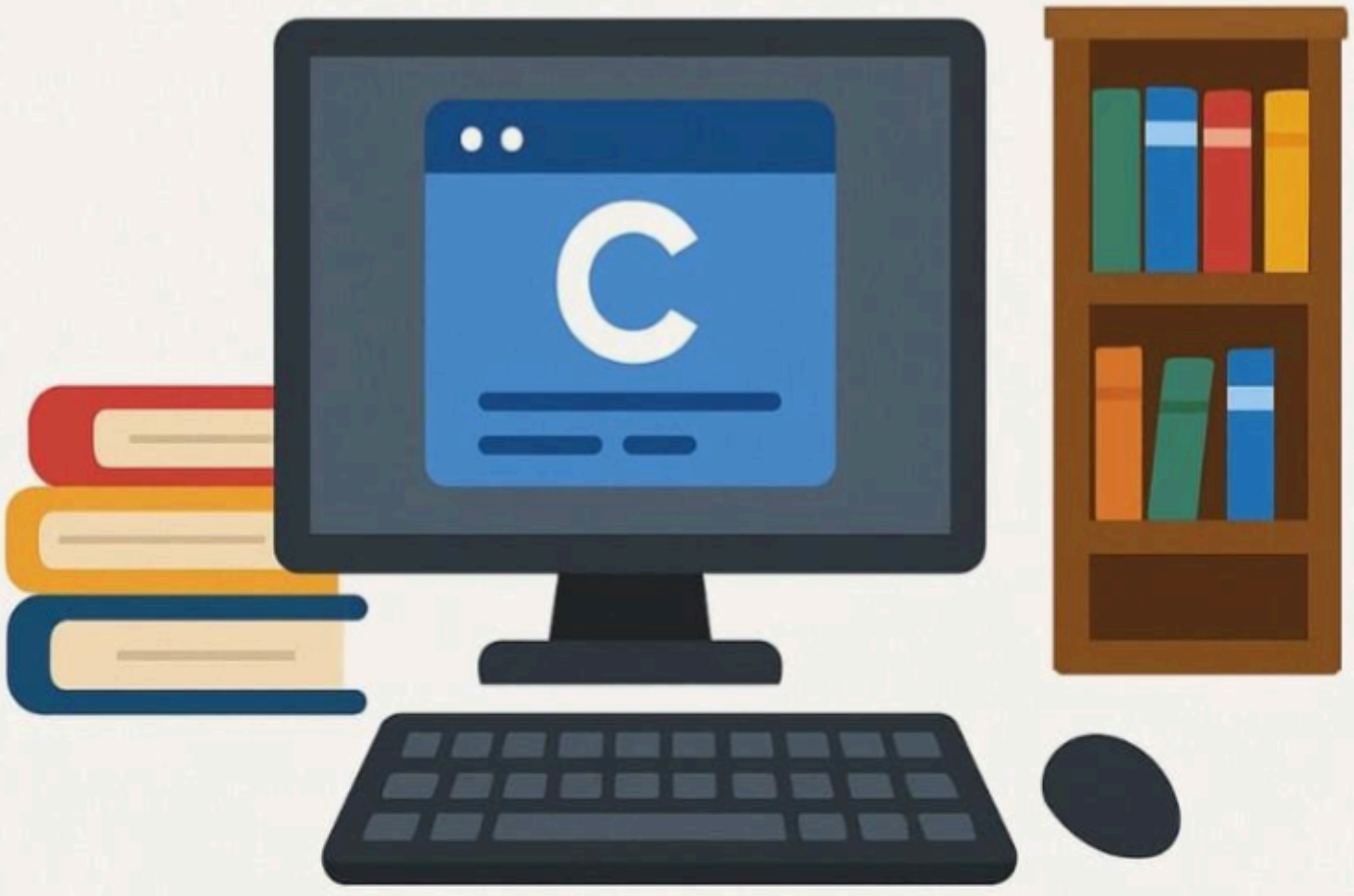
LIBRARY MANAGEMENT SYSTEM

using C in Dev-C++

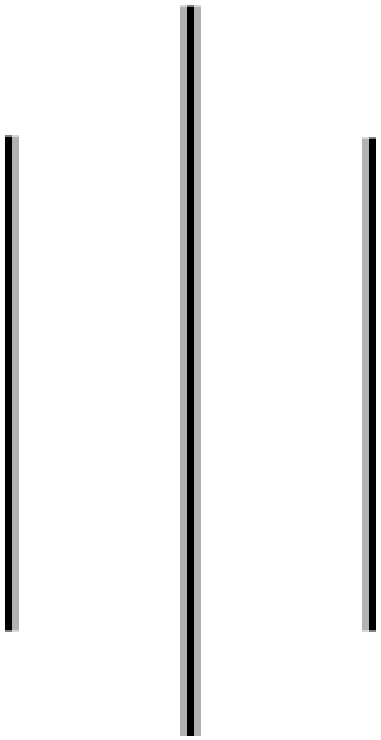


Written by:
Pasang Chand
Science Faculty

PADMODAYA SECONDARY SCHOOL

RamShahPath,Kathmandu

C Programming Project Proposal
Library Management System



Submitted To

.....

Department of Computer Science

Er.Kamal Shrestha

Date of Submission:9th Nov 2025

Submitted By

Name: Pasang Chand
Roll No. : 126
Class: 12(XII)
Subject: Computer
Faculty: Science

Acknowledgement

I would like to express my sincere gratitude to my Computer Science teacher **Er. Kamal Shrestha** for his continuous guidance, valuable suggestions, and support throughout the completion of this project. His motivation and mentorship have been crucial in understanding the concepts of programming and implementing them in practice.

I also thank **Padmodaya Secondary School** for providing the platform and resources needed for this work. My special thanks to my friends and family for their constant encouragement and help during the preparation of this project.

-Pasang Chand
Class: 12 (Science)
Roll No: 126

PADMODAYA SECONDARY SCHOOL

RamShahPath,Kathmandu

Letter Of Certification

This is to certify that Pasang Chand, a student of Class 12, Science Faculty, Roll No. 126, of Padmodaya Secondary School, Ramshah Path, Kathmandu, has successfully completed the Computer Science project entitled:

“Library Management System Using C Programming”

under my supervision, as a part of the partial fulfillment of the requirement for the Grade 12 Computer Science Practical Examination.

This project is based on the student’s own study and demonstrates his understanding of concepts such as data structures, file handling, and modular programming in C language.

Teacher’s Name: Er. Kamal Shrestha
Subject Teacher (Computer Science)
Signature: _____
Date: _____
School Stamp: _____

Abstract

This project aims to familiarize students with the practical use of the C programming language by developing a Library Management System (LMS). The system automates major library operations like adding, searching, editing, issuing, and deleting books.

It applies file handling, structures, and modular programming, and is implemented using Dev C++ IDE with the MinGW compiler. The LMS will serve as a simple, console-based system to manage book records efficiently and reduce manual workload.

Contents

1. Introduction

1.1 Introduction of C language

C is a general-purpose programming language that is extremely popular, simple, and flexible to use. It is a structured programming language that is machine-independent and extensively used to write various applications, operating systems like Windows, and many other complex programs like Oracle database, Python interpreter, and more.

It is said that “C” is the god’s programming language. One can say C is the base for programming. If you know C, you can easily grasp the knowledge of other programming languages that use the concept of C.

It is essential to have a background in computer memory mechanisms because it is an important aspect when dealing with the C programming language.

In 1972, a great computer scientist Dennis Ritchie created a new programming language called ‘C’ at Bell Laboratories. The C language combines the features of many earlier programming languages and introduces additional concepts that make it unique.

If someone wants to start learning programming, they should begin with the C language first because it forms the foundation of modern programming.

1.1.1 Types of Programming Language

Programming languages are divided into two main categories:

✓ Low-Level Language
✓ High-Level Language

- Low-Level Language:

Low-level language is difficult and time-consuming to develop programs. It is machine dependent, which means the code written for one type of computer might not work on another. Programmers need detailed knowledge of computer hardware architecture to use this language.

Low-level language can be further categorized as:

✓ Machine Language (1GL)

✓ Assembly Language (2GL)

- High-Level Language:

Instructions in high-level languages closely resemble human language or English-like words and mathematical notations. It is easier to learn, write, and maintain programs in high-level languages.

The high-level language is converted into machine-level code by a compiler or an interpreter. Examples of high-level languages include BASIC, FORTRAN, ALGOL, COBOL, and C.

High-level languages can be further categorized as:

✓ Procedural Oriented Language (3GL)

✓ Problem Oriented Language (4GL)

✓ Natural Language (5GL)

List of some high-level programming languages:

- C
- C++
- Java
- PHP
- JavaScript
- Python
- C#

1.1.2 Features of C Programming Language

- **Simple and Efficient:**
  The basic syntax style of implementing C language is very simple and easy to learn.
- **Fast:**
  C is a statically typed language, so it is faster than many dynamic languages.

- **Portability:**
  C provides source code portability, meaning a C program can be compiled and run on different machines with minimal or no modification.
- **Extensibility:**
  C is a collection of functions supported by a standard library, and users can add their own functions to extend its capabilities.
- **Mid-Level Language:**
  C supports features of both low-level and high-level languages, which is why it is called a middle-level language.

1.1.3 Structure of C Program

A C program is divided into six main sections:

1. Documentation Section
2. Link Section
3. Definition Section
4. Global Declaration Section
5. Main () Function Section
6. Subprogram Section

A program is a sequence of statements. Each statement is terminated by a semicolon (;). One or more statements can form a block enclosed within a pair of braces `{}`.

➢ Advantages of C Programming Language

- The code written in C language is simple to write and understand.
- The code executes fast and efficiently.
- It is easy to learn and use.
- C is useful for developing system software and operating systems.
- Supports user-defined functions for modular and reusable code.

➢ Disadvantages of C Programming Language

- C is a procedural-oriented language and does not support OOP concepts.
- It lacks features like inheritance and encapsulation.
- It does not support garbage collection.
- It has no constructors or destructors.
- C is case-sensitive and can be complex for large projects.

1.1.4 Uses of C Programming Language

- System Programming: Used for operating systems, device drivers, and firmware.
- Embedded Systems: Used in microcontrollers and hardware-level programming.
- Game Development: Early games and performance-critical parts of modern games are written in C.
- Compilers and Interpreters: Many are written in C for speed and control over hardware.

1.2. Function in C

In C programming, a **function** is a block of code that performs a specific task. Functions divide a large program into smaller parts, making it easier to manage and understand.

Without functions, repetitive code would make programs large, difficult to debug, and error-prone. Using functions increases modularity and efficiency.

1.2.1Types of Function:

- **Library Functions**
- **User-defined Functions**

- Library Function

Library functions are already defined in the C standard library. Examples include `printf()`, `scanf()`, `strcat()`, etc. These are ready-to-use and only require including the proper header files.

Some important types of library functions:

- Character handling functions
- Standard I/O functions
- String handling functions
- Mathematical functions
- Time manipulation functions
- Process handling functions

- User-defined Function

User-defined functions are functions created by the programmer. They improve code reusability, readability, and maintainability.

There are four main types of user-defined functions:

1. Function with no arguments and no return value
2. Function with no arguments but a return value
3. Function with arguments and no return value
4. Function with arguments and a return value

1.2.2 Benefits of Using Functions

▪ Provides modular structure.
▪ Promotes code reusability.
▪ Makes debugging and editing easier.
▪ Increases readability and understanding.

1.2.3 Function Declaration

A function declaration defines the function name, return type, and parameters. It tells the compiler what to expect from the function.
Example:

```
int add(int, int);
```

1.2.4 Function Call

A function call is used to execute a function. The number and type of parameters must match the function declaration.

Example:

```
add(5, 10);
```

1.2.5 Function Definition

The function definition contains the actual set of instructions that are executed when the function is called.

Example:

```
int add(int a, int b) {

   return a + b;

}
```

1.2.6. File Handling

File handling in C refers to the process of reading from and writing to files. It allows data to be stored permanently.
Common operations include:

- Creating and opening files (`fopen()`)
- Writing data (`fprintf()` or `fwrite()`)
- Reading data (`fscanf()` or `fread()`)
- Closing files (`fclose()`)

Example: Writing Data to a File

```
#include <stdio.h>
int main() {
   int val;
   FILE *fptr;
   fptr = fopen("open.txt", "w");
```

```
    if (fptr == NULL) {
        printf("File error!");
        return 1;
    }
    printf("Enter a number: ");
    scanf("%d", &val);
    fprintf(fptr, "%d", val);
    fclose(fptr);
    return 0;
}
```

---

2. Objective

The main objectives of this project are:

- To add, search, and delete book records.
- To issue and return books to students.
- To edit and view existing records.
- To store and retrieve data using file handling.
- To automate library management using C programming.

---

3. Methodology

The development of this project follows a structured and modular approach using the C programming language. The project is divided into different stages to ensure smooth progress and successful completion.

3.1 Tools and Techniques Used

- Programming Language: C
- IDE**:** Dev C++ (with MinGW compiler)
- Concepts Used:
    - Structures
    - Functions
    - File handling
    - Conditional statements
    - Loops (`for`, `while`, `do-while`)
    - `switch-case` for menu control

3.2 Stages of Development

1. **Requirement Gathering:**
   In this stage, the main requirements of the system were collected. The essential functionalities identified included adding, searching, deleting, issuing, and returning books. This phase helped to clearly define the scope and objectives of the project.
2. **System Design:**
   During system design, structures were created to store book and student data efficiently. The logical flow and overall architecture of the program were planned using pseudocode and flowcharts. This provided a blueprint for the implementation phase and ensured that the program would be organized and modular.
3. **Implementation:**
   The system was coded in the C programming language using modular functions. Each feature, such as add, search, edit, delete, issue, and return, was developed as a separate function. This approach enhanced readability, maintainability, and modularity of the code.
4. **Testing and Debugging:**
   Each module was individually tested to ensure correct outputs. After module-level testing, the entire program was tested using sample inputs to detect and fix errors. This ensured the smooth operation and reliability of the system.

5. **Documentation:**

   After successful testing, the project report and user manual were prepared. These documents explain the design, logic, and usage of the system, providing a comprehensive guide for both users and future developers.

3.3 Flow Chart

The flowchart below represents the logical sequence and working procedure of the Library Management System. It shows how the program starts, accepts inputs, performs operations, and terminates after saving data.

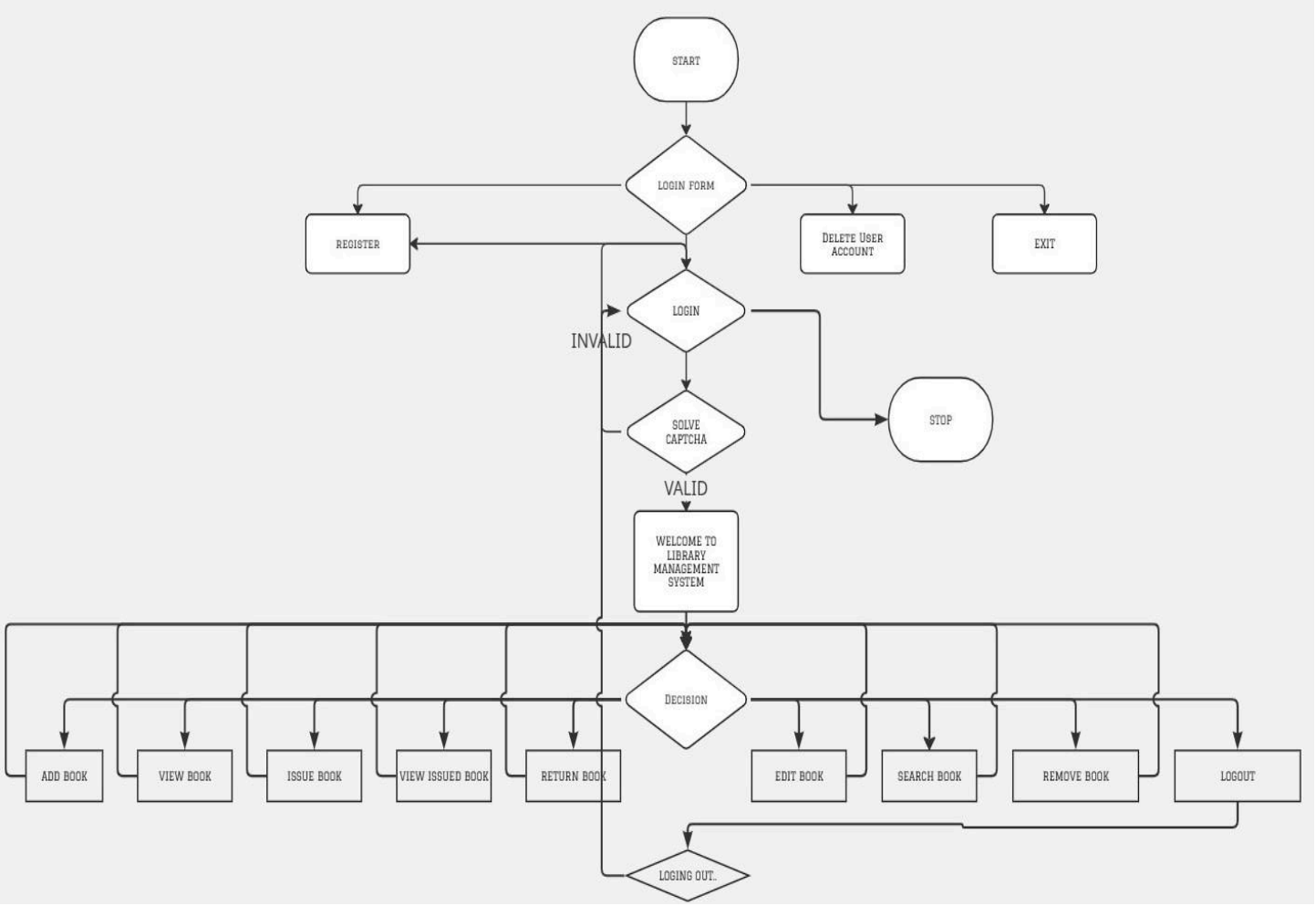


Fig: Flowchart of The Project

---

4. Coding

All the code given below:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#define MAX_LEN 50
#define EMAIL_LEN 100

struct user {
    char username[MAX_LEN];
    char email[EMAIL_LEN];
    char password[MAX_LEN];
};

struct Book {
    int id;
    char title[MAX_LEN];
    char author[MAX_LEN];
    int quantity;
};

struct IssuedBook {
    char clientName[MAX_LEN];
    char clientContact[30];
    char clientAddress[200];
    int bookID;
    char title[MAX_LEN];
    char author[MAX_LEN];
    char date[20];
    char returnDate[20];
};

void startMenu();
void registerUser();
void deleteUser();
int logIn();
void menu();
void addBook();
void viewBooks();
void issueBook();
void viewIssuedBooks();
void returnBook();
```

```
void editBook();
void searchBook();
void removeBook();
int generateCaptcha();

int main() {
    srand(time(NULL));
    startMenu();
    return 0;
}

void startMenu() {
    int choice;
    while(1) {
        printf("\n===== LOGIN SYSTEM =====\n");
        printf("1. Register\n2. Log In\n3. Delete User Account\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        getchar();
        if(choice == 1) registerUser();
        else if(choice == 2) {
            if(logIn()) {
                printf("\nLogin successful!! \n");
                menu();
            } else {
                printf("\nLogin failed! Try again.\n");
            }
        }
        else if(choice == 3) deleteUser();
        else if(choice == 4) {
            printf("Exiting program...\n");
            exit(0);
        }
        else printf("Invalid choice! Try again.\n");
    }
}

void registerUser() {
    struct user u, temp;
    int exists = 0;
    char line[300];

    printf("Enter username: ");
    fgets(u.username, MAX_LEN, stdin); u.username[strcspn(u.username,"\n")]=0;
    printf("Enter email: ");
    fgets(u.email, EMAIL_LEN, stdin); u.email[strcspn(u.email,"\n")]=0;
    printf("Enter password: ");
    fgets(u.password, MAX_LEN, stdin); u.password[strcspn(u.password,"\n")]=0;
```

```
    FILE *f = fopen("users.txt","r");
    if(f) {
        while(fgets(line,sizeof(line),f)) {
            line[strcspn(line,"\n")]=0;
            sscanf(line,"%[^\t]\t%[^\t]\t%[^\n]",temp.username,temp.email,temp.password);
            if(strcmp(u.username,temp.username)==0 || strcmp(u.email,temp.email)==0) {
                printf("Username or Email already exists!\n");
                exists=1; break;
            }
        }
        fclose(f);
    }

    if(exists) return;

    f=fopen("users.txt","a");
    if(!f){printf("Error opening file\n"); return;}
    fprintf(f,"%s\t%s\t%s\n",u.username,u.email,u.password);
    fclose(f);
    printf("User registered successfully!\n");
}

void deleteUser() {
    char username[MAX_LEN], line[300], u[MAX_LEN], e[EMAIL_LEN], p[MAX_LEN];
    int found=0;
    printf("Enter username to delete: ");
    fgets(username,MAX_LEN,stdin); username[strcspn(username,"\n")]=0;

    FILE *f=fopen("users.txt","r");
    FILE *temp=fopen("temp.txt","w");
    if(!f || !temp){ printf("Error opening file\n"); return;}
    while(fgets(line,sizeof(line),f)) {
        line[strcspn(line,"\n")]=0;
        sscanf(line,"%[^\t]\t%[^\t]\t%[^\n]",u,e,p);
        if(strcmp(u,username)!=0) fprintf(temp,"%s\t%s\t%s\n",u,e,p);
        else found=1;
    }
    fclose(f); fclose(temp);
    remove("users.txt"); rename("temp.txt","users.txt");
    if(found) printf("User deleted successfully.\n");
    else printf("User not found.\n");
}

int generateCaptcha() {
    int a=rand()%20+1;
    int b=rand()%20+1;
    int op=rand()%3, res=0;
```

```
    if(op==0){ printf("CAPTCHA: %d + %d =  ",a,b); res=a+b;}
    else if(op==1){ printf("CAPTCHA: %d - %d =  ",a,b); res=a-b;}
    else { printf("CAPTCHA: %d * %d =  ",a,b); res=a*b;}
    return res;
}

int logIn() {
    char id[MAX_LEN], pass[MAX_LEN], line[300];
    struct user u;
    printf("Enter username or email: ");
    fgets(id,MAX_LEN,stdin); id[strcspn(id,"\n")]=0;
    printf("Enter password: ");
    fgets(pass,MAX_LEN,stdin); pass[strcspn(pass,"\n")]=0;

    int captcha=generateCaptcha();
    int ans; scanf("%d",&ans); getchar();
    if(ans!=captcha){ printf("CAPTCHA failed!\n"); return 0;}

    FILE *f=fopen("users.txt","r");
    if(!f){ printf("No users registered yet.\n"); return 0;}
    while(fgets(line,sizeof(line),f)) {
        line[strcspn(line,"\n")]=0;
        sscanf(line,"%[^\t]\t%[^\t]\t%[^\n]",u.username,u.email,u.password);
        if((strcmp(id,u.username)==0 || strcmp(id,u.email)==0) &&
strcmp(pass,u.password)==0) {
            fclose(f); return 1;
        }
    }
    fclose(f);
    return 0;
}

void menu() {printf("\n\n========== welcome To library Management System
==========\n");
    int choice;
    while(1) {
        printf("\n===== Library Menu =====\n1.Add Book\n2.View Books\n3.Issue
Book\n4.View Issued Books\n5.Return Book\n6.Edit Book\n7.Search Book\n8.Remove
Book\n9.Logout\n");
        printf("Enter choice: "); scanf("%d",&choice); getchar();
        if(choice==1) addBook();
        else if(choice==2) viewBooks();
        else if(choice==3) issueBook();
        else if(choice==4) viewIssuedBooks();
        else if(choice==5) returnBook();
        else if(choice==6) editBook();
        else if(choice==7) searchBook();
        else if(choice==8) removeBook();
```

```
        else if(choice==9){ printf("Logging out...\n"); return;}
        else printf("Invalid choice\n");
    }
}

void addBook() {
    struct Book b,temp;
    int exists=0;
    char line[200];
    printf("Enter Book ID: ");
          scanf("%d",&b.id); getchar();
    FILE *f=fopen("library.txt","r");
    if(f){
        while(fgets(line,sizeof(line),f)){
            sscanf(line,"%d\t%[^\t]\t%[^\t]\t%d",&temp.id,temp.title,temp.author,&temp.quantity);
            if(temp.id==b.id){ exists=1; break;}
        }
        fclose(f);
    }
    if(exists){ printf("Book ID exists\n"); return;}

    printf("Enter Title: "); fgets(b.title,MAX_LEN,stdin); b.title[strcspn(b.title,"\n")]=0;
    printf("Enter Author: "); fgets(b.author,MAX_LEN,stdin); b.author[strcspn(b.author,"\n")]=0;
    printf("Enter Quantity: "); scanf("%d",&b.quantity);

    f=fopen("library.txt","a");
    if(!f){ printf("Error opening file\n"); return;}
    fprintf(f,"%d\t%s\t%s\t%d\n",b.id,b.title,b.author,b.quantity);
    fclose(f);
    printf("Book added successfully\n");
}

void viewBooks() {
    struct Book b;
    char line[200];
    FILE *f=fopen("library.txt","r");
    printf("\n%-10s %-20s %-30s %-10s\n", "Book ID", "Title", "Author", "Qty");

printf("================================================================
=\n");
    if(!f){ printf("No books found\n"); return;}
    while(fgets(line,sizeof(line),f)){
        sscanf(line,"%d\t%[^\t]\t%[^\t]\t%d",&b.id,b.title,b.author,&b.quantity);
        printf("%-10d %-20s %-30s %-10d\n",b.id,b.title,b.author,b.quantity);
    }
    fclose(f);
}
```

```
void issueBook() {
    struct Book b;
    struct IssuedBook ib;
    int found=0;
    char line[200], tempFile[]="temp.txt";
    FILE *f=fopen("library.txt","r");
    FILE *temp=fopen(tempFile,"w");
    if(!f || !temp){ printf("Error\n"); return;}

    printf("Enter Book ID to issue: "); scanf("%d",&ib.bookID); getchar();
    while(fgets(line,sizeof(line),f)){
        sscanf(line,"%d\t%[^\t]\t%[^\t]\t%d",&b.id,b.title,b.author,&b.quantity);
        if(b.id==ib.bookID){
            if(b.quantity>0){
                b.quantity--; found=1;
                strcpy(ib.title,b.title); strcpy(ib.author,b.author);
                printf("Client Name: "); fgets(ib.clientName,MAX_LEN,stdin);
ib.clientName[strcspn(ib.clientName,"\n")]=0;
                printf("Client Contact: "); fgets(ib.clientContact,30,stdin);
ib.clientContact[strcspn(ib.clientContact,"\n")]=0;
                printf("Client Address: "); fgets(ib.clientAddress,200,stdin);
ib.clientAddress[strcspn(ib.clientAddress,"\n")]=0;
                time_t t=time(NULL); strftime(ib.date,sizeof(ib.date),"%d-%m-%Y
%H:%M:%S",localtime(&t));
                FILE *ibf=fopen("issuedBooks.txt","a");
                if(ibf){
fprintf(ibf,"%s\t%s\t%s\t%d\t%s\t%s\t%s\n",ib.clientName,ib.clientContact,ib.clientAddress,ib.
bookID,ib.title,ib.author,ib.date); fclose(ibf);}
            }else printf("Book not available\n");
        }
        fprintf(temp,"%d\t%s\t%s\t%d\n",b.id,b.title,b.author,b.quantity);
    }
    fclose(f); fclose(temp);
    remove("library.txt"); rename(tempFile,"library.txt");
    if(found) printf("Book issued successfully\n");
    else printf("Book not found\n");
}

void viewIssuedBooks() {
    struct IssuedBook ib;
    char line[300];
    FILE *f=fopen("issuedBooks.txt","r");
    if(!f){ printf("No issued books\n"); return;}
    printf("\n%-20s %-15s %-25s %-10s %-20s %-20s %-12s\n", "Client Name", "Contact",
"Address", "Book ID", "Title", "Author", "Date");

printf("============================================================
```

```
==========================================================================\n"
);
   while(fgets(line,sizeof(line),f)){

sscanf(line,"%[^\t]\t%[^\t]\t%[^\t]\t%d\t%[^\t]\t%[^\t]\t%[^\n]",ib.clientName,ib.clientContact,ib.c
lientAddress,&ib.bookID,ib.title,ib.author,ib.date);
      printf("%-20s %-15s %-25s %-10d %-20s %-20s
%-12s\n",ib.clientName,ib.clientContact,ib.clientAddress,ib.bookID,ib.title,ib.author,ib.date);
   }
   fclose(f);
}

void returnBook() {
   struct IssuedBook ib;
   struct Book b;
   char line[300], tempFile[]="tempIssued.txt";
   int bookID, found=0;
   char clientName[50];
   printf("Enter Book ID: "); scanf("%d",&bookID); getchar();
   printf("Enter Client Name: "); fgets(clientName,50,stdin);
clientName[strcspn(clientName,"\n")]=0;

   FILE *f=fopen("issuedBooks.txt","r");
   FILE *temp=fopen(tempFile,"w");
   if(!f || !temp){ printf("Error\n"); return;}
   while(fgets(line,sizeof(line),f)){

sscanf(line,"%[^\t]\t%[^\t]\t%[^\t]\t%d\t%[^\t]\t%[^\t]\t%[^\n]",ib.clientName,ib.clientContact,ib.c
lientAddress,&ib.bookID,ib.title,ib.author,ib.date);
      if(ib.bookID==bookID && strcmp(ib.clientName,clientName)==0){
         found=1;
         time_t t=time(NULL);
         char rdate[20]; strftime(rdate,sizeof(rdate),"%d-%m-%Y %H:%M:%S",localtime(&t));
         printf("Return Date: %s\n",rdate);
         FILE *rf=fopen("returnedBooks.txt","a");
         if(rf){
fprintf(rf,"%s\t%s\t%s\t%d\t%s\t%s\t%s\n",ib.clientName,ib.clientContact,ib.clientAddress,ib.
bookID,ib.title,ib.author,rdate); fclose(rf);}
         FILE *lib=fopen("library.txt","r+");
         if(lib){
            struct Book tempB;
            char libLine[200]; long pos;
            while(!feof(lib)){
               pos=ftell(lib);
               if(!fgets(libLine,sizeof(libLine),lib)) break;

sscanf(libLine,"%d\t%[^\t]\t%[^\t]\t%d",&tempB.id,tempB.title,tempB.author,&tempB.quantity);
```

```
                if(tempB.id==bookID){ tempB.quantity++; fseek(lib,pos,SEEK_SET);
fprintf(lib,"%d\t%s\t%s\t%d\n",tempB.id,tempB.title,tempB.author,tempB.quantity); break;}
            }
            fclose(lib);
        }
    }else fprintf(temp,"%s",line);
  }
  fclose(f); fclose(temp);
  remove("issuedBooks.txt"); rename(tempFile,"issuedBooks.txt");
  if(found) printf("Book returned successfully\n");
  else printf("Issued book not found\n");
}

void editBook() {
  struct Book b;
  char line[200], tempFile[]="temp.txt";
  int bookID, found=0;
  printf("Enter Book ID to edit: "); scanf("%d",&bookID); getchar();
  FILE *f=fopen("library.txt","r"); FILE *temp=fopen(tempFile,"w");
  if(!f || !temp){ printf("Error\n"); return;}
  while(fgets(line,sizeof(line),f)){
    sscanf(line,"%d\t%[^\t]\t%[^\t]\t%d",&b.id,b.title,b.author,&b.quantity);
    if(b.id==bookID){
      found=1;
      printf("Enter new Title: "); fgets(b.title,MAX_LEN,stdin); b.title[strcspn(b.title,"\n")]=0;
      printf("Enter new Author: "); fgets(b.author,MAX_LEN,stdin);
b.author[strcspn(b.author,"\n")]=0;
      printf("Enter new Quantity: "); scanf("%d",&b.quantity); getchar();
    }
    fprintf(temp,"%d\t%s\t%s\t%d\n",b.id,b.title,b.author,b.quantity);
  }
  fclose(f); fclose(temp);
  remove("library.txt"); rename(tempFile,"library.txt");
  if(found) printf("Book edited successfully\n"); else printf("Book not found\n");
}

void searchBook() {
  struct Book b; char line[200], keyword[MAX_LEN]; int found=0;
  printf("Enter Title or Author to search: "); fgets(keyword,MAX_LEN,stdin);
keyword[strcspn(keyword,"\n")]=0;
  FILE *f=fopen("library.txt","r"); if(!f){printf("No books found\n"); return;}
  printf("\n%-10s %-20s %-30s %-10s\n", "Book ID", "Title", "Author", "Qty");

printf("================================================================
=\n");
  while(fgets(line,sizeof(line),f)){
    sscanf(line,"%d\t%[^\t]\t%[^\t]\t%d",&b.id,b.title,b.author,&b.quantity);
```

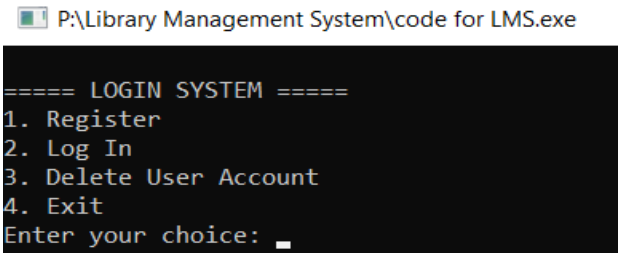
```
        if(strstr(b.title,keyword) || strstr(b.author,keyword)){ found=1; printf("%-10d %-20s
%-30s %-10d\n",b.id,b.title,b.author,b.quantity);}
    }
    fclose(f);
    if(!found) printf("No matching books found\n");
}

void removeBook() {
    struct Book b; char line[200], tempFile[]="temp.txt"; int bookID, found=0;
    printf("Enter Book ID to remove: "); scanf("%d",&bookID); getchar();
    FILE *f=fopen("library.txt","r"); FILE *temp=fopen(tempFile,"w");
    if(!f || !temp){ printf("Error\n"); return;}
    while(fgets(line,sizeof(line),f)){
        sscanf(line,"%d\t%[^\t]\t%[^\t]\t%d",&b.id,b.title,b.author,&b.quantity);
        if(b.id==bookID) found=1; else
fprintf(temp,"%d\t%s\t%s\t%d\n",b.id,b.title,b.author,b.quantity);
    }
    fclose(f); fclose(temp);
    remove("library.txt"); rename(tempFile,"library.txt");
    if(found) printf("Book removed successfully\n"); else printf("Book not found\n");
}
```

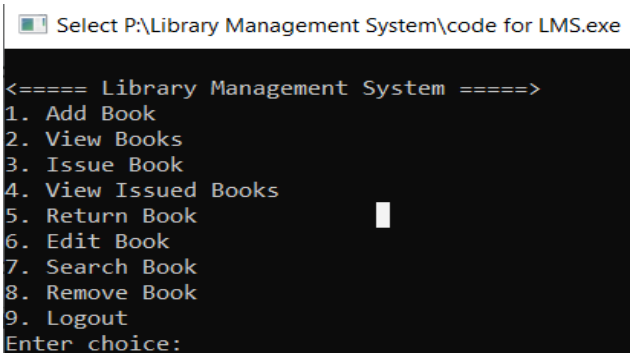
---

5. OUTPUTS

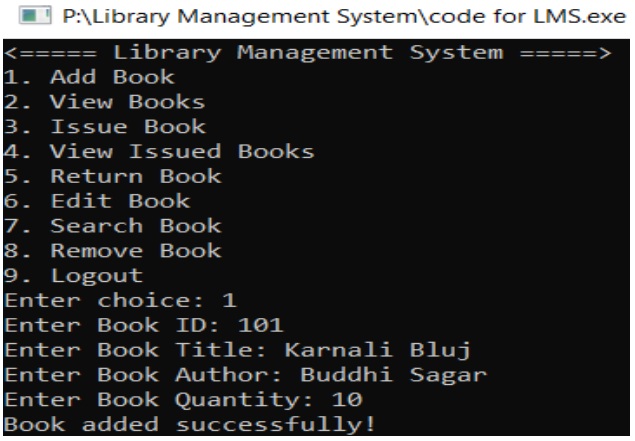
1. Login System



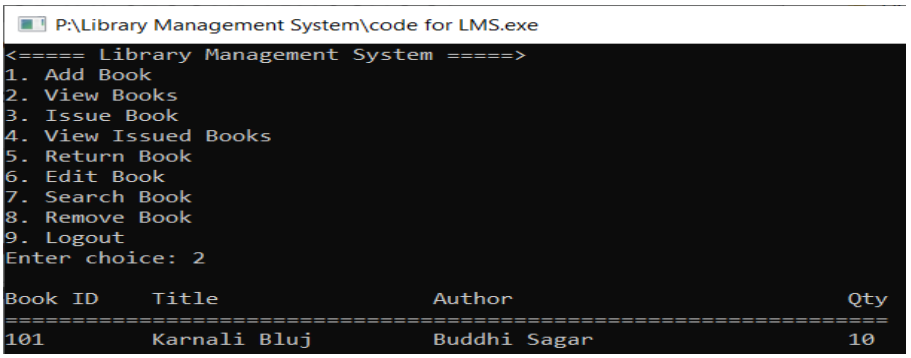
2. Library Menu



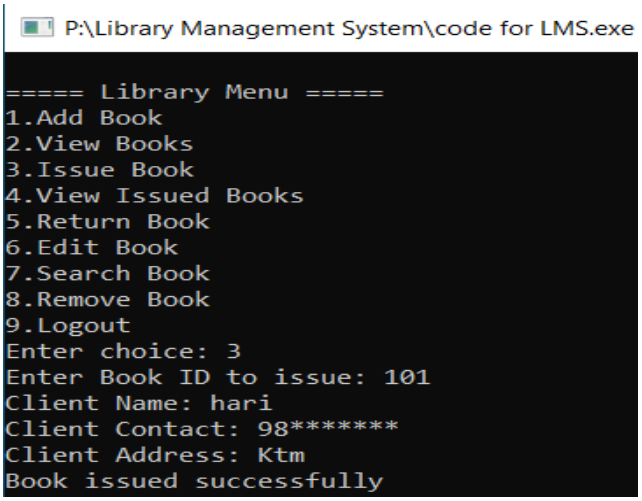
3. Add Book



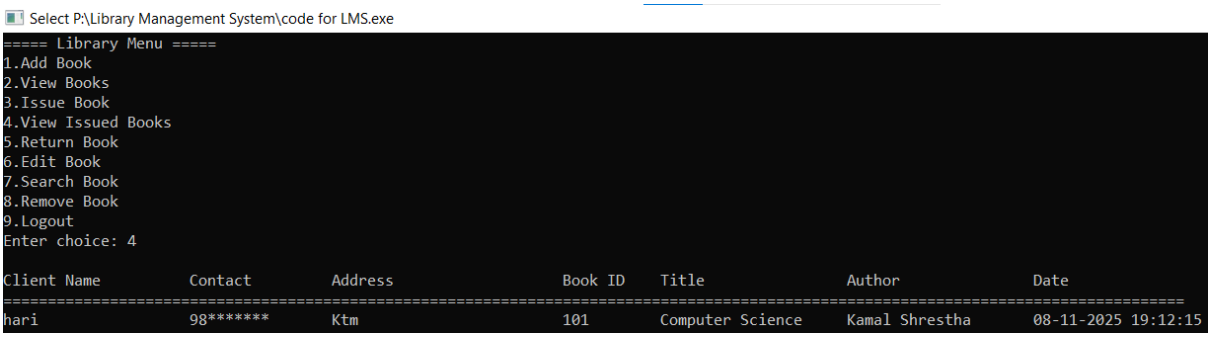
4. View Book



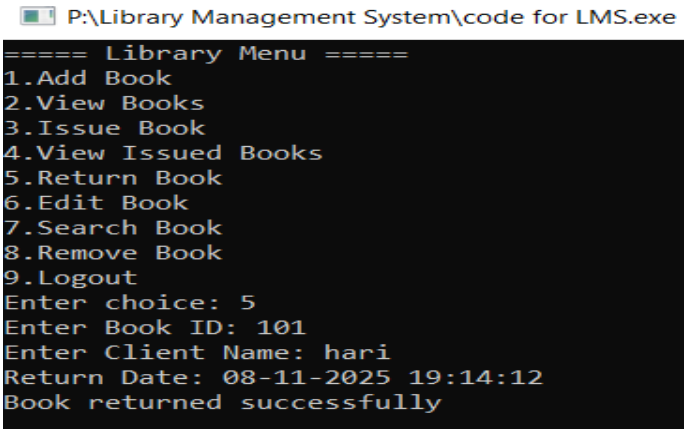
5. Issue Book



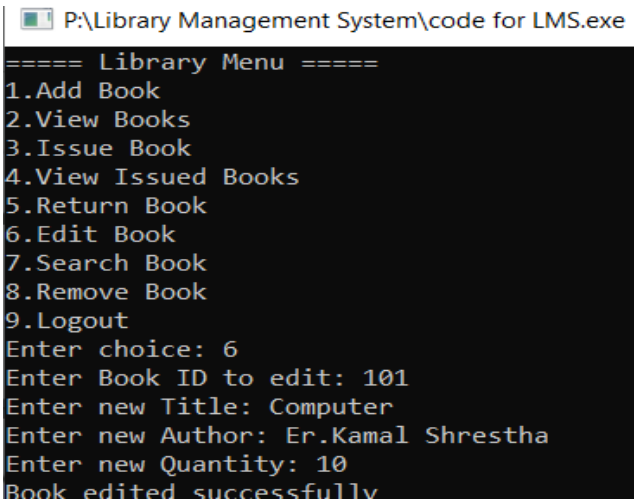
6. ViewIssued Book



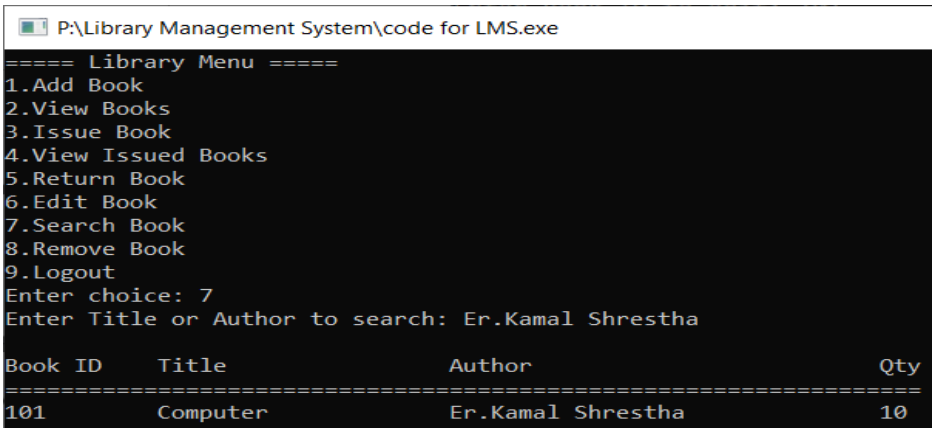
7. Return Book



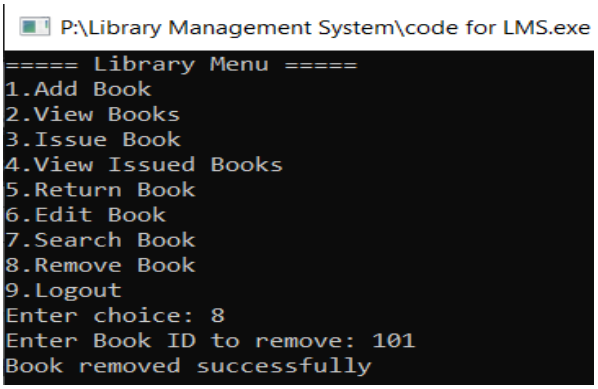
8. Edit Book



9. Search Book



10. Remove Book



6. Conclusion

The Library Management System project in C was successfully developed to perform all essential operations, including adding, searching, editing, deleting, issuing, and viewing books. File handling was used to ensure permanent data storage, while functions allowed a modular and structured program design. Compared to manual library operations, this system provides increased speed, accuracy, and efficiency. Through this project, practical experience was gained in file operations (`fopen`, `fwrite`, `fread`), using structures for complex data, modular programming, and designing logical flow with flowcharts. Future enhancements could include integrating a graphical user interface and using a database management system to handle large-scale library data.

7. Bibliography

- https://www.w3schools.com/c/c_break_continue.php
- https://www.tutorialspoint.com/ansi_c/c_working_with_files.htm#:~:text=For%20C%20File%20I%2FO,which%20returns%20a%20FILE%20pointer
- https://www.javatpoint.com/file-handling-in-c
- https://www.programiz.com/c-programming
- https://youtu.be/irqbmMNs2Bo?si=Gx02GbyEsame4Scq (ApnaCollege)
- https://youtu.be/VSEnzzjAm0c?si=YZnJQp3WjoqS0H8n(Learn Coding)
- https://youtu.be/87SH2Cn0s9A?si=asSBo0BXzIzU04IP(Bro Code)