Library Management

System

**Utkarsh Tyagi**

Page Writeup

Title: Efficient Library Management System

Introduction: The Library Management System (LMS) project is a comprehensive solution designed to streamline and enhance the operations of a library, ensuring efficient management of resources, easy accessibility of information, and an improved user experience. Developed with a user-centric approach, the system aims to automate traditional library processes, making them more time-efficient and error-free.

Key Features:

1. **User-friendly Interface:** The LMS boasts an intuitive and user-friendly interface, making it easy for both library staff and patrons to navigate through accessibility for users of all technical backgrounds.
2. **Book Catalog Management:** The heart of the system lies in its ability to manage the library's extensive book catalog. Librarians can efficiently add, edit, and delete book records, including details such as title, author, genre, and availability status. The system also supports ISBN integration for swift cataloging.
3. **Borrowing and Returning:** The LMS simplifies the borrowing and returning process, allowing users to check the availability of books in real-time. Automated notifications remind users of upcoming due dates, reducing the likelihood of overdue materials.
4. **User Authentication and Access Control:** Security is a top priority, with robust user authentication protocols in place. Access levels are defined for various user roles, such as librarians, administrators, and patrons, ensuring that sensitive information remains secure.
5. **Search:** The system incorporates advanced search and filtering capabilities, enabling users to quickly locate specific books based on criteria such as title, author, or genre. This feature significantly enhances the overall efficiency of the library.
6. **Reports and Analytics:** Librarians can generate detailed reports and analytics on various aspects of library operations, such as book circulation, user activity, and overdue items. These insights aid in informed decision-making and resource allocation.
7. **Integration with External Systems:** To facilitate seamless data flow, the LMS supports integration with external systems, such as online databases, allowing librarians to stay updated with the latest book information and trends.
8. **Mobile Accessibility:** Recognizing the growing reliance on mobile devices, the LMS includes a mobile-responsive design, enabling users to access the system on the go. This promotes convenience for both library staff and patrons.

Conclusion: In conclusion, the Library Management System is a powerful tool that revolutionizes the traditional library experience. By combining efficiency, security, and user-friendliness, the system empowers librarians to better serve their patrons and ensures a more enjoyable and accessible library experience for all.

Code

Main file

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include "admin.h"

#include "user.h"

int reg(){

    char username[50], name[50], ch[50], fname;

    FILE\*p=fopen("users.txt","r");

    printf("\t\t\tEnter username to create acccount.");

    scanf("\t\t\t%s",username);

    while(fgets(ch,sizeof(ch),p)!=NULL){

        ch[strcspn(ch,"\n")]='\0';

        if(strcmp(ch,username)==0){

            fclose(p);

            printf("\t\t\tUsername already exists.\n");

            reg();

        }

    }

    fclose(p);

    p=fopen("users.txt", "a");

    fprintf(p,"%s\n", username);

    fclose(p);

    adduserData(username,"Null","Null",0);

    printf("\t\t\tSuccessfully Registered\n");

    main();

    return 0;

}

int login(){

    char username[50], ch[50];

    int found=0;

    FILE\*p=fopen("users.txt","r");

    printf("\t\t\tEnter username: ");

    scanf("\t\t\t%s",username);

    while (fgets(ch,sizeof(ch),p)!=NULL){

        ch[strcspn(ch,"\n")]='\0';

        if (strcmp(ch,username)==0){

            found=1;

            break;

        }

    }

    fclose(p);

    if(found==1){

        if(strcmp(username,"Admin")==0){

            adminmenu();

            main();

        }

        else{

            printf("\n\n\n\t\t\tWelcome %s",username);

            usermenu();

            main();

        }

    }

    else{

        printf("\n\t\t\tInvalid ID.\n");

        login();

    }

}

void project\_details(){

    printf("\t\t\tProject:Library Management System \n\t\t\tBy:Utkarsh Tyagi\n\t\t\tClass:3-C");

}

int main(){

    printf("\n\n\n\n\n\n\n\n\n\n");

    printf("\t\t\t\t\t\t\t\tLIBRARY MANAGEMENT SYSTEM\t\t\t\t\t\t\t\t\n");

    printf("\n\n\n\n\n\n");

    int n=0;

    do{

        printf("\t\t\tPress 1 to Register\n\t\t\tPress 2 to Login\n\t\t\tPress 3 to view 'Project Details'\n");

        scanf("\t\t\t%d",&n);

        if(n==1){

            reg();

        }

        else if(n==2){

            login();

            return 0;

        }

        else if(n==3){

            project\_details();

            main();

        }

        else{

            printf("\t\t\tInvalid choice entered.");

        }

    }while(n<=3);

    return 0;

}

User File

struct node2{

    char bookname[150];

    int qty;

    int issued;

    struct node2\*loc;

};

struct node2\*Start=NULL;

struct usernode2{

    char username[50];

    char book1[100];

    char book2[100];

    int userIssued;

    struct usernode2\*loc;

};

struct usernode2\*Userstart=NULL;

int usermenu(){

    printf("\n\n\n\n\n");

    printf("\t\t\t\t\t\t\t\tUSERS MENU\t\t\t\t\t\t\n");

    printf("\n\n\n");

    int n=0;

    do{

        printf("\n\n\t\t\tPress 1 to view list of Books\n\t\t\tPress 2 to view list of Books issued\n");

        printf("\t\t\tPress 3 to check available Books\n\t\t\tPress 0 to return to main menu\n");

        scanf("\t\t\t%d",&n);

        if(n==0){

            return 0;

        }

        else if(n==1){

            showBooks();

        }

        else if(n==2){

            viewIssued();

        }

        else if(n==3){

            viewStatus();

        }

        else{

            printf("\t\t\tInvalid choice.");

        }

    }while(n<=3);

    return 0;

}

int showBooks(){

    char booknqty[200];

    char name[100];

    char status[100];

    int qty,issued;

    FILE\*p=fopen("books.txt","r");

    if(p==NULL){

        printf("Error");

        return 0;

    }

    printf("\n\t\t\tBOOK \t\t\tSTATUS\n\n");

    while(fgets(booknqty,sizeof(booknqty),p)!=NULL){

        if(sscanf(booknqty,"%s %d %d",name,&qty,&issued)==3) {

            if(qty>issued){

                strcpy(status,"Available");

                printf("\t\t\t%s\t\t\t%s\n",name,status);

            }

            else{

                strcpy(status,"Not Available");

                printf("\t\t\t%s\t\t\t%s\n",name,status);

            }

        }

    }

    fclose(p);

    return 0;

}

int viewIssued() {

    char username[50];

    struct usernode2\* userStart = NULL;

    struct usernode2\* userEnd = NULL;

    FILE\* p = fopen("usersdata.txt", "r");

    if (p == NULL) {

        printf("\n\t\t\tError\n\n");

        return 0;

    }

    char userData[300];

    while (fgets(userData, sizeof(userData), p) != NULL) {

        char exusername[50], exbook1[100], exbook2[100];

        int exuserIssued;

        if (sscanf(userData, "%s %s %s %d", exusername, exbook1, exbook2, &exuserIssued) == 4) {

            struct usernode2\* temp = (struct usernode2\*)malloc(sizeof(struct usernode2));

            strcpy(temp->username, exusername);

            strcpy(temp->book1, exbook1);

            strcpy(temp->book2, exbook2);

            temp->userIssued = (exuserIssued > 0) ? 1 : 0;

            temp->loc = NULL;

            if (userEnd == NULL) {

                userStart = temp;

            }

            else {

                userEnd->loc = temp;

            }

            userEnd = temp;

        }

    }

    fclose(p);

    printf("\t\t\tEnter your username: ");

    scanf("%s", username);

    struct usernode2\* ucurrent = userStart;

    while (ucurrent != NULL) {

        if (strcmp(ucurrent->username, username) == 0) {

            if (ucurrent->userIssued == 0) {

                printf("\n\t\t\tNo books have been issued\n");

                return 0;

            }

            else if (ucurrent->userIssued == 1) {

                printf("\n\t\t\t1 book is issued.\n");

                printf("\t\t\tName of the book: %s\n", ucurrent->book1);

                return 0;

            }

            else if (ucurrent->userIssued == 2) {

                printf("\n\t\t\t2 books are issued. Only one book can be issued at a time.\n");

                printf("\t\t\tName of the book: %s\n", ucurrent->book1);

                return 0;

            }

            break;

        }

        ucurrent = ucurrent->loc;

    }

    return 0;

}

int viewStatus(){

    char name[100];

    printf("\t\t\tEnter Book: ");

    scanf("%s",name);

    struct node2\*bookStart=NULL;

    struct node2\*bookEnd=NULL;

    FILE\*p=fopen("books.txt","r");

    if(p==NULL){

        addBooks();

        return 0;

    }

    char bookData[200];

    while(fgets(bookData,sizeof(bookData),p)!=NULL){

        char exname[100];

        int exqty,exissued;

        if(sscanf(bookData,"%s %d %d",exname,&exqty,&exissued)==3){

            struct node2\*temp=(struct node2\*)malloc(sizeof(struct node2));

            strcpy(temp->bookname,exname);

            temp->qty=exqty;

            temp->issued=exissued;

            temp->loc=NULL;

            if(bookEnd==NULL){

                bookStart=temp;

            }

            else{

                bookEnd->loc=temp;

            }

            bookEnd=temp;

        }

    }

    fclose(p);

    struct node2\*current=bookStart;

    while(current!=NULL) {

        if(strcmp(current->bookname,name)==0) {

            if(current->issued<current->qty){

                printf("\n\t\t\tBook is available.\n\n");

            }

            else{

                printf("\n\t\t\tBook is not available.\n\n");

            }

            return 0;

        }

        current=current->loc;

    }

    printf("\n\t\t\tBook is not available.\n\n");

    return 0;

}

Admin File

struct usernode{

    char username[50];

    char book1[100];

    char book2[100];

    int userIssued;

    struct usernode\*loc;

};

struct usernode\*userstart=NULL;

adduserData(char username[], char book1[],char book2[], int userIssued){

    struct usernode\*t1=(struct usernode\*)malloc(sizeof(struct usernode));

    struct usernode\*t=NULL;

    userstart=t1;

    strcpy(t1->username,username);

    strcpy(t1->book1,book1);

    strcpy(t1->book2,book2);

    t1->userIssued=userIssued;

    FILE\*p=fopen("usersdata.txt","a");

    fprintf(p,"%s %s %s %d\n",username,book1,book2,userIssued);

    fclose(p);

    t1->loc=(struct usernode\*)malloc(sizeof(struct usernode));

    t=t1;

    t1=t1->loc;

    free(t1);

    t->loc=NULL;

    return 0;

}

int adminmenu(){

    int n=0;

    printf("\n\n\n\n\n");

    printf("\t\t\t\t\t\t\t\tADMIN MENU\t\t\t\t\t\t\n");

    printf("\n\n\n");

    do{

        printf("\n\t\t\tPress 1 to view list of Books in the Library\n\t\t\tPress 2 to add new Books in the Library\n");

        printf("\t\t\tPress 3 to delete a Book from the Library\n\t\t\tPress 4 to check Book status\n");

        printf("\t\t\tPress 5 to issue a Book to a user\n\t\t\tPress 6 to process return from a user\n");

        printf("\t\t\tPress 0 to return to main menu\n");

        scanf("\t\t\t%d",&n);

        if(n==0){

            return 0;

        }

        else if(n==1){

            displayBooks();

        }

        else if(n==2){

            addBooks();

        }

        else if(n==3){

            deleteBook();

        }

        else if(n==4){

            checkStatus();

        }

        else if(n==5){

            issueBook();

        }

        else if(n==6){

            returnBook();

        }

        else{

            printf("\t\t\tInvalid");

        }

    }while(n<=6);

    return 0;

}

struct node{

    char bookname[150];

    int qty;

    int issued;

    struct node\*loc;

};

struct node\*start=NULL;

int updateQty(char name[], int newqty) {

    struct node\* bookStart = NULL;

    struct node\* bookEnd = NULL;

    FILE\* p = fopen("books.txt", "r");

    if (p == NULL) {

        printf("Error opening file for reading\n");

        return -1;

    }

    char bookData[200];

    while (fgets(bookData, sizeof(bookData), p) != NULL) {

        char exname[100];

        int exqty, exissued;

        if (sscanf(bookData, "%s %d %d", exname, &exqty, &exissued) == 3) {

            struct node\* temp = (struct node\*)malloc(sizeof(struct node));

            strcpy(temp->bookname, exname);

            temp->qty = exqty;

            temp->issued = exissued;

            temp->loc = NULL;

            if (bookEnd == NULL) {

                bookStart = temp;

            } else {

                bookEnd->loc = temp;

            }

            bookEnd = temp;

        }

    }

    fclose(p);

    int bookPresent = 0;

    struct node\* current = bookStart;

    while (current != NULL) {

        if (strcmp(current->bookname, name) == 0) {

            bookPresent = 1;

            current->qty += newqty;

            break;

        }

        current = current->loc;

    }

    if (bookPresent == 0) {

        return -1;

    } else {

        p = fopen("books.txt", "w");

        if (p == NULL) {

            printf("Error\n");

            return -1;

        }

        current = bookStart;

        while (current != NULL) {

            fprintf(p, "%s %d %d\n", current->bookname, current->qty, current->issued);

            struct node\* temp = current;

            current = current->loc;

            free(temp); // Free memory allocated for the node

        }

        fclose(p);

        return 0;

    }

}

int addBooks() {

    int num, i, qty, issued, cont;

    char name[100];

    struct node\* t1 = (struct node\*)malloc(sizeof(struct node));

    struct node\* t = NULL;

    if (t1 == NULL) {

        printf("failed\n");

        return -1;

    }

    start = t1;

    printf("\t\t\tHow many Books do you wish to add? ");

    scanf("%d", &num);

    for (i = 0; i < num; i++) {

        printf("\t\t\tEnter name of Book: ");

        scanf("%s", name);

        printf("\t\t\tEnter quantity: ");

        scanf("%d", &qty);

        cont = updateQty(name, qty);

        if (cont == -1) {

            strcpy(t1->bookname, name);

            t1->qty = qty;

            issued = 0;

            t1->issued = issued;

            FILE\* p = fopen("books.txt", "a");

            if (p == NULL) {

                printf("Error opening file for writing\n");

                return -1;

            }

            fprintf(p, "%s %d %d\n", name, qty, issued);

            fclose(p);

            t1->loc = (struct node\*)malloc(sizeof(struct node));

            if (t1->loc == NULL) {

                printf("Memory allocation failed\n");

                return -1;

            }

            t = t1;

            t1 = t1->loc;

        }

    }

    free(t1);

    if (t != NULL) {

        t->loc = NULL;

    }

    printf("\t\t\tAdded successfully\n\n");

    return 0;

}

int displayBooks(){

    char booknqty[200];

    char name[100];

    int qty,issued;

    FILE\*p=fopen("books.txt","r");

    if(p==NULL){

        printf("Error");

        return 0;

    }

    printf("\n\t\t\tBOOK NAME\t\t\tQUANTITY\t\t\tISSUED\n\n");

    while(fgets(booknqty,sizeof(booknqty),p)!=NULL){

        if(sscanf(booknqty,"%s %d %d",name,&qty,&issued)==3) {

            printf("\t\t\t%s\t\t\t\t%d\t\t\t%d\n",name,qty,issued);

        }

    }

    fclose(p);

    return 0;

}

int deleteBook(){

    char name[100];

    printf("\t\t\tEnter name of Book to be removed: ");

    scanf("%s",name);

    struct node\*bookStart=NULL;

    struct node\*bookEnd=NULL;

    FILE\*p=fopen("books.txt","r");

    if(p==NULL){

        addBooks();

        return 0;

    }

    char bookData[200];

    while(fgets(bookData,sizeof(bookData),p)!=NULL){

        char exname[100];

        int exqty,exissued;

        if(sscanf(bookData,"%s %d %d",exname,&exqty,&exissued)==3){

            struct node\*temp=(struct node\*)malloc(sizeof(struct node));

            strcpy(temp->bookname,exname);

            temp->qty=exqty;

            temp->issued=exissued;

            temp->loc=NULL;

            if(bookEnd==NULL){

                bookStart=temp;

            }

            else{

                bookEnd->loc=temp;

            }

            bookEnd=temp;

        }

    }

    fclose(p);

    int bookPresent=0;

    struct node\*current=bookStart;

    while(current!=NULL) {

        if(strcmp(current->bookname,name)==0) {

            bookPresent=1;

            break;

        }

        current=current->loc;

    }

    if(bookPresent==0){

        printf("\n\t\t\tNo such book was present\n\n");

        return 0;

    }

    else{

        p=fopen("books.txt","w");

        current=bookStart;

        while(current!=NULL){

            if(strcmp(current->bookname,name)!=0){

                fprintf(p,"%s %d %d\n",current->bookname,current->qty,current->issued);

                struct node\*temp=current;

            }

            current=current->loc;

        }

        fclose(p);

        printf("\t\t\tDeleted successfully\n\n");

        return 0;

    }

}

int checkStatus(){

    char name[100];

    printf("\t\t\tEnter name of Book: ");

    scanf("%s",name);

    struct node\*bookStart=NULL;

    struct node\*bookEnd=NULL;

    FILE\*p=fopen("books.txt","r");

    if(p==NULL){

        addBooks();

        return 0;

    }

    char bookData[200];

    while(fgets(bookData,sizeof(bookData),p)!=NULL){

        char exname[100];

        int exqty,exissued;

        if(sscanf(bookData,"%s %d %d",exname,&exqty,&exissued)==3){

            struct node\*temp=(struct node\*)malloc(sizeof(struct node));

            strcpy(temp->bookname,exname);

            temp->qty=exqty;

            temp->issued=exissued;

            temp->loc=NULL;

            if(bookEnd==NULL){

                bookStart=temp;

            }

            else{

                bookEnd->loc=temp;

            }

            bookEnd=temp;

        }

    }

    fclose(p);

    struct node\*current=bookStart;

    while(current!=NULL) {

        if(strcmp(current->bookname,name)==0) {

            if(current->issued<current->qty){

                printf("\n\t\t\tBook is available\n\n");

            }

            else{

                printf("\n\t\t\tBook is currently not available\n\n");

            }

            return 0;

        }

        current=current->loc;

    }

    printf("\n\t\t\tNo such book is available.\n\n");

    return 0;

}

int updateuserData(char username[],char bookname[],int issORret){

    struct usernode\*userStart=NULL;

    struct usernode\*userEnd=NULL;

    FILE\*p=fopen("usersdata.txt","r");

    if(p==NULL){

        printf("\n\t\t\tError\n\n");

        return 0;

    }

    char userData[300];

    while(fgets(userData,sizeof(userData),p)!=NULL){

        char exusername[50],exbook1[100],exbook2[100];

        int exuserIssued;

        if(sscanf(userData,"%s %s %s %d",exusername,exbook1,exbook2,&exuserIssued)==4){

            struct usernode\*temp=(struct usernode\*)malloc(sizeof(struct usernode));

            strcpy(temp->username,exusername);

            strcpy(temp->book1,exbook1);

            strcpy(temp->book2,exbook2);

            temp->userIssued=exuserIssued;

            temp->loc=NULL;

            if(userEnd==NULL){

                userStart=temp;

            }

            else{

                userEnd->loc=temp;

            }

            userEnd=temp;

        }

    }

    fclose(p);

    struct usernode\*current=userStart;

    if(issORret==1){

        p=fopen("usersdata.txt","w");

        while(current!=NULL){

            if(strcmp(current->username,username)==0) {

                current->userIssued+=1;

                if(strcmp(current->book1,"Null")==0){

                    strcpy(current->book1,bookname);

                    strcpy(current->book2,"Null");

                }

                else{

                    strcpy(current->book2,bookname);

                }

            }

            fprintf(p,"%s %s %s %d\n",current->username,current->book1,current->book2,current->userIssued);

            struct node\*temp=current;

            current=current->loc;

        }

        fclose(p);

        printf("\n\t\t\tUser data updated successfully.\n");

        return 0;

    }

    else{

        p=fopen("usersdata.txt","w");

        while(current!=NULL){

            if(strcmp(current->username,username)==0) {

                if(strcmp(current->book1,bookname)==0){

                    strcpy(current->book1,current->book2);

                    strcpy(current->book2,"Null");

                    current->userIssued-=1;

                    printf("\n\t\t\tUser data updated successfully.\n");

                }

                else if(strcmp(current->book2,bookname)==0){

                    strcpy(current->book2,"Null");

                    current->userIssued-=1;

                    printf("\n\t\t\tUser data updated successfully.\n");

                }

                else{

                    printf("\n\t\t\tCurrently, this particular book has not been issued by this user.\n");

                    return -1;

                }

            }

            fprintf(p,"%s %s %s %d\n",current->username,current->book1,current->book2,current->userIssued);

            struct node\*temp=current;

            current=current->loc;

        }

        fclose(p);

        return 0;

    }

}

int updateIssued(char bookname[],int issORret){

    struct node\*bookStart=NULL;

    struct node\*bookEnd=NULL;

    FILE\*p=fopen("books.txt","r");

    if(p==NULL){

        printf("\n\t\t\tError\n\n");

        return 0;

    }

    char bookData[200];

    while(fgets(bookData,sizeof(bookData),p)!=NULL){

        char exname[100];

        int exqty,exissued;

        if(sscanf(bookData,"%s %d %d",exname,&exqty,&exissued)==3){

            struct node\*temp=(struct node\*)malloc(sizeof(struct node));

            strcpy(temp->bookname,exname);

            temp->qty=exqty;

            temp->issued=exissued;

            temp->loc=NULL;

            if(bookEnd==NULL){

                bookStart=temp;

            }

            else{

                bookEnd->loc=temp;

            }

            bookEnd=temp;

        }

    }

    fclose(p);

    struct node\*current=bookStart;

    if(issORret==1){

        p=fopen("books.txt","w");

        while(current!=NULL){

            if(strcmp(current->bookname,bookname)==0) {

                current->issued+=1;

            }

            fprintf(p,"%s %d %d\n",current->bookname,current->qty,current->issued);

            struct node\*temp=current;

            current=current->loc;

        }

        fclose(p);

        printf("\t\t\tBook updated\n");

        return 0;

    }

    else{

        p=fopen("books.txt","w");

        while(current!=NULL){

            if(strcmp(current->bookname,bookname)==0) {

                current->issued-=1;

            }

            fprintf(p,"%s %d %d\n",current->bookname,current->qty,current->issued);

            struct node\*temp=current;

            current=current->loc;

        }

        fclose(p);

        printf("\t\t\tBook updated.\n");

        return 0;

    }

}

int issueBook(){

    char username[50],bookname[100];

    int userPresent=0;

    int bookPresent=0;

    struct usernode\*userStart=NULL;

    struct usernode\*userEnd=NULL;

    FILE\*p=fopen("usersdata.txt","r");

    if(p==NULL){

        printf("\n\t\t\tError\n\n");

        return 0;

    }

    char userData[300];

    while(fgets(userData,sizeof(userData),p)!=NULL){

        char exusername[50],exbook1[100],exbook2[100];

        int exuserIssued;

        if(sscanf(userData,"%s %s %s %d",exusername,exbook1,exbook2,&exuserIssued)==4){

            struct usernode\*temp=(struct usernode\*)malloc(sizeof(struct usernode));

            strcpy(temp->username,exusername);

            strcpy(temp->book1,exbook1);

            strcpy(temp->book2,exbook2);

            temp->userIssued=exuserIssued;

            temp->loc=NULL;

            if(userEnd==NULL){

                userStart=temp;

            }

            else{

                userEnd->loc=temp;

            }

            userEnd=temp;

        }

    }

    fclose(p);

    struct node\*bookStart=NULL;

    struct node\*bookEnd=NULL;

    FILE\*f=fopen("books.txt","r");

    if(f==NULL){

        printf("\n\t\t\tError\n\n");

        return 0;

    }

    char bookData[200];

    while(fgets(bookData,sizeof(bookData),f)!=NULL){

        char exname[100];

        int exqty,exissued;

        if(sscanf(bookData,"%s %d %d",exname,&exqty,&exissued)==3){

            struct node\*temp=(struct node\*)malloc(sizeof(struct node));

            strcpy(temp->bookname,exname);

            temp->qty=exqty;

            temp->issued=exissued;

            temp->loc=NULL;

            if(bookEnd==NULL){

                bookStart=temp;

            }

            else{

                bookEnd->loc=temp;

            }

            bookEnd=temp;

        }

    }

    fclose(f);

    printf("\t\t\tEnter name of user: ");

    scanf("%s",username);

    struct usernode\*ucurrent=userStart;

    while(ucurrent!=NULL) {

        if(strcmp(ucurrent->username,username)==0) {

            userPresent=1;

            if(ucurrent->userIssued==1){

                printf("\n\t\t\tThis user has 1 books issued currently return it\n");

                return 0;

            }

            break;

        }

        ucurrent=ucurrent->loc;

    }

    if (userPresent==0){

            printf("\n\t\t\tUser not found\n");

            return 0;

    }

    else{

        printf("\t\t\tEnter name of Book to be issued: ");

        scanf("%s",bookname);

        struct node\*bcurrent=bookStart;

        while(bcurrent!=NULL) {

            if(strcmp(bcurrent->bookname,bookname)==0) {

                bookPresent=1;

                if(bcurrent->issued==bcurrent->qty) {

                    printf("\n\t\t\tBook is currently unavailable\n");

                    return 0;

                }

                break;

            }

            bcurrent=bcurrent->loc;

        }

    }

    if (bookPresent==0){

        printf("\n\t\t\tThe required book couldn't be found.\n");

        return 0;

    }

    else{

        updateuserData(username,bookname,1);

        updateIssued(bookname,1);

        printf("\t\t\tBook has been issued successfully.");

    }

    return 0;

}

int returnBook(){

    char username[50],bookname[100];

    int userPresent=0;

    int bookPresent=0;

    struct usernode\*userStart=NULL;

    struct usernode\*userEnd=NULL;

    FILE\*p=fopen("usersdata.txt","r");

    if(p==NULL){

        printf("\n\t\t\tError\n\n");

        return 0;

    }

    char userData[300];

    while(fgets(userData,sizeof(userData),p)!=NULL){

        char exusername[50],exbook1[100],exbook2[100];

        int exuserIssued;

        if(sscanf(userData,"%s %s %s %d",exusername,exbook1,exbook2,&exuserIssued)==4){

            struct usernode\*temp=(struct usernode\*)malloc(sizeof(struct usernode));

            strcpy(temp->username,exusername);

            strcpy(temp->book1,exbook1);

            strcpy(temp->book2,exbook2);

            temp->userIssued=exuserIssued;

            temp->loc=NULL;

            if(userEnd==NULL){

                userStart=temp;

            }

            else{

                userEnd->loc=temp;

            }

            userEnd=temp;

        }

    }

    fclose(p);

    struct node\*bookStart=NULL;

    struct node\*bookEnd=NULL;

    FILE\*f=fopen("books.txt","r");

    if(f==NULL){

        printf("\n\t\t\tError\n\n");

        return 0;

    }

    char bookData[200];

    while(fgets(bookData,sizeof(bookData),f)!=NULL){

        char exname[100];

        int exqty,exissued;

        if(sscanf(bookData,"%s %d %d",exname,&exqty,&exissued)==3){

            struct node\*temp=(struct node\*)malloc(sizeof(struct node));

            strcpy(temp->bookname,exname);

            temp->qty=exqty;

            temp->issued=exissued;

            temp->loc=NULL;

            if(bookEnd==NULL){

                bookStart=temp;

            }

            else{

                bookEnd->loc=temp;

            }

            bookEnd=temp;

        }

    }

    fclose(f);

    printf("\t\t\tEnter name of user: ");

    scanf("%s",username);

    struct usernode\*ucurrent=userStart;

    while(ucurrent!=NULL) {

        if(strcmp(ucurrent->username,username)==0) {

            userPresent=1;

            if(ucurrent->userIssued==0){

                printf("\n\t\t\tThis user has 0 books issued currently. Thus, they have got no book to return.\n");

                return 0;

            }

            break;

        }

        ucurrent=ucurrent->loc;

    }

    if (userPresent==0){

            printf("\n\t\t\tUser not found.\n");

            return 0;

    }

    else{

        printf("\t\t\tEnter name of Book to be returned: ");

        scanf("%s",bookname);

        struct node\*bcurrent=bookStart;

        while(bcurrent!=NULL) {

            if(strcmp(bcurrent->bookname,bookname)==0) {

                bookPresent=1;

                if(bcurrent->issued==0) {

                    printf("\n\t\t\tBook is currently not issued by anyone. Return can't be processed.\n");

                    return 0;

                }

                break;

            }

            bcurrent=bcurrent->loc;

        }

    }

    if (bookPresent==0){

        printf("\n\t\t\tThere's no such book\n");

        return 0;

    }

    else{

        int check = updateuserData(username,bookname,-1);

        if(check!=-1){

            updateIssued(bookname,-1);

            printf("\t\t\tBook has been returned successfully.");

            return 0;

        }

        else{

            printf("\t\t\tReturn unsuccessful.");

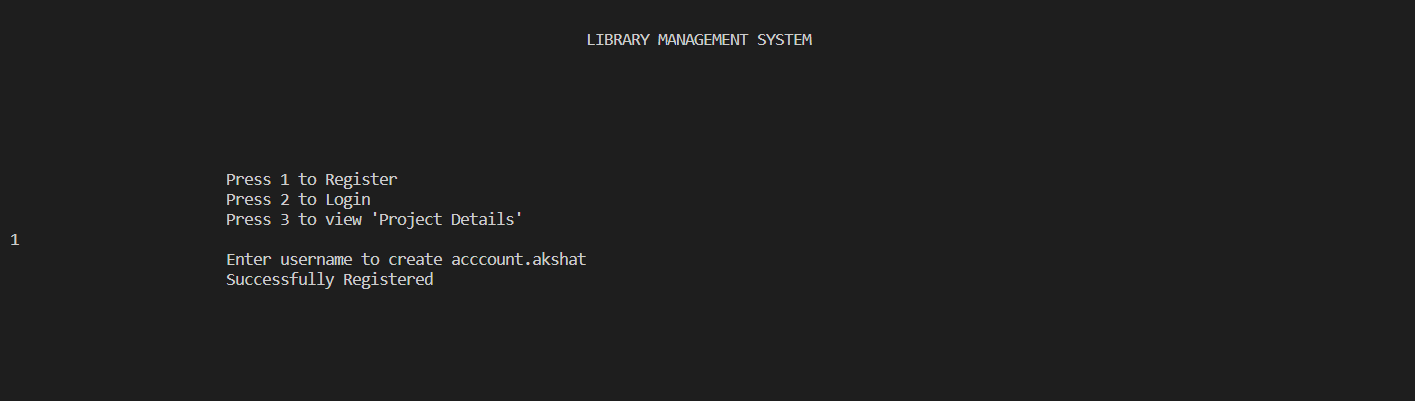
            return 0;

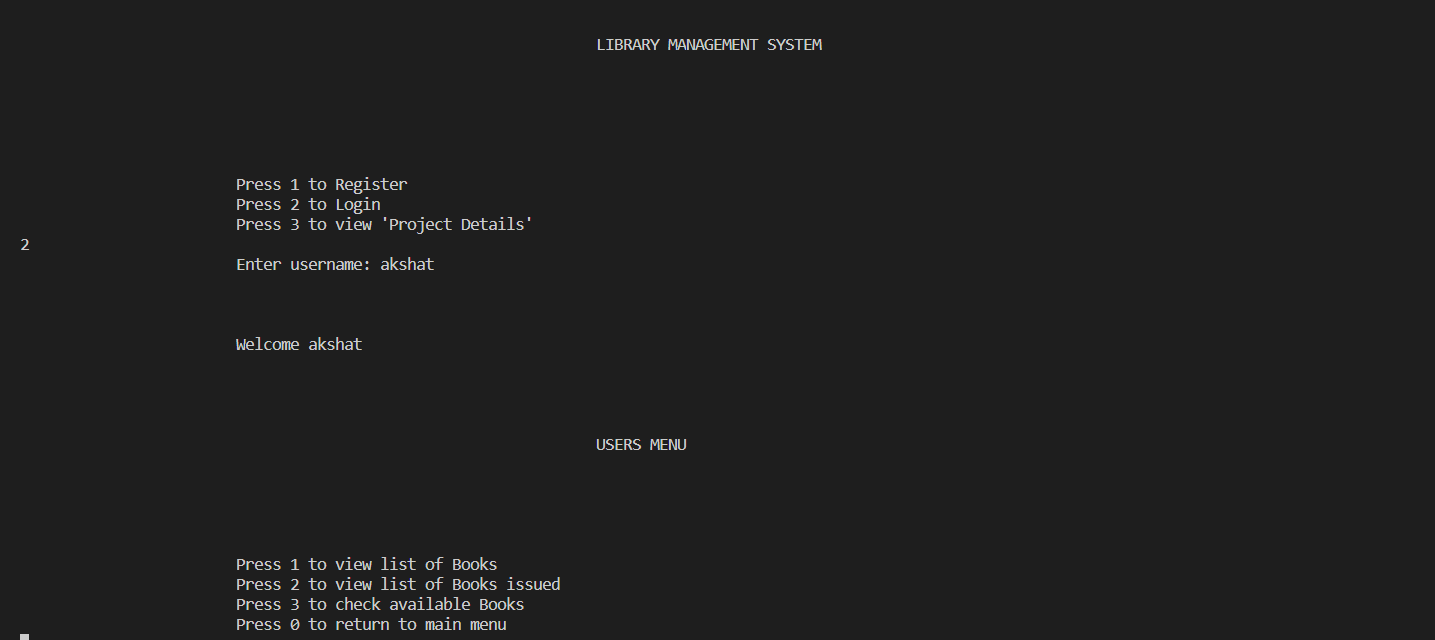
        }

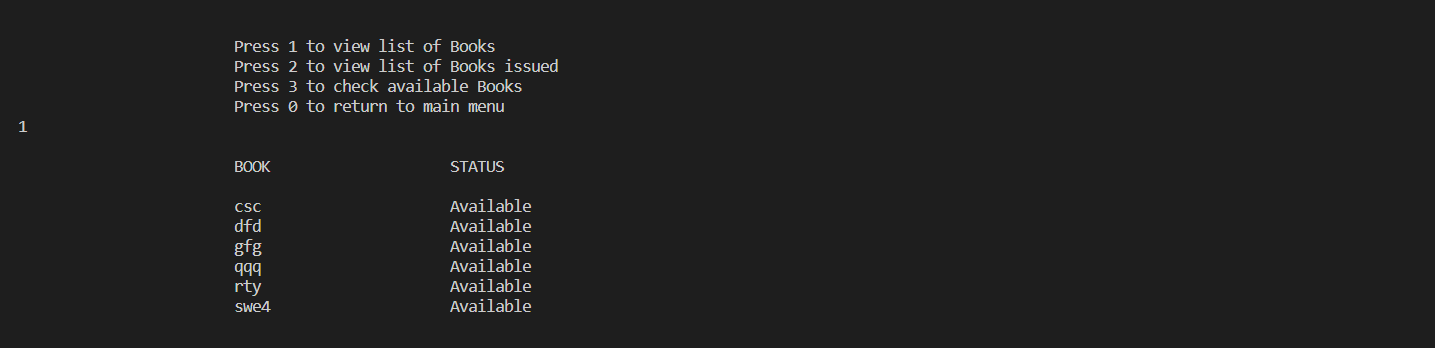
    }

}

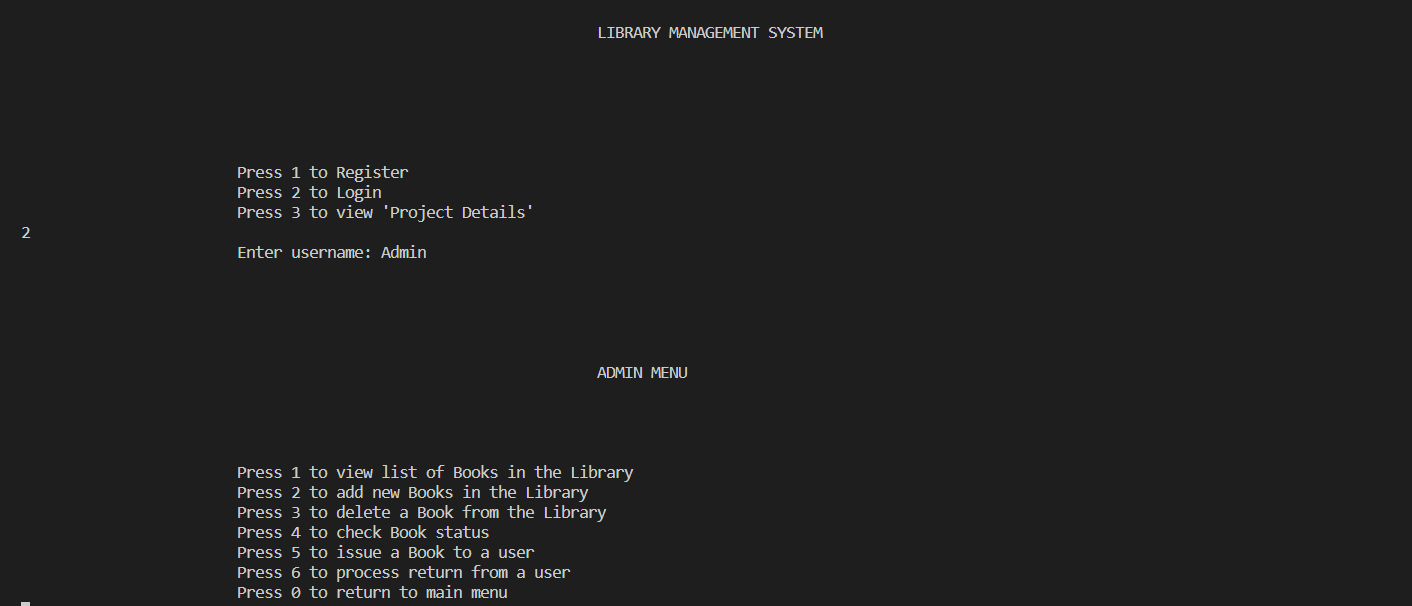
Output-

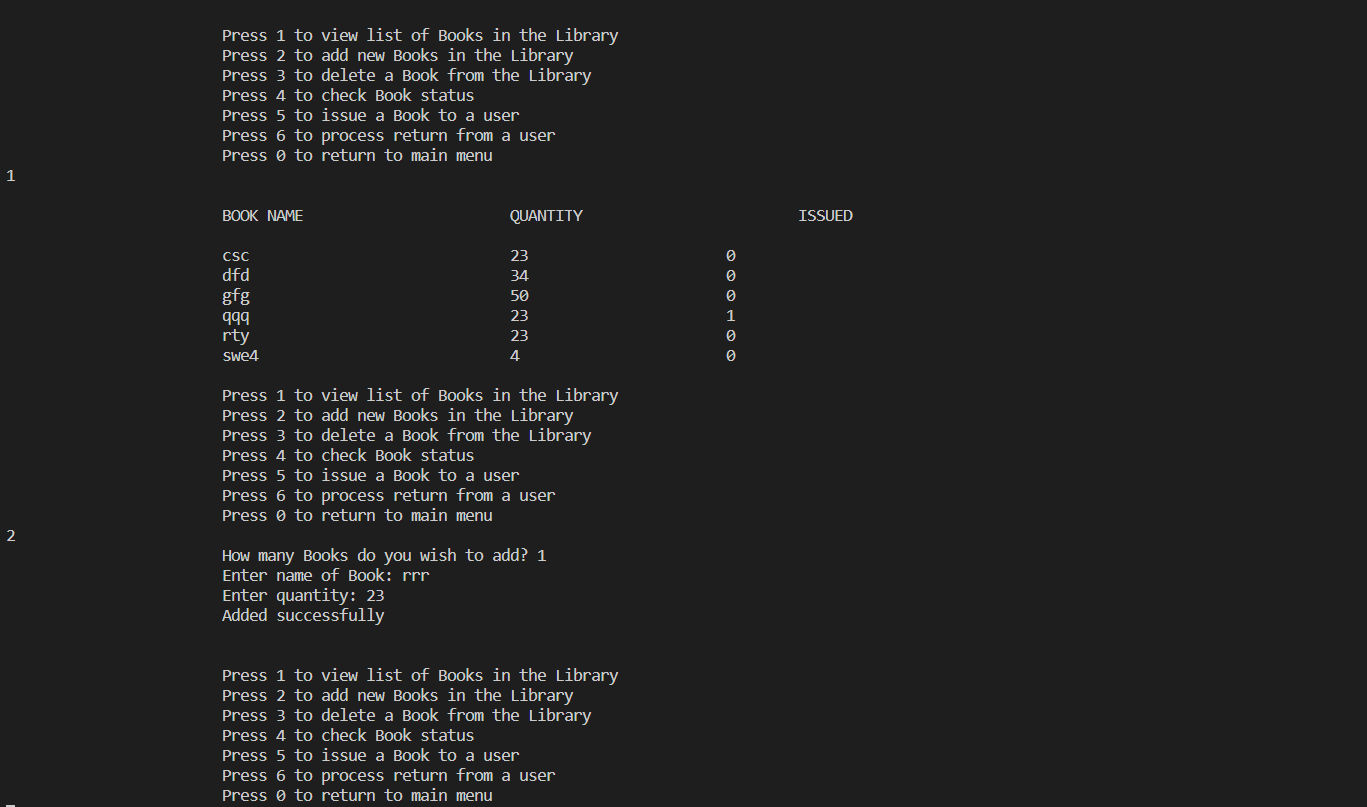


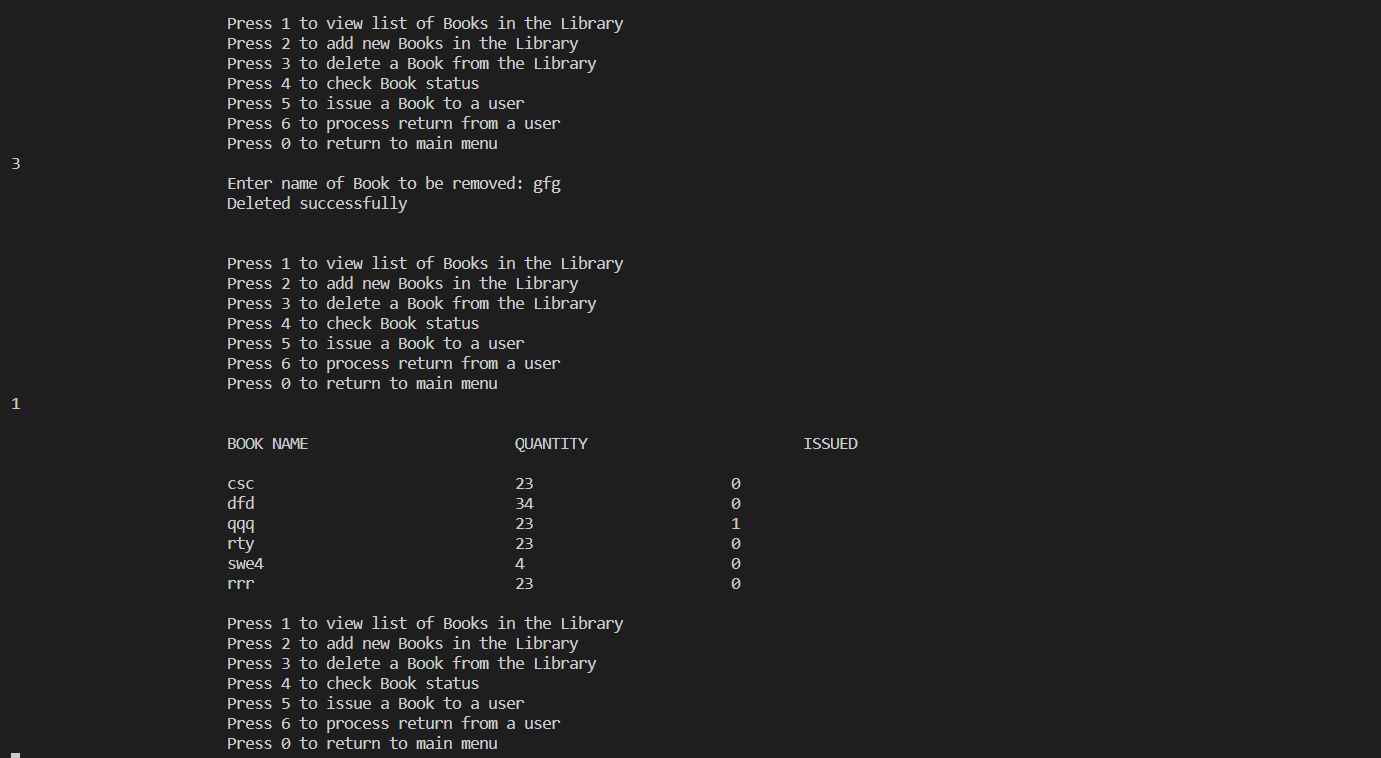


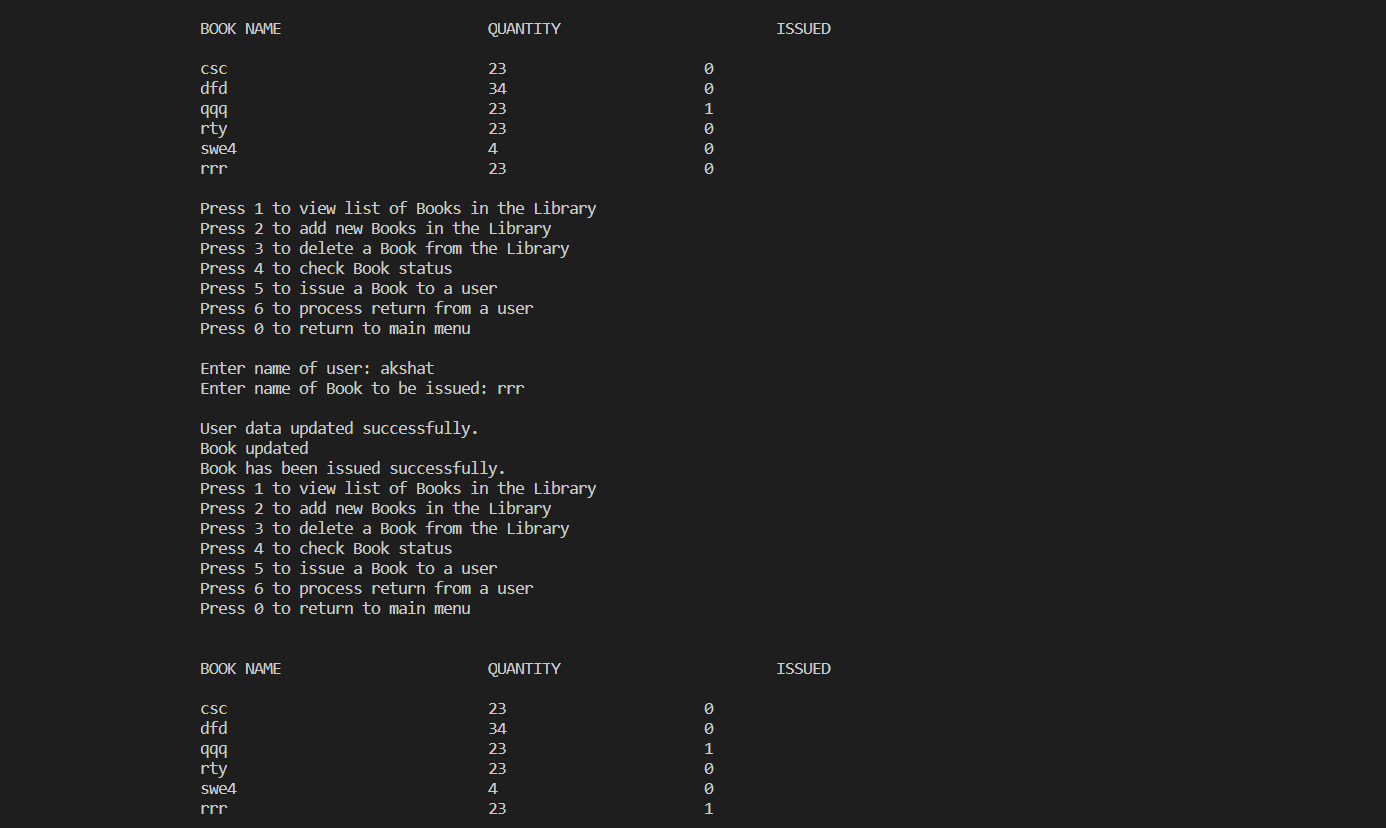












ThankYou