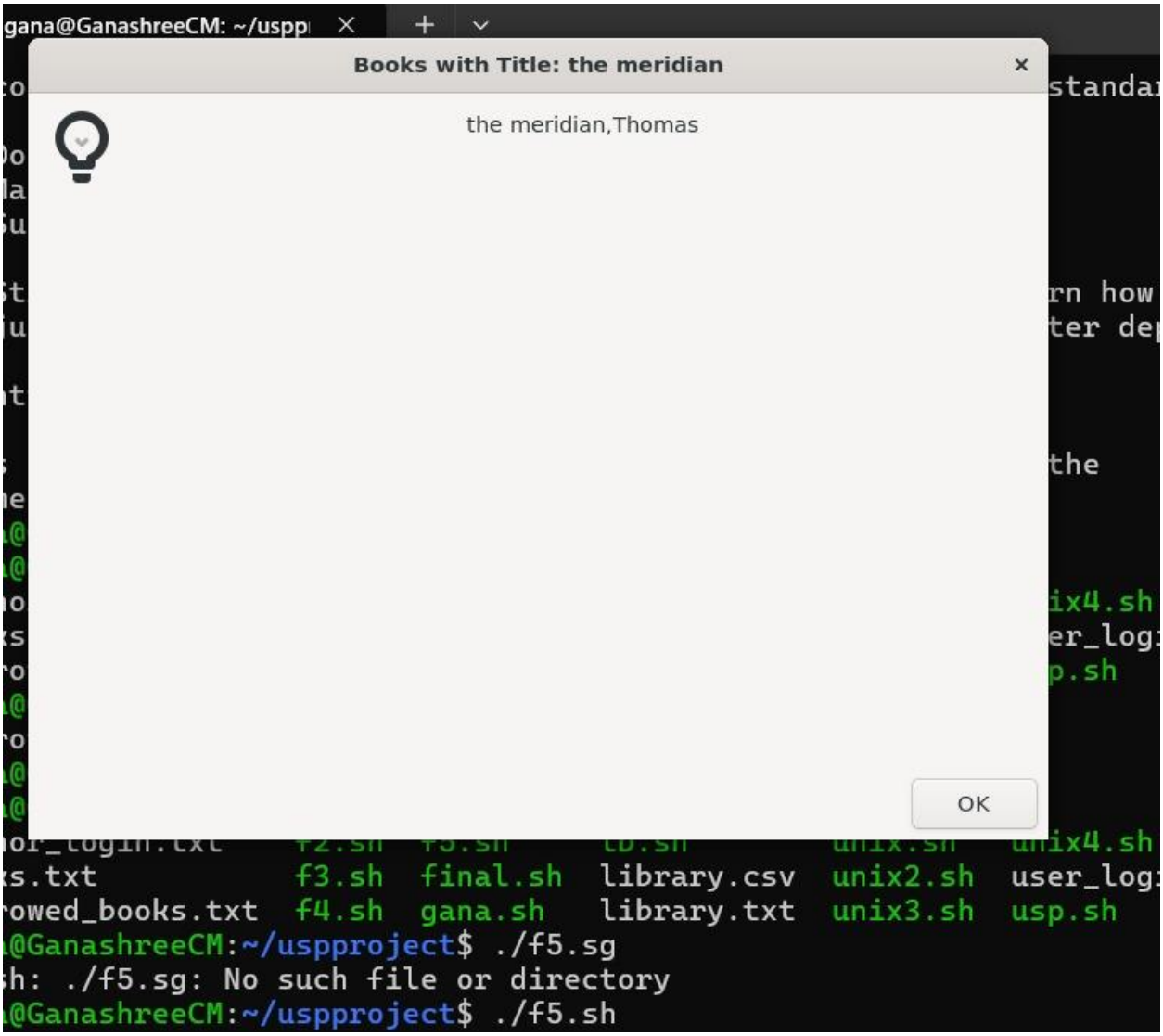

User Login
Enter user username:
Cancel
OK

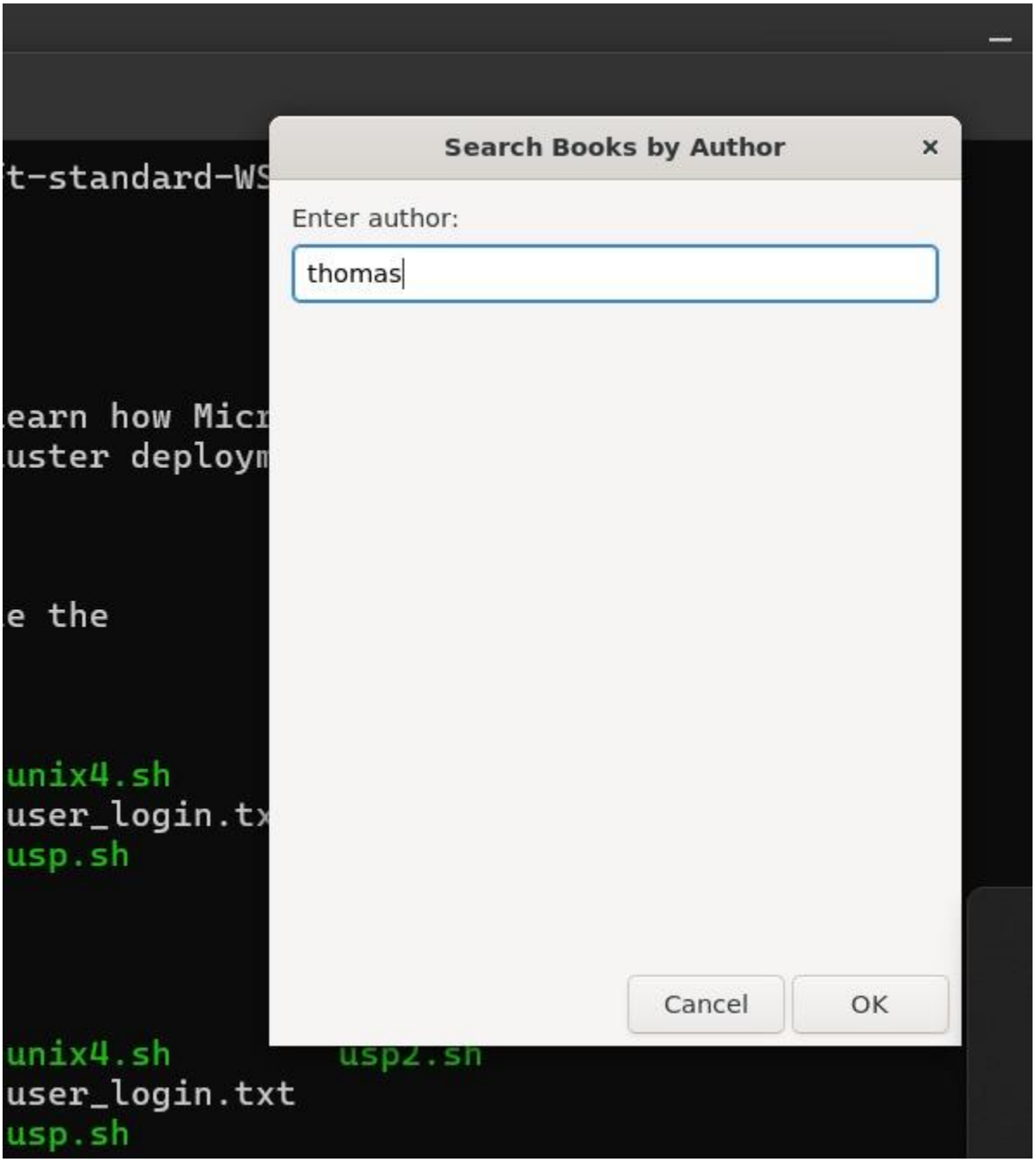

* Support: https://ubuntu.com/advantage
* Strictly confined Kubernetes makes edge and I
Information
User authentication successful!
OK
tailwindcss

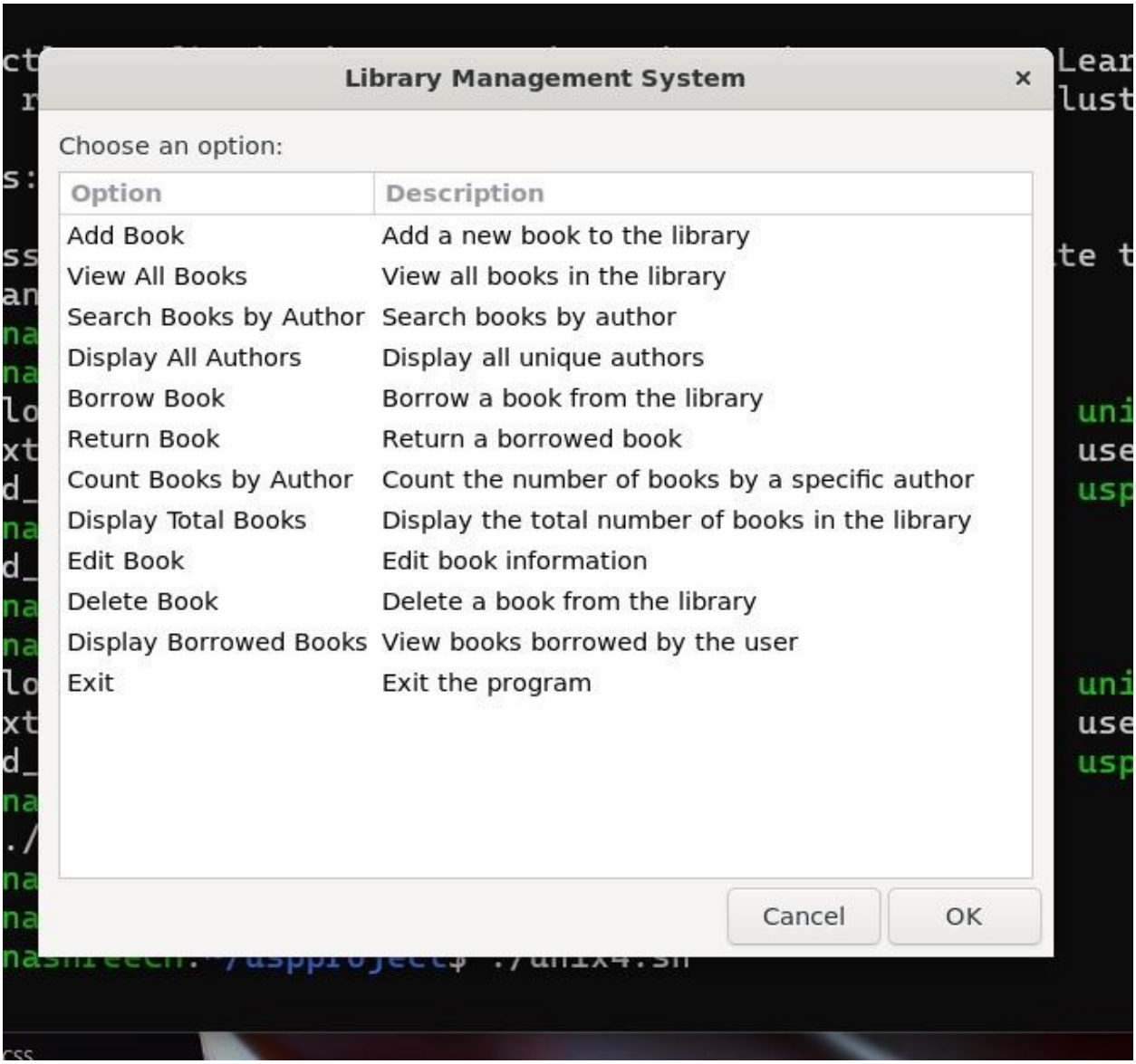

Error
Only authors can add books. Please log in as an author.
OK
borrowed_books.txt f4.sh gana.sh library.
gana@GanashreeCM:~/uspproject$ borrowed_books

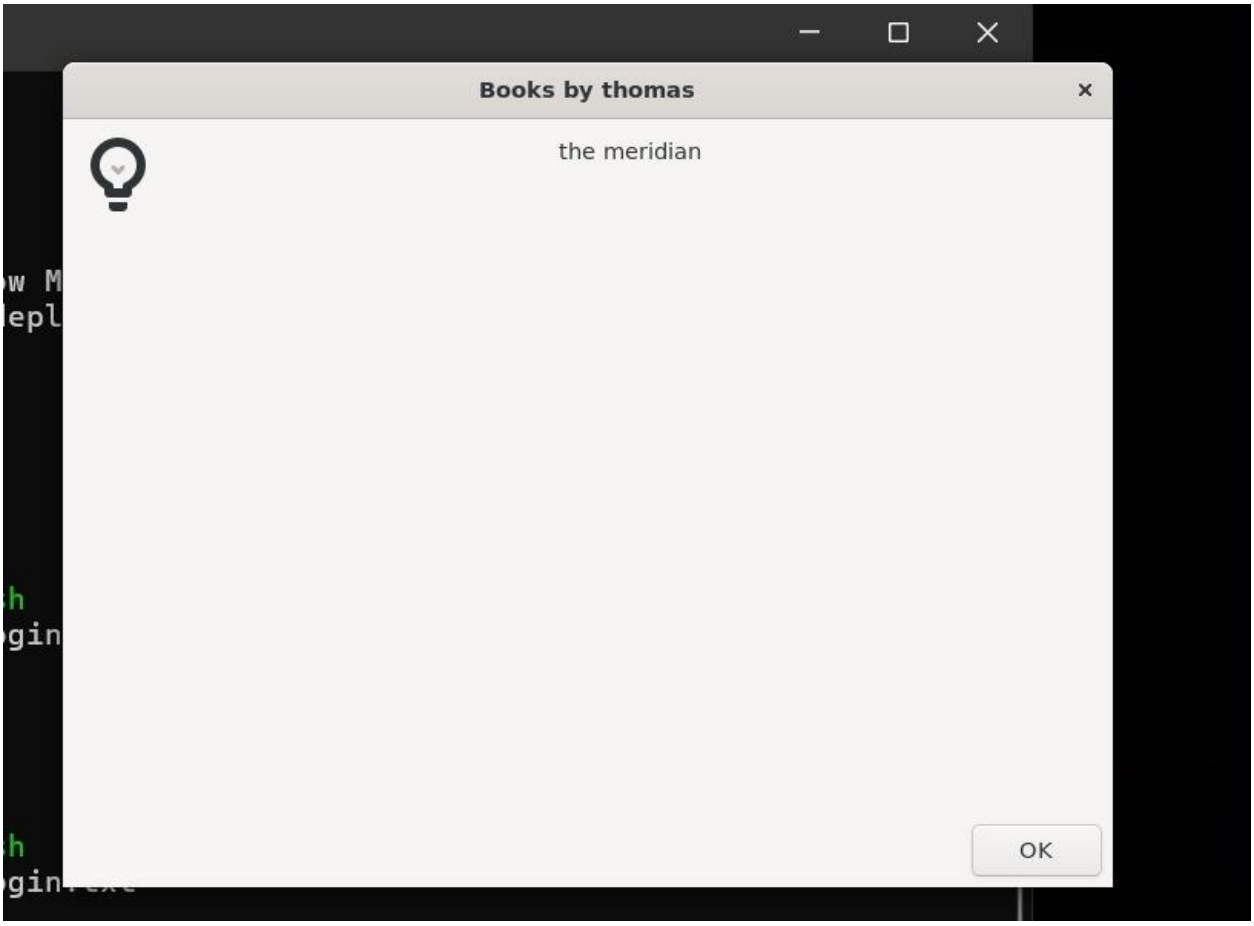

View All Books
the meridian,Thomas
dune, Frank
Can't hurt me, David Goggins
Cancel
OK

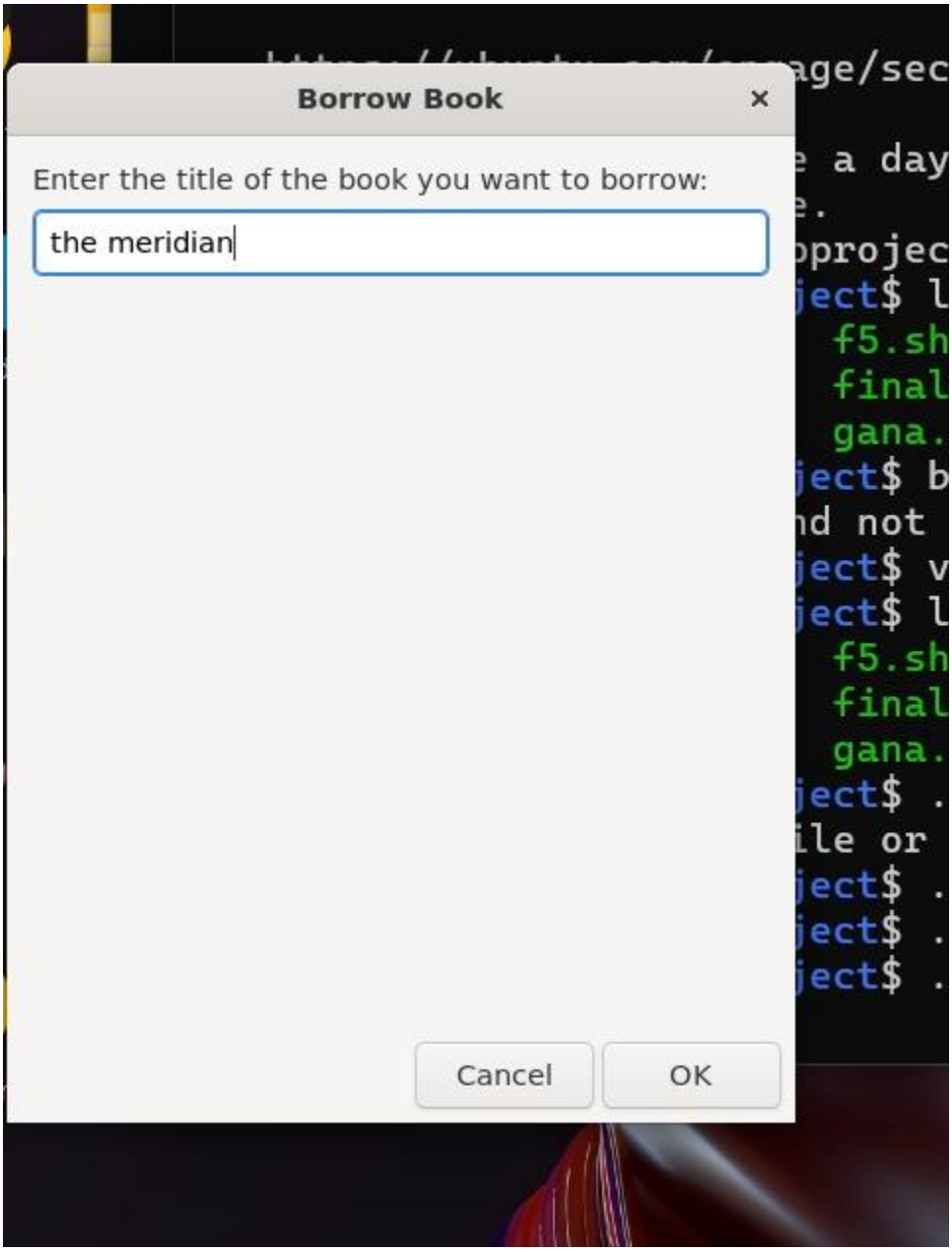

Search Books by Title
Enter title:
Cancel
OK

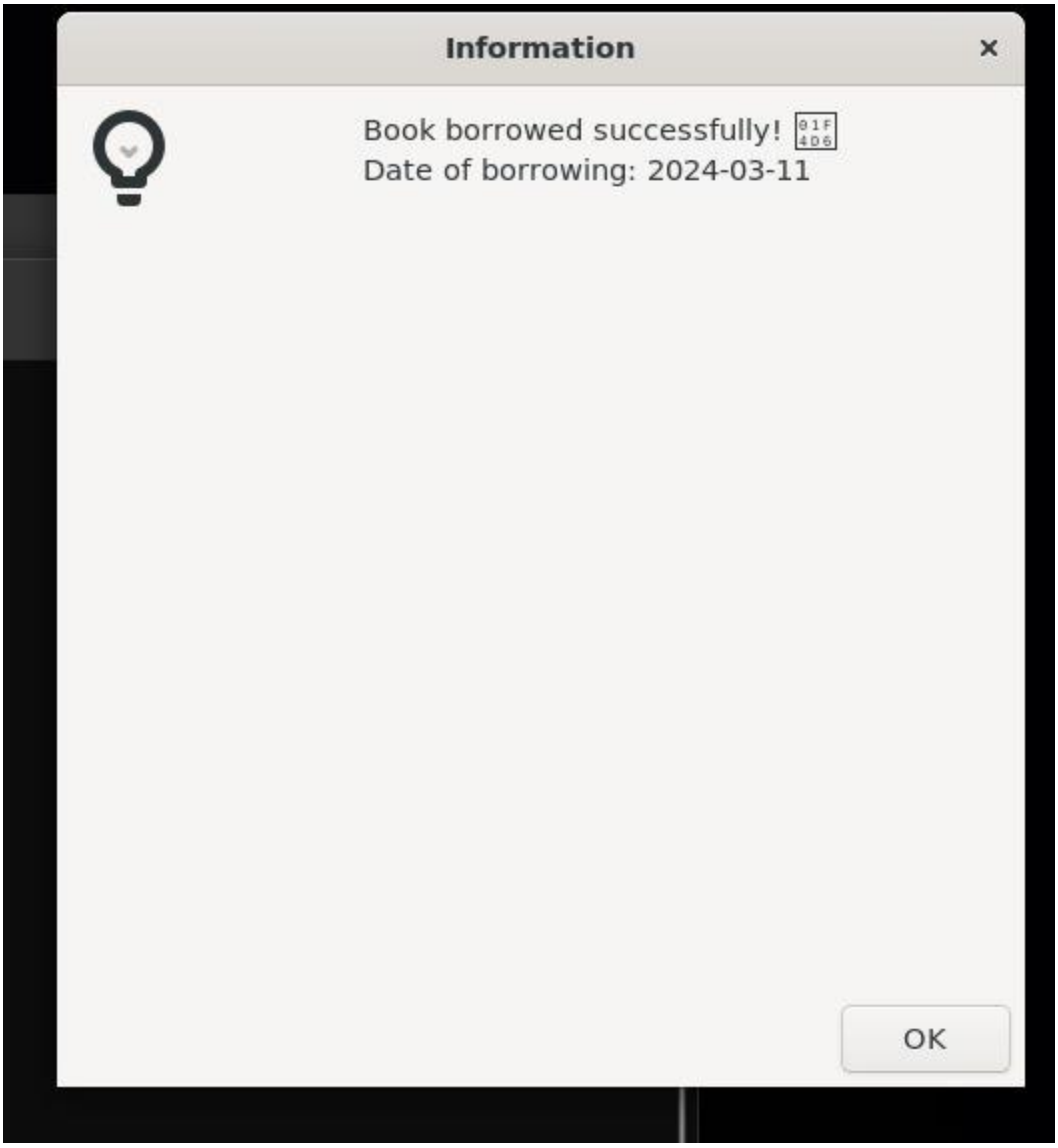

gana@GanashreeCM: ~/uspp
Books with Title: the meridian
the meridian,Thomas
OK
standa
rn how
ter dep
the
ix4.sh
er_logi
p.sh
ix4.sh
s.txt f3.sh final.sh library.csv unix2.sh user_logi
owed_books.txt f4.sh gana.sh library.txt unix3.sh usp.sh
@GanashreeCM:~/uspproject$ ./f5.sg
h: ./f5.sg: No such file or directory
@GanashreeCM:~/uspproject$ ./f5.sh

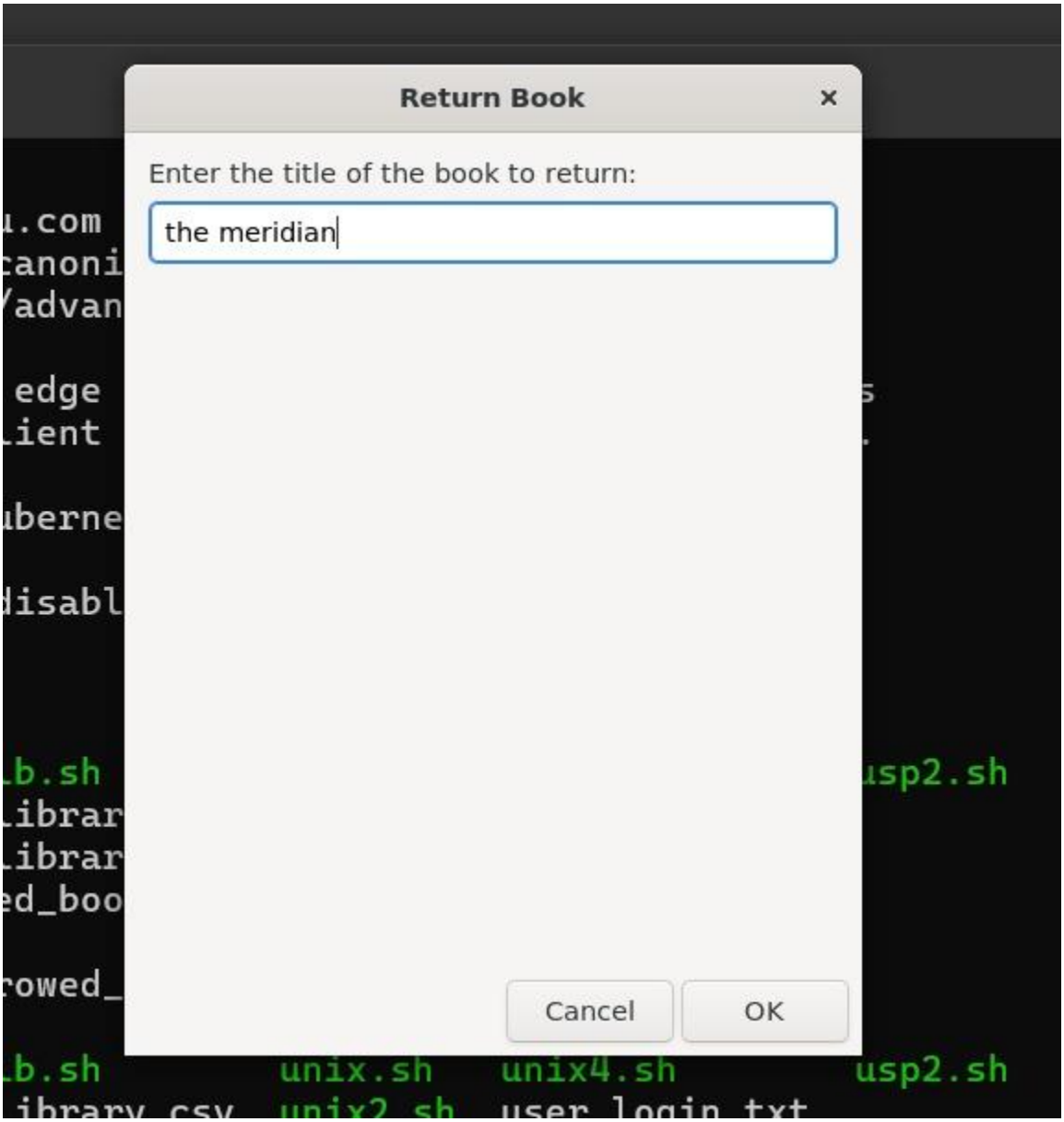

Search Books by Author
Enter author:
thomas
Cancel
OK
unix4.sh
user_login.txt
usp.sh
usp2.sh

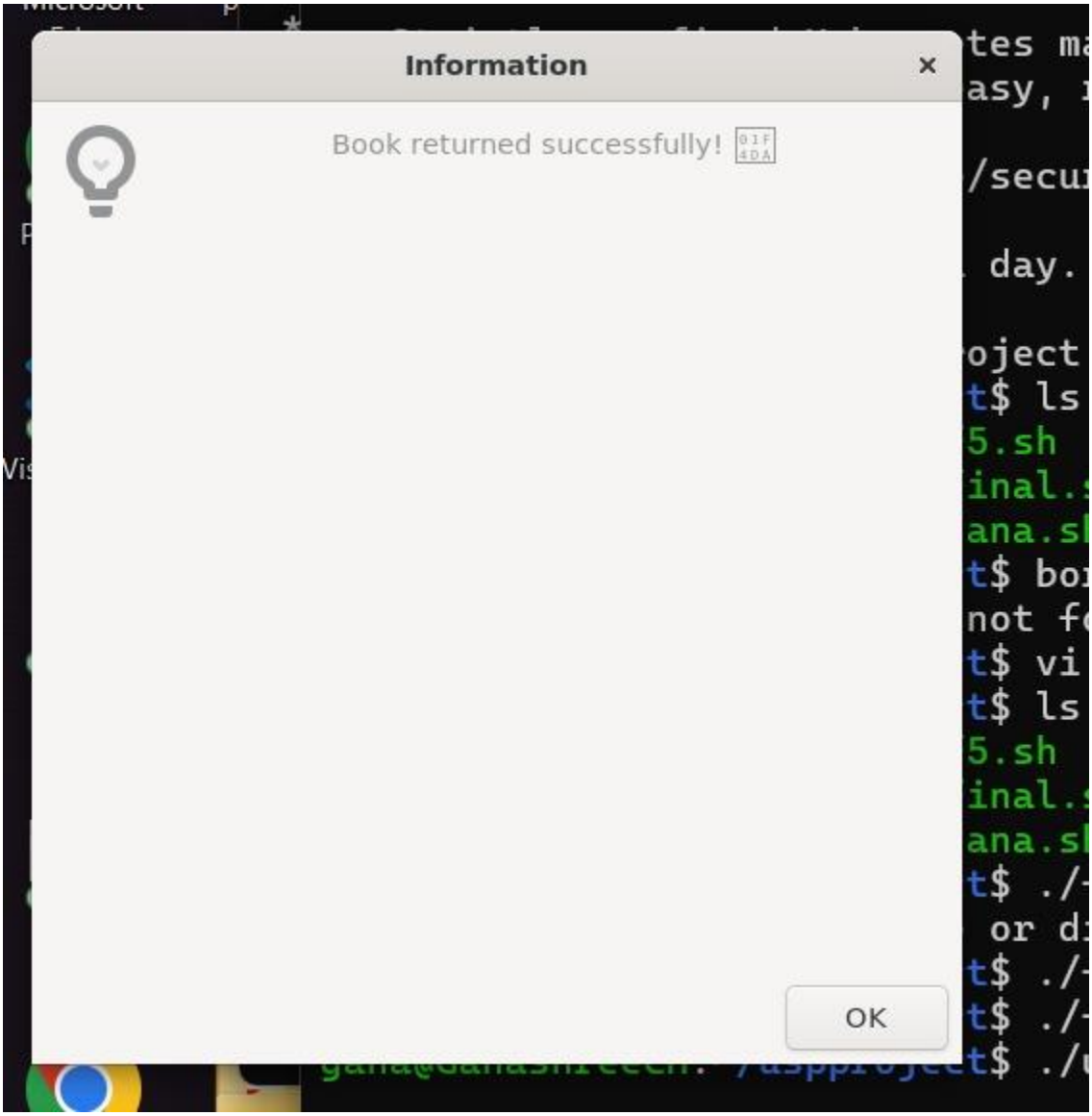

Library Management System
Choose an option:
Option
Description
Add Book
Add a new book to the library
View All Books
View all books in the library
Search Books by Author
Search books by author
Display All Authors
Display all unique authors
Borrow Book
Borrow a book from the library
Return Book
Return a borrowed book
Count Books by Author
Count the number of books by a specific author
Display Total Books
Display the total number of books in the library
Edit Book
Edit book information
Delete Book
Delete a book from the library
Display Borrowed Books
View books borrowed by the user
Exit
Exit the program
Cancel
OK

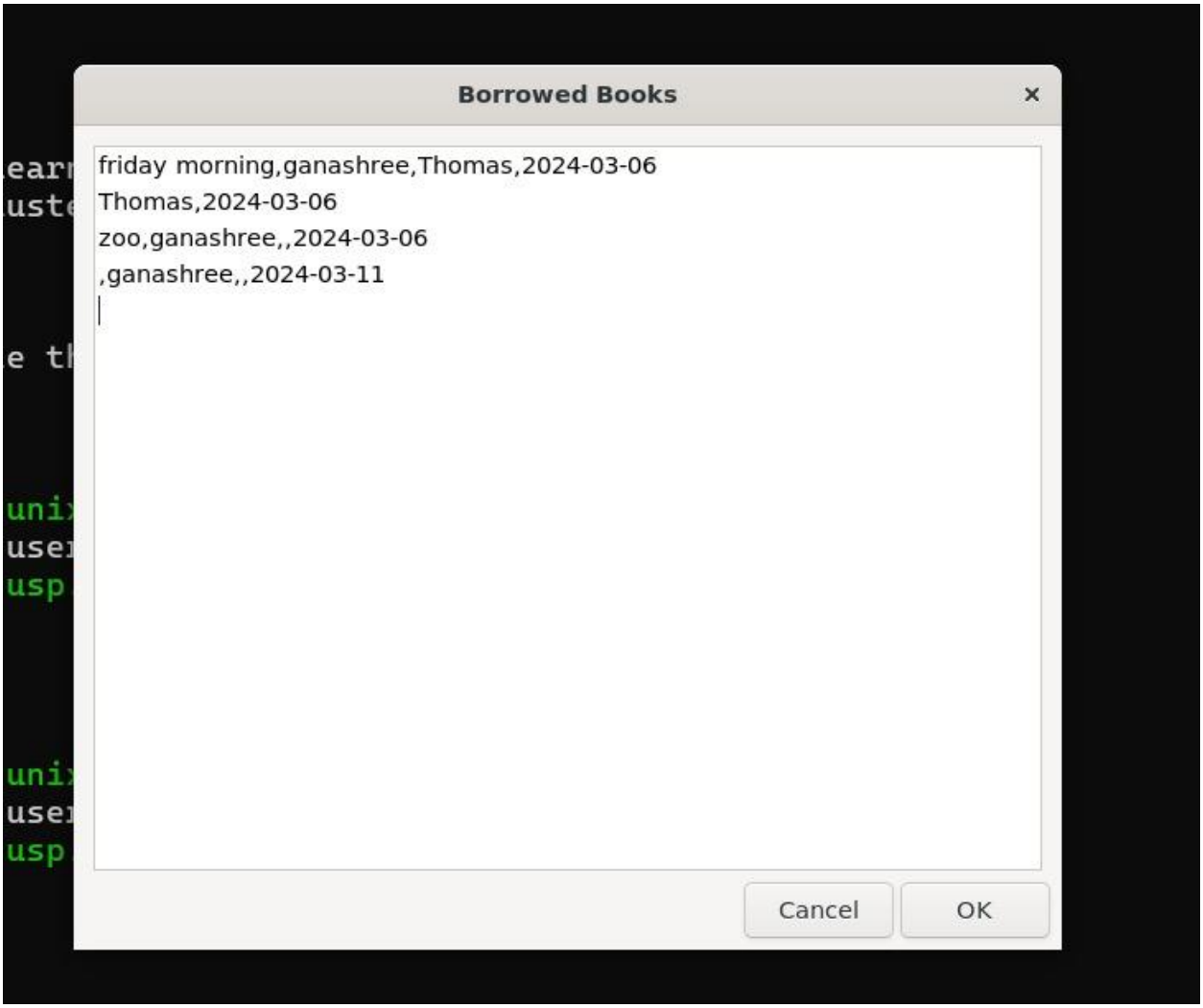

Books by thomas
the meridian
OK

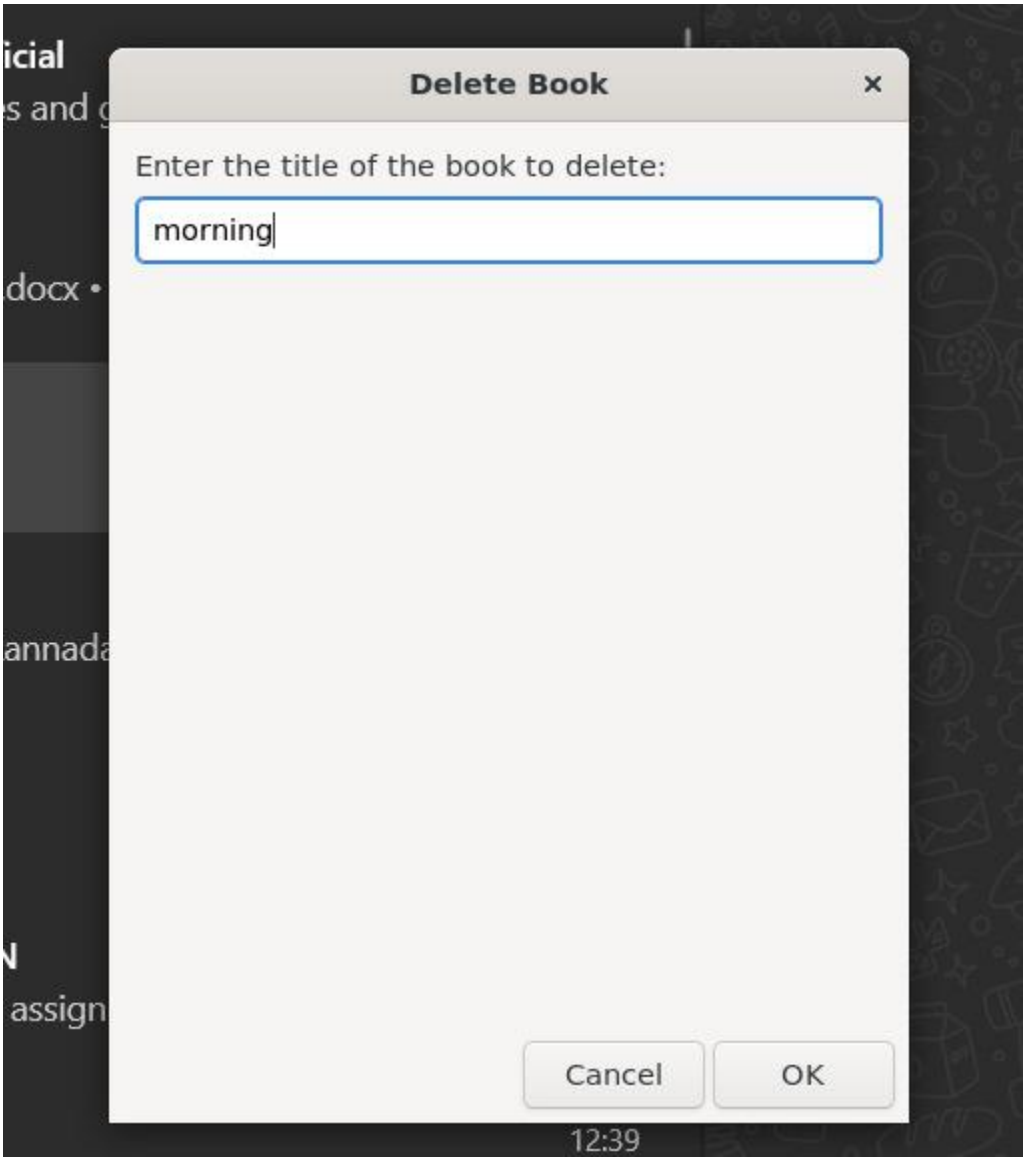

Borrow Book
Enter the title of the book you want to borrow:
the meridian
Cancel
OK

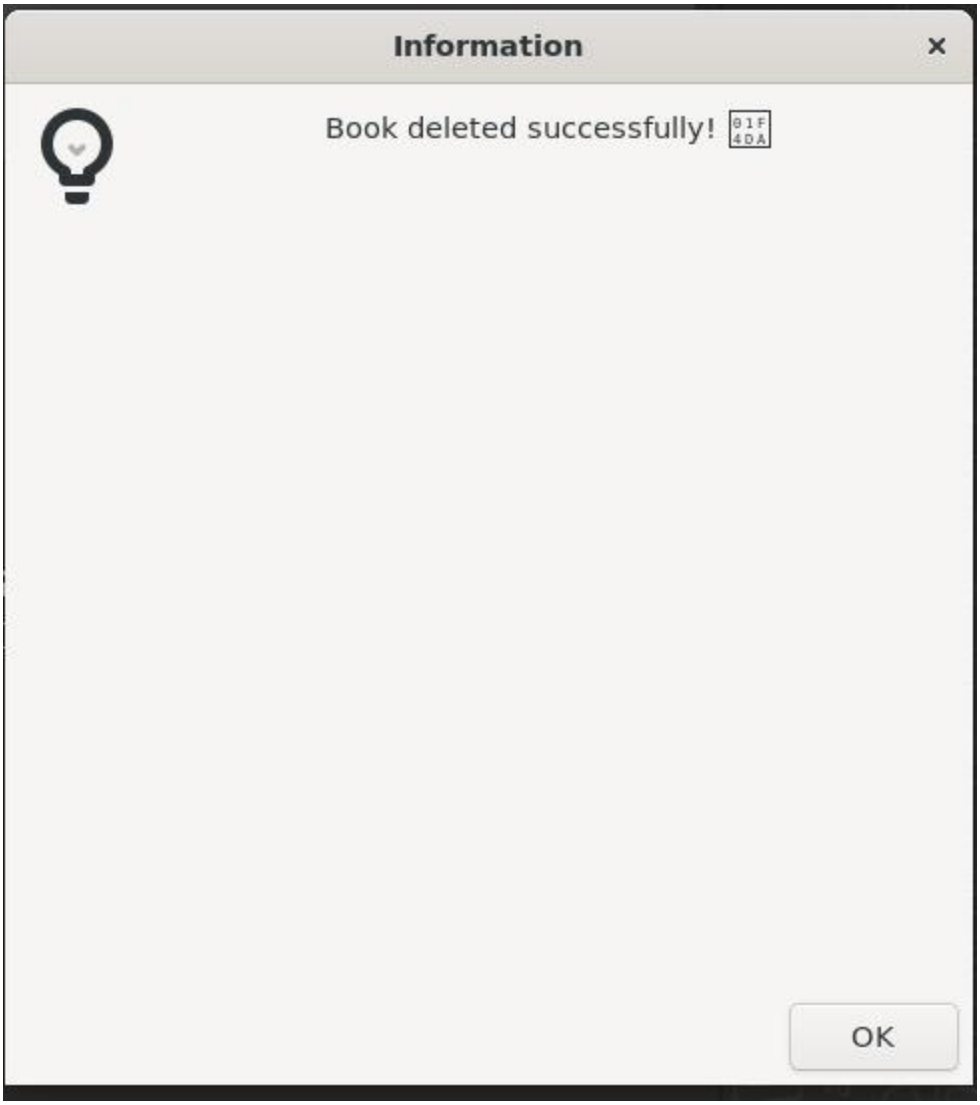

Information
Book borrowed successfully!
Date of borrowing: 2024-03-11
OK

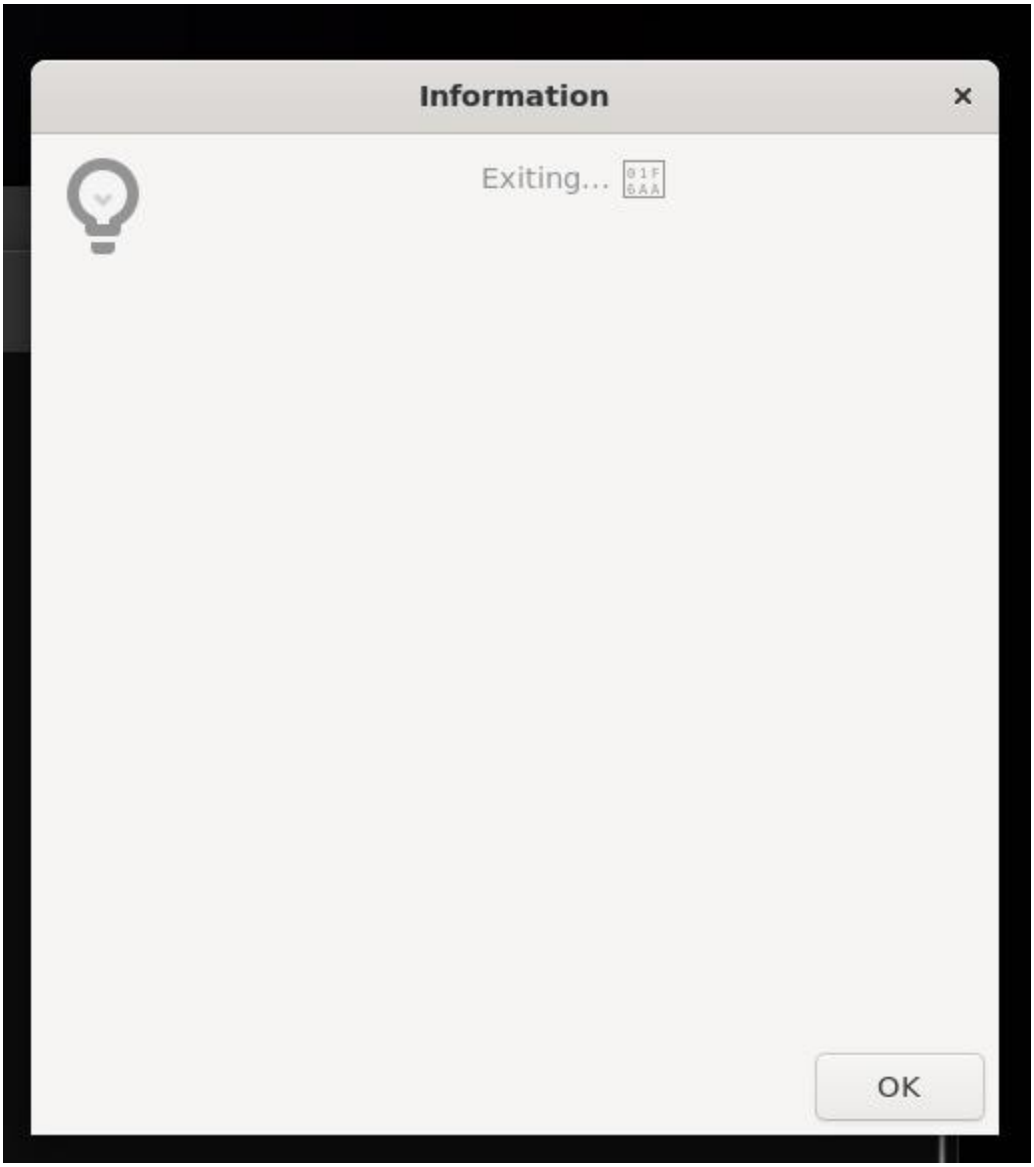

Return Book
Enter the title of the book to return:
the meridian
Cancel
OK
unix.sh
unix4.sh
usp2.sh


Information
Book returned successfully!
OK


Borrowed Books
friday morning,ganashree,Thomas,2024-03-06
Thomas,2024-03-06
zoo,ganashree,,2024-03-06
,ganashree,,2024-03-11
Cancel
OK


Delete Book
Enter the title of the book to delete:
morning
Cancel
OK
12:39


Information
Book deleted successfully!
OK


Information
Exiting...
OK

5.Conclusion

In conclusion, the implementation of the UNIX shell script for a library management system provides a solid foundation for managing library operations efficiently. By leveraging various UNIX commands and shell scripting techniques, the script offers essential functionalities such as user registration, authentication, book management, and user interactions through a graphical user interface.

The script demonstrates the power and versatility of shell scripting in automating tasks and handling data manipulation effectively. It underscores the importance of utilizing UNIX commands in conjunction with scripting to create robust and user-friendly solutions for real-world applications.

While the current implementation fulfills basic requirements for library management, there is room for further improvement and expansion. Future iterations of the script could include additional features such as advanced search capabilities, book reservations, user feedback mechanisms, and administrative tools to enhance usability and functionality.

Overall, the library management system implemented in the UNIX shell script showcases the potential of shell scripting in building practical solutions for managing information and streamlining workflows. With continued refinement and innovation, it has the potential to evolve into a comprehensive tool for libraries and organizations to efficiently manage their resources and serve their users effectively.

6.References

1. Stallings, W. (2014). "Operating Systems: Internals and Design Principles." Pearson.

2. Robbins, A., & Robbins, A. H. (2005). "UNIX Systems Programming: Communication, Concurrency and Threads." Pearson Education.

ONLINE REFERENCES:

3. Bash Reference Manual. (n.d.). Retrieved from https://www.gnu.org/software/bash/manual/

4. Zenity Manual. (n.d.). Retrieved from https://help.gnome.org/users/zenity/stable/