East West University

CSE103: Structured Programming

Fall 2024

Project Report

Library Management System

Submitted to:

Md. Ashraful Haider Chowdhury
Lecturer
Department of Computer Science and Engineering

Submitted by :

Student ID	Student Name
2024-3-60-076	Muhammad Taef Ibrahim
2024-3-60-077	Mashrafi Islam Mahin
2024-3-60-080	Naheda Islam Ela
2024-3-60-082	Abdullah Al Junaid

Group I'D : 01

Section : 28

Date Of Submission : 22-01-2025

Table of Contents

Abstraction

This project is a simple Library Management System implemented in C. It allows users to manage book and borrower records efficiently. The main features of the system include:

- **Add Book:** Add new books to the library's inventory.
- **Search Book:** Search for books by name, author, or category.
- **Issue Book:** Issue books to borrowers and keep track of issue and return dates.
- **Return Book:** Return books and calculate fines for late returns.
- **Calculate Fine:** Calculate fines for overdue books.
- **Display Books:** Display a list of all books in the library.
- **Display Borrowers:** Display a list of all borrowers and their borrowed books.
- **Delete Book:** Remove books from the library's inventory.
- **Edit Book:** Edit the details of existing books.
- **Delete Borrower:** Remove borrowers from the system.
- **Edit Borrower:** Edit the details of existing borrowers.
- **Clear All Records:** Clear all book and borrower records from the system.

The system uses file I/O to save and load records, ensuring data persistence between sessions.

Introduction

Managing a library's inventory and borrower records can be a complex and time-consuming task. Traditional methods of record-keeping, such as using paper logs or basic spreadsheets, are prone to errors, inefficiencies, and data loss. This project, a Library Management System (LMS) implemented in C, addresses these challenges by providing a streamlined and automated solution for managing library operations.

The LMS allows librarians to efficiently manage book inventories, track borrower information, and handle book issuance and returns. By automating these processes, the system reduces the likelihood of errors, ensures data consistency, and saves time for library staff. Additionally, the system's ability to calculate fines for overdue books helps maintain accountability among borrowers.

Overall, this project is important because it enhances the efficiency and reliability of library management, ultimately improving the user experience for both librarians and borrowers.

Projects Aims and Objectives

Aims

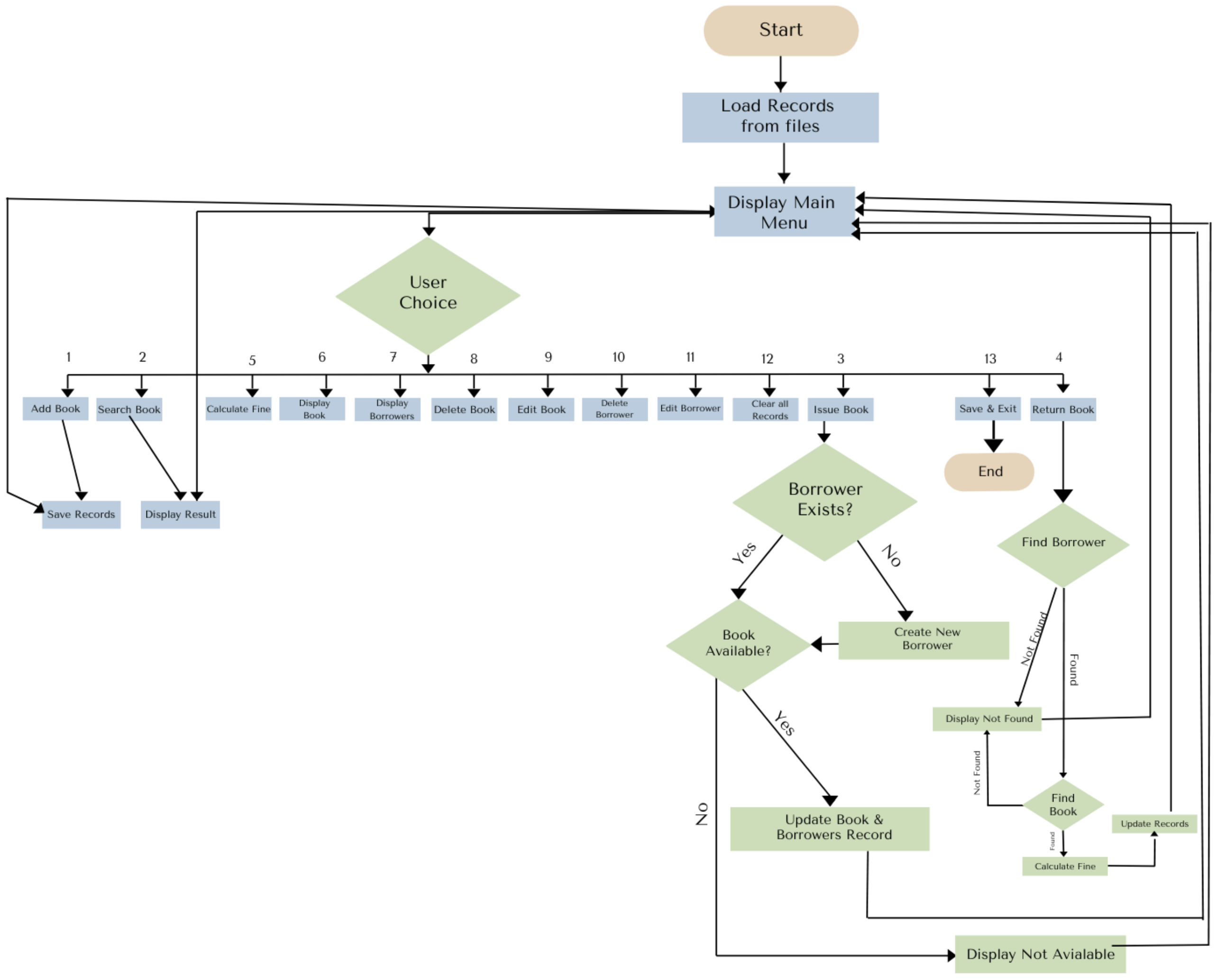
The main aim of this project is to create a Library Management System that simplifies and automates the process of managing books and borrower records.

Objectives

- **Add Books:** Allow librarians to add new books to the library's inventory.
- **Search Books:** Enable users to search for books by name, author, or category.
- **Issue Books:** Provide a way to issue books to borrowers and track issue and return dates.
- **Return Books:** Allow borrowers to return books and calculate fines for late returns.
- **Display Records**: Display lists of all books and borrowers.
- **Edit Records:** Allow librarians to edit book and borrower details.
- **Delete Records**: Provide functionality to remove books and borrowers from the system.
- **Data Persistence**: Ensure that all records are saved and loaded correctly between sessions using file I/O.

By achieving these objectives, the project aims to make library management more efficient and user-friendly.

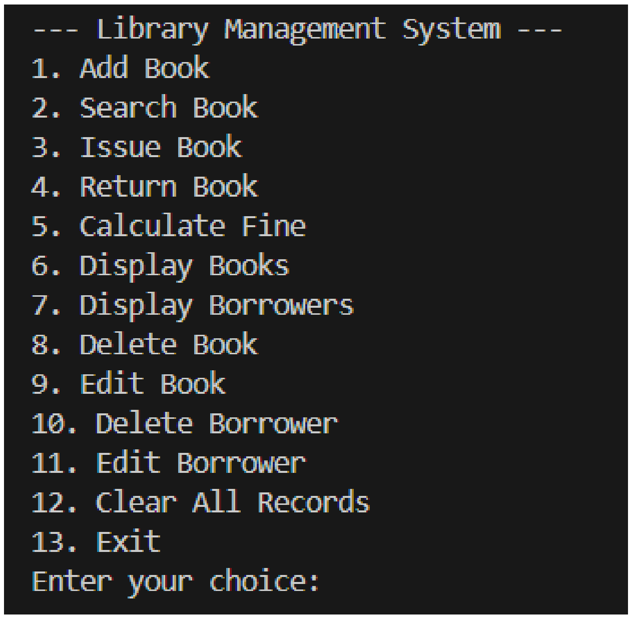
Flowchart



Start
Load Records from files
Display Main Menu
User Choice
1
2
5
6
7
8
9
10
11
12
3
13
4
Add Book
Search Book
Calculate Fine
Display Book
Display Borrowers
Delete Book
Edit Book
Delete Borrower
Edit Borrower
Clear all Records
Issue Book
Save & Exit
Return Book
Save Records
Display Result
End
Borrower Exists?
Yes
No
Find Borrower
Book Available?
Create New Borrower
Not Found
Found
Display Not Found
Yes
Not Found
Find Book
Update Records
No
Update Book & Borrowers Record
Found
Calculate Fine
Display Not Avialable

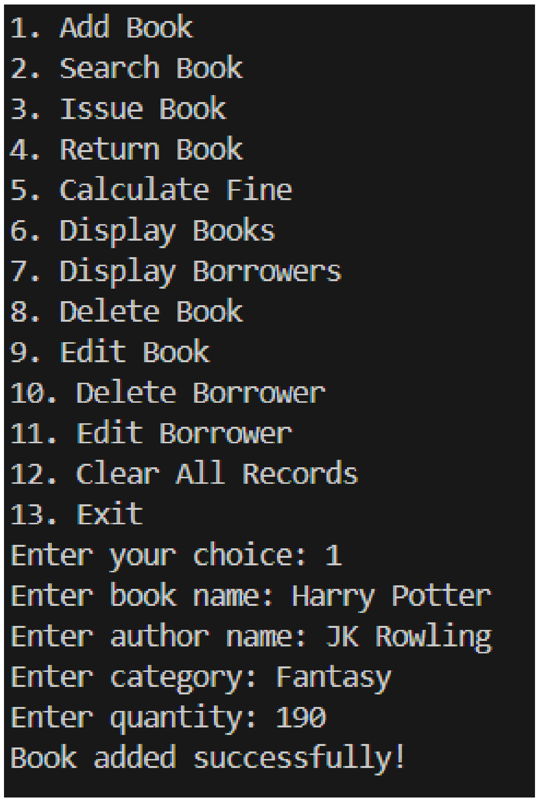
Program Output Screenshot

Main Menu:



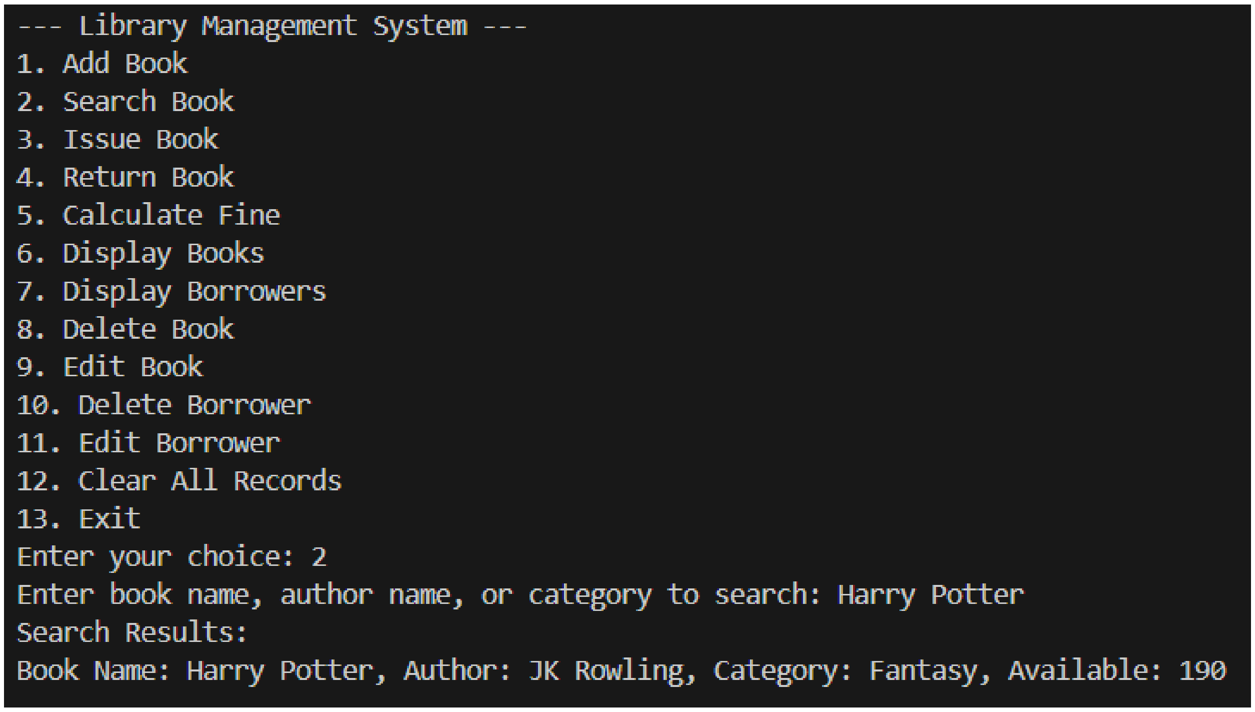
```
--- Library Management System ---
1. Add Book
2. Search Book
3. Issue Book
4. Return Book
5. Calculate Fine
6. Display Books
7. Display Borrowers
8. Delete Book
9. Edit Book
10. Delete Borrower
11. Edit Borrower
12. Clear All Records
13. Exit
Enter your choice:
```

Add Book:



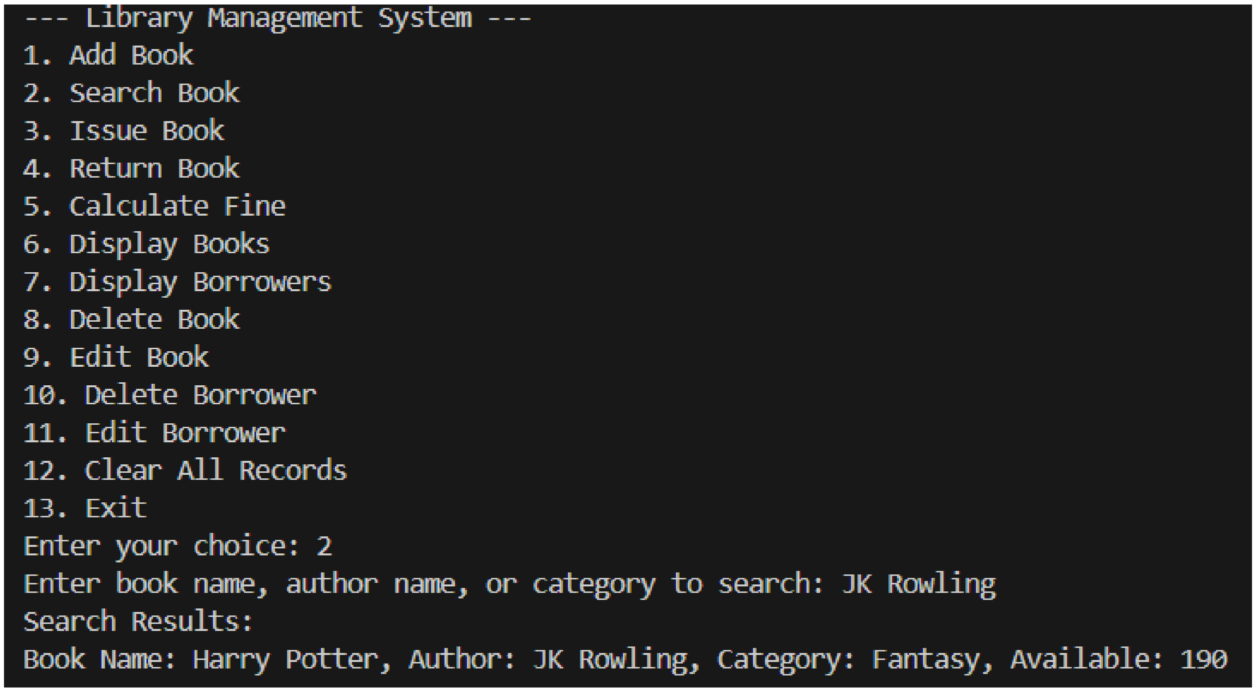
```
1. Add Book
2. Search Book
3. Issue Book
4. Return Book
5. Calculate Fine
6. Display Books
7. Display Borrowers
8. Delete Book
9. Edit Book
10. Delete Borrower
11. Edit Borrower
12. Clear All Records
13. Exit
Enter your choice: 1
Enter book name: Harry Potter
Enter author name: JK Rowling
Enter category: Fantasy
Enter quantity: 190
Book added successfully!
```

Searching Books by Book Name:



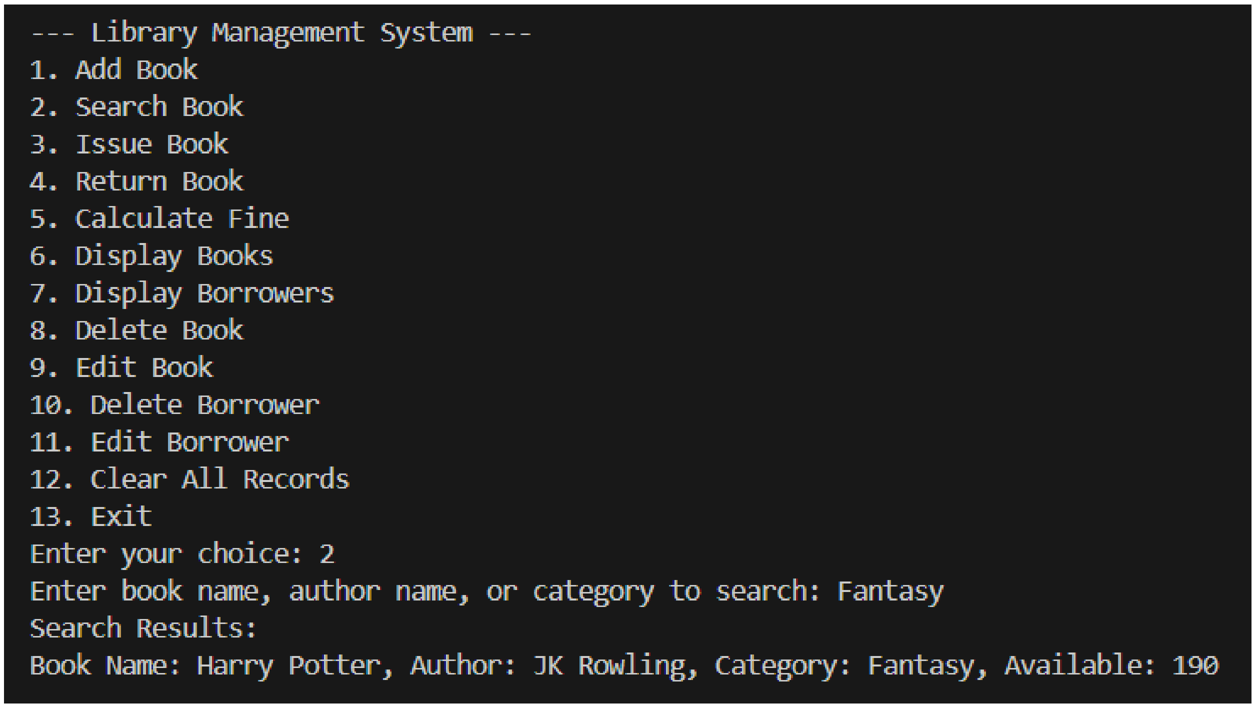
```
--- Library Management System ---
1. Add Book
2. Search Book
3. Issue Book
4. Return Book
5. Calculate Fine
6. Display Books
7. Display Borrowers
8. Delete Book
9. Edit Book
10. Delete Borrower
11. Edit Borrower
12. Clear All Records
13. Exit
Enter your choice: 2
Enter book name, author name, or category to search: Harry Potter
Search Results:
Book Name: Harry Potter, Author: JK Rowling, Category: Fantasy, Available: 190
```

Searching Books by Book Name:



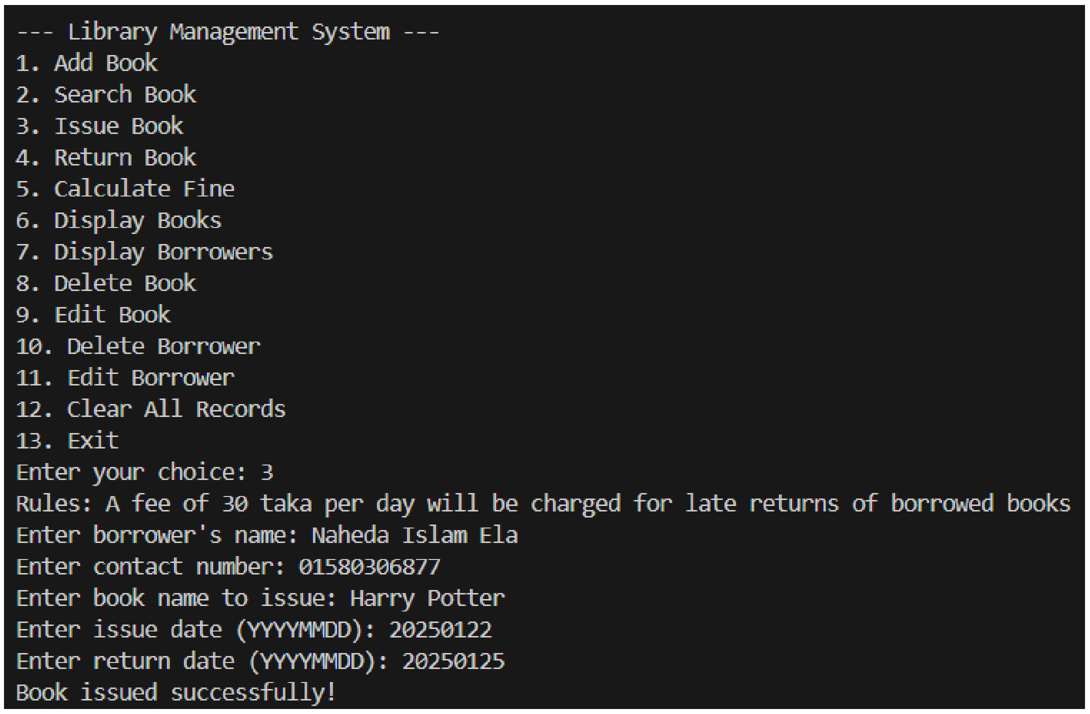
```
--- Library Management System ---
1. Add Book
2. Search Book
3. Issue Book
4. Return Book
5. Calculate Fine
6. Display Books
7. Display Borrowers
8. Delete Book
9. Edit Book
10. Delete Borrower
11. Edit Borrower
12. Clear All Records
13. Exit
Enter your choice: 2
Enter book name, author name, or category to search: JK Rowling
Search Results:
Book Name: Harry Potter, Author: JK Rowling, Category: Fantasy, Available: 190
```

Searching Books by Category:



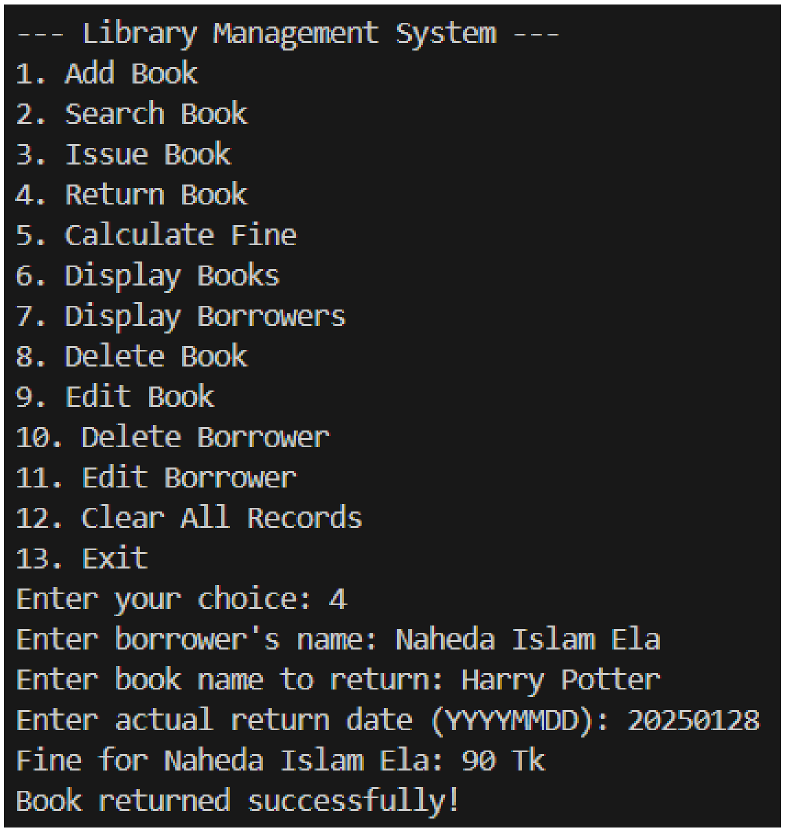
```
--- Library Management System ---
1. Add Book
2. Search Book
3. Issue Book
4. Return Book
5. Calculate Fine
6. Display Books
7. Display Borrowers
8. Delete Book
9. Edit Book
10. Delete Borrower
11. Edit Borrower
12. Clear All Records
13. Exit
Enter your choice: 2
Enter book name, author name, or category to search: Fantasy
Search Results:
Book Name: Harry Potter, Author: JK Rowling, Category: Fantasy, Available: 190
```

Issue Book:



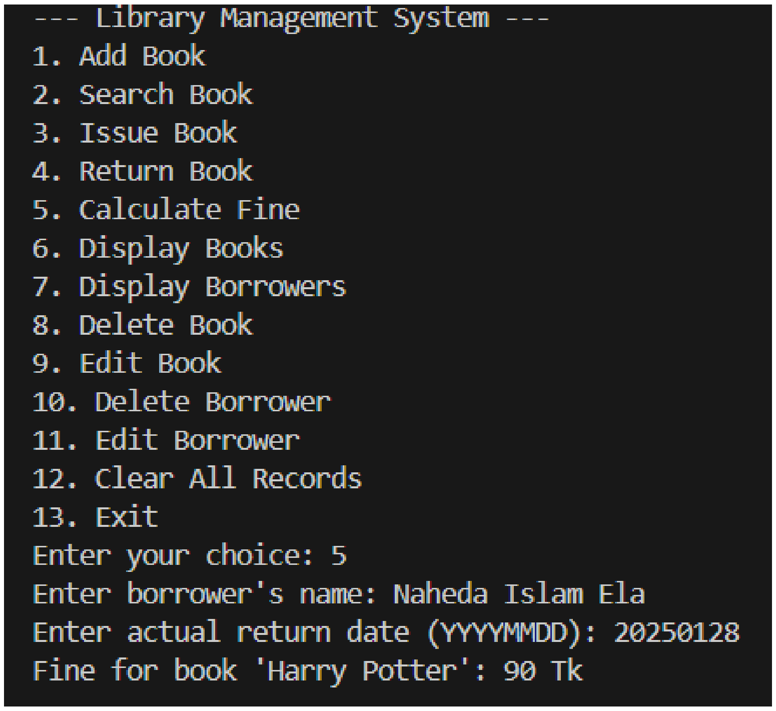
```
--- Library Management System ---
1. Add Book
2. Search Book
3. Issue Book
4. Return Book
5. Calculate Fine
6. Display Books
7. Display Borrowers
8. Delete Book
9. Edit Book
10. Delete Borrower
11. Edit Borrower
12. Clear All Records
13. Exit
Enter your choice: 3
Rules: A fee of 30 taka per day will be charged for late returns of borrowed books
Enter borrower's name: Naheda Islam Ela
Enter contact number: 01580306877
Enter book name to issue: Harry Potter
Enter issue date (YYYYMMDD): 20250122
Enter return date (YYYYMMDD): 20250125
Book issued successfully!
```

Return Book:



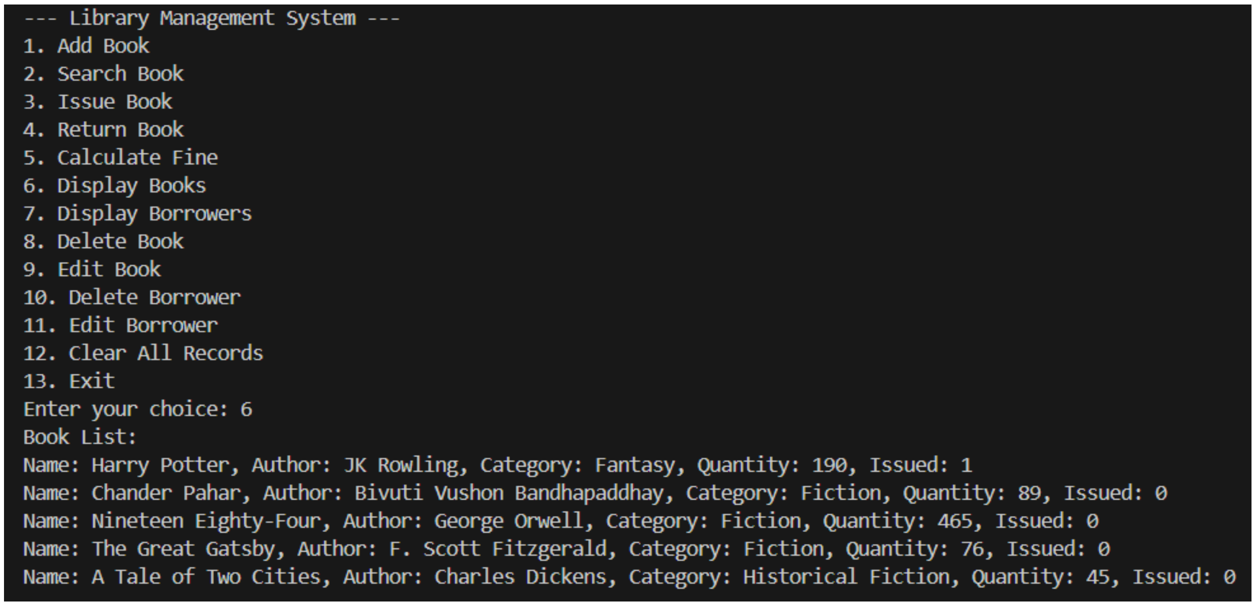
```
--- Library Management System ---
1. Add Book
2. Search Book
3. Issue Book
4. Return Book
5. Calculate Fine
6. Display Books
7. Display Borrowers
8. Delete Book
9. Edit Book
10. Delete Borrower
11. Edit Borrower
12. Clear All Records
13. Exit
Enter your choice: 4
Enter borrower's name: Naheda Islam Ela
Enter book name to return: Harry Potter
Enter actual return date (YYYYMMDD): 20250128
Fine for Naheda Islam Ela: 90 Tk
Book returned successfully!
```

Calculating Fine:



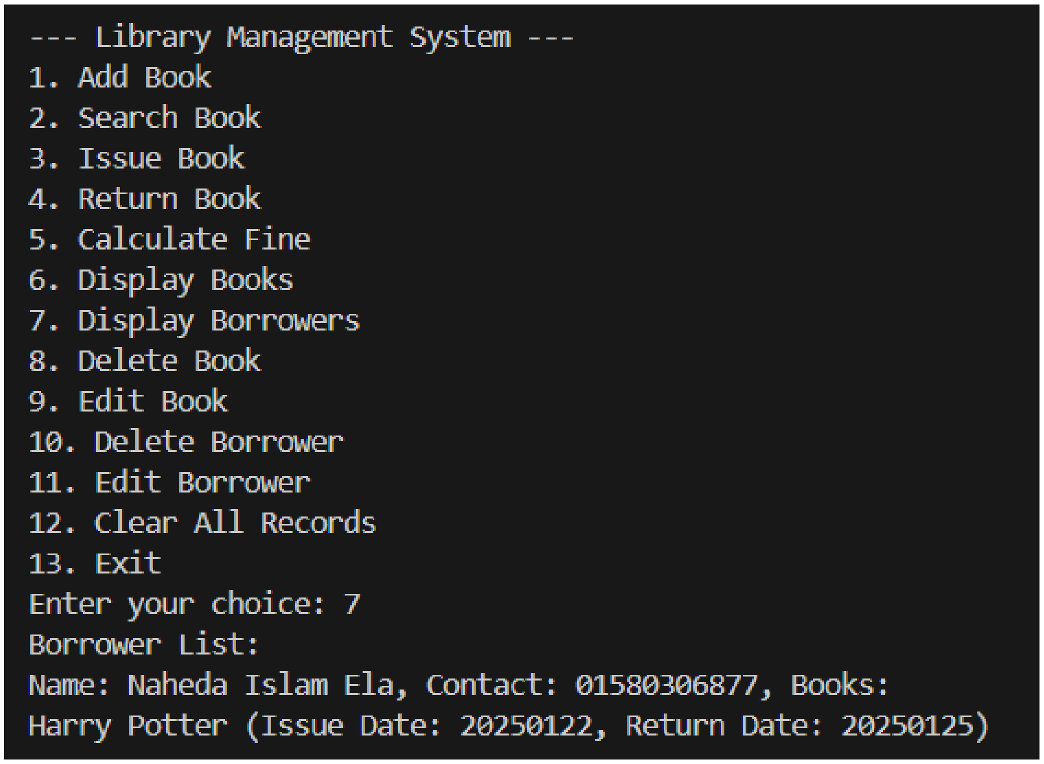
```
--- Library Management System ---
1. Add Book
2. Search Book
3. Issue Book
4. Return Book
5. Calculate Fine
6. Display Books
7. Display Borrowers
8. Delete Book
9. Edit Book
10. Delete Borrower
11. Edit Borrower
12. Clear All Records
13. Exit
Enter your choice: 5
Enter borrower's name: Naheda Islam Ela
Enter actual return date (YYYYMMDD): 20250128
Fine for book 'Harry Potter': 90 Tk
```

Display Books:



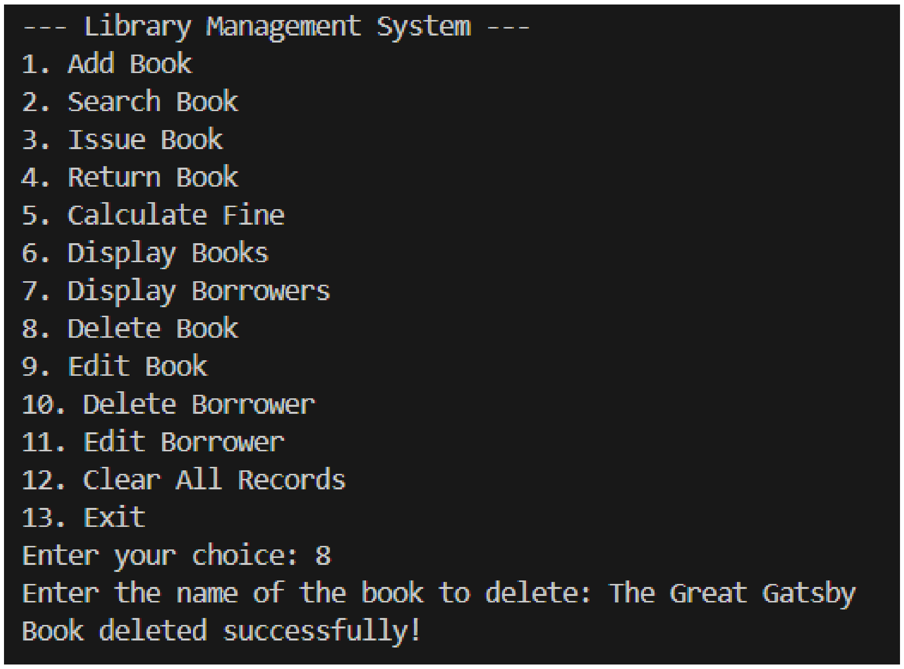
```
--- Library Management System ---
1. Add Book
2. Search Book
3. Issue Book
4. Return Book
5. Calculate Fine
6. Display Books
7. Display Borrowers
8. Delete Book
9. Edit Book
10. Delete Borrower
11. Edit Borrower
12. Clear All Records
13. Exit
Enter your choice: 6
Book List:
Name: Harry Potter, Author: JK Rowling, Category: Fantasy, Quantity: 190, Issued: 1
Name: Chander Pahar, Author: Bivuti Vushon Bandhapaddhay, Category: Fiction, Quantity: 89, Issued: 0
Name: Nineteen Eighty-Four, Author: George Orwell, Category: Fiction, Quantity: 465, Issued: 0
Name: The Great Gatsby, Author: F. Scott Fitzgerald, Category: Fiction, Quantity: 76, Issued: 0
Name: A Tale of Two Cities, Author: Charles Dickens, Category: Historical Fiction, Quantity: 45, Issued: 0
```

Display Borrower:



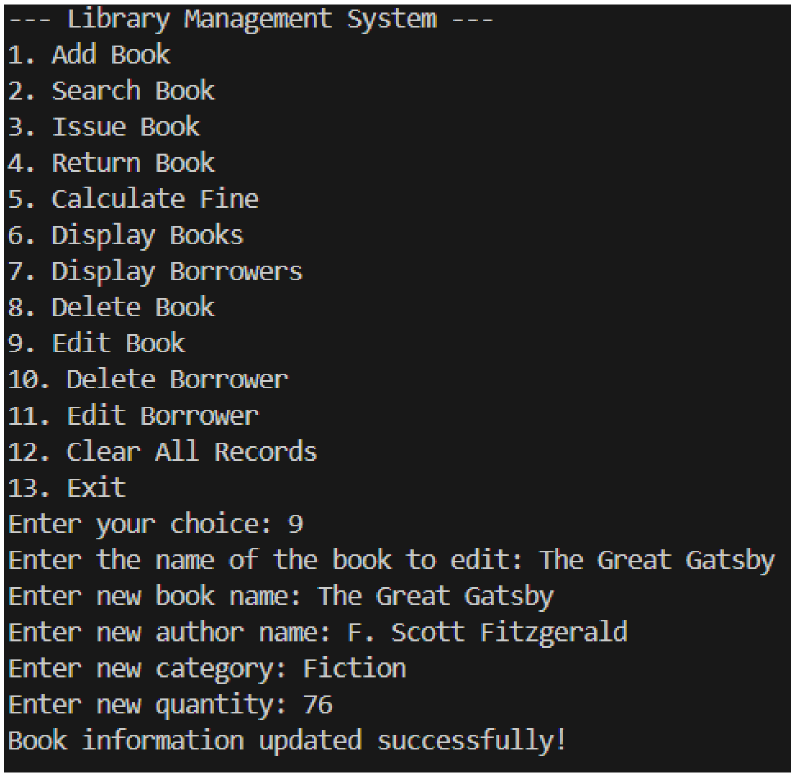
```
--- Library Management System ---
1. Add Book
2. Search Book
3. Issue Book
4. Return Book
5. Calculate Fine
6. Display Books
7. Display Borrowers
8. Delete Book
9. Edit Book
10. Delete Borrower
11. Edit Borrower
12. Clear All Records
13. Exit
Enter your choice: 7
Borrower List:
Name: Naheda Islam Ela, Contact: 01580306877, Books:
Harry Potter (Issue Date: 20250122, Return Date: 20250125)
```

Deleting Book:



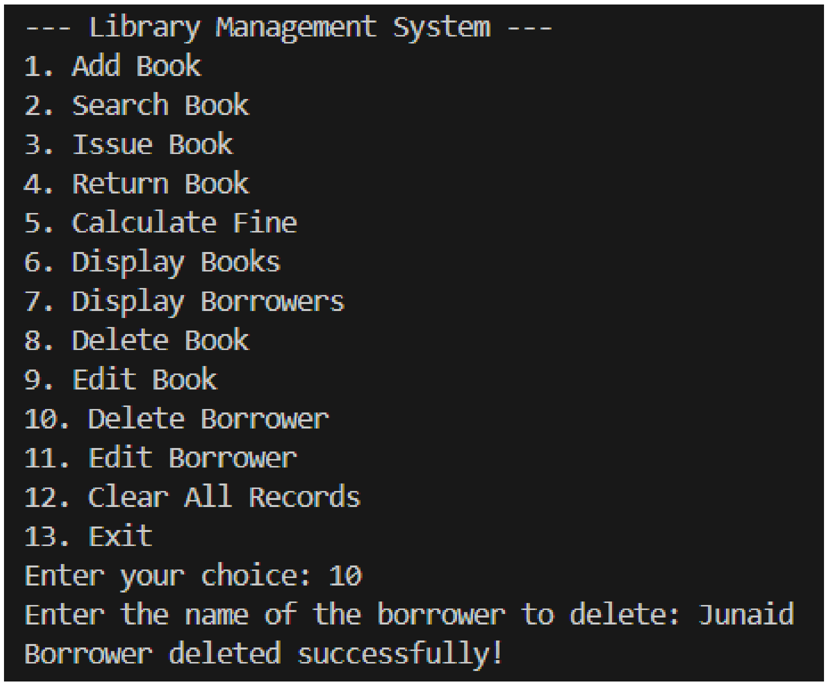
```
--- Library Management System ---
1. Add Book
2. Search Book
3. Issue Book
4. Return Book
5. Calculate Fine
6. Display Books
7. Display Borrowers
8. Delete Book
9. Edit Book
10. Delete Borrower
11. Edit Borrower
12. Clear All Records
13. Exit
Enter your choice: 8
Enter the name of the book to delete: The Great Gatsby
Book deleted successfully!
```

Edit Book Information:



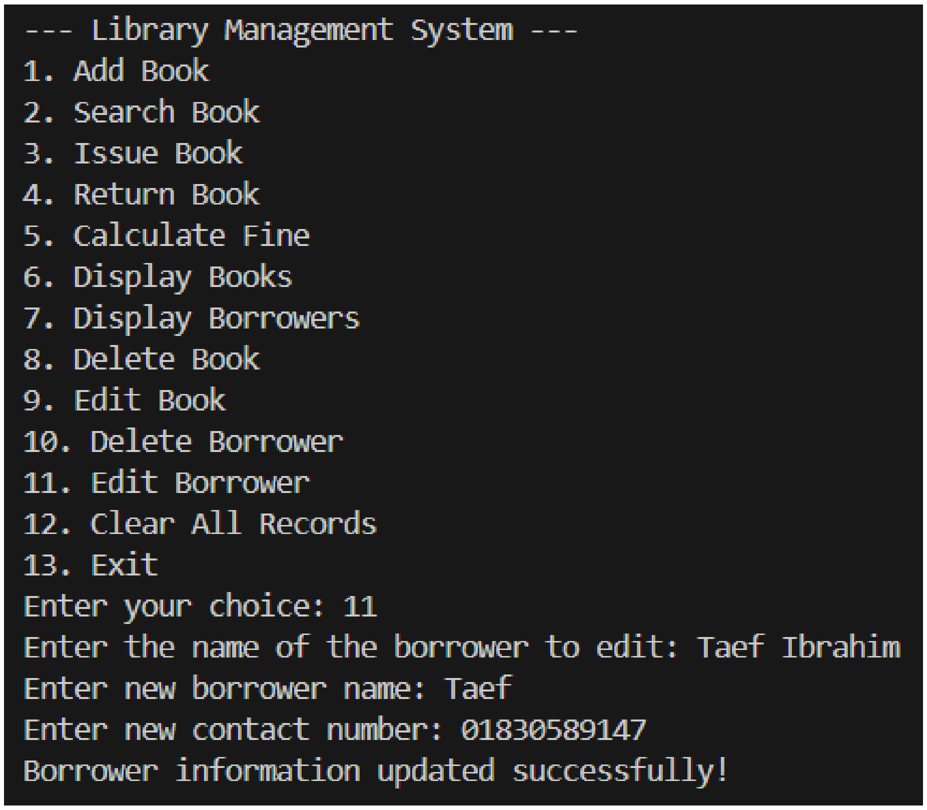
```
--- Library Management System ---
1. Add Book
2. Search Book
3. Issue Book
4. Return Book
5. Calculate Fine
6. Display Books
7. Display Borrowers
8. Delete Book
9. Edit Book
10. Delete Borrower
11. Edit Borrower
12. Clear All Records
13. Exit
Enter your choice: 9
Enter the name of the book to edit: The Great Gatsby
Enter new book name: The Great Gatsby
Enter new author name: F. Scott Fitzgerald
Enter new category: Fiction
Enter new quantity: 76
Book information updated successfully!
```

Delete Borrower:



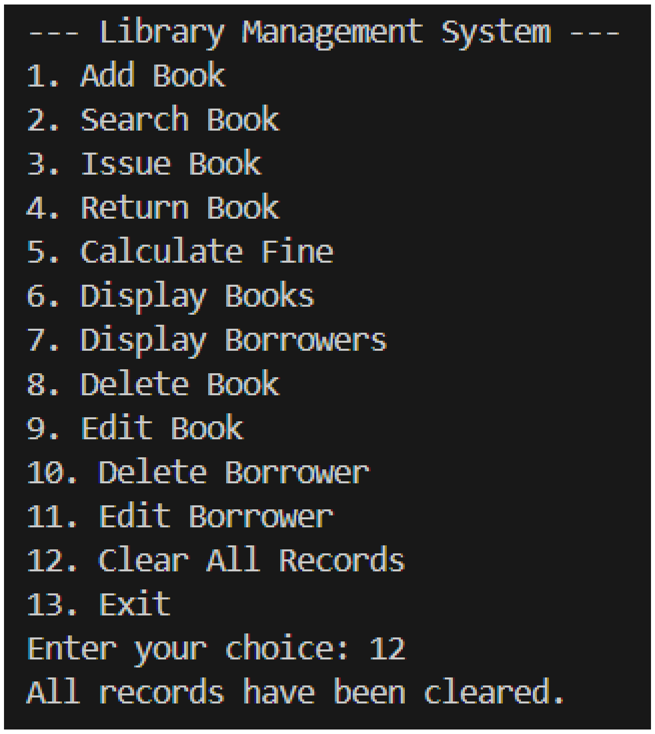
```
--- Library Management System ---
1. Add Book
2. Search Book
3. Issue Book
4. Return Book
5. Calculate Fine
6. Display Books
7. Display Borrowers
8. Delete Book
9. Edit Book
10. Delete Borrower
11. Edit Borrower
12. Clear All Records
13. Exit
Enter your choice: 10
Enter the name of the borrower to delete: Junaid
Borrower deleted successfully!
```

Edit Borrowers Information:



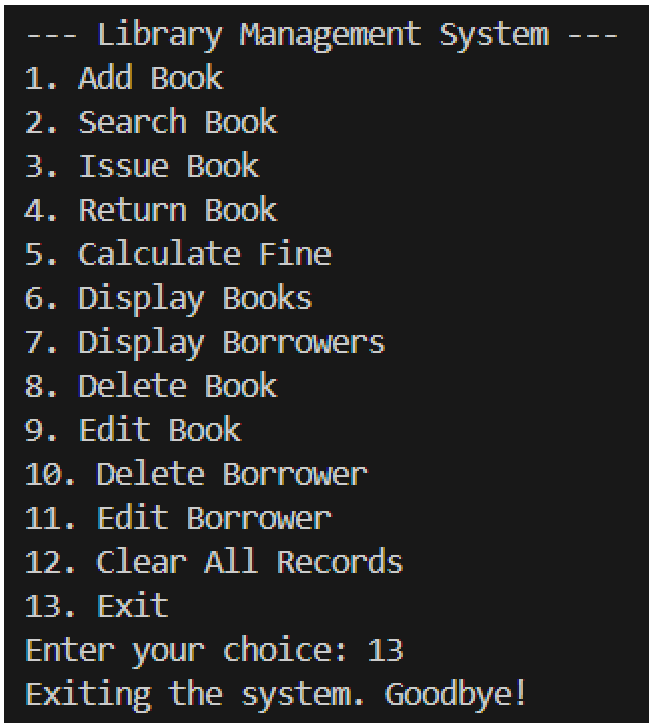
```
--- Library Management System ---
1. Add Book
2. Search Book
3. Issue Book
4. Return Book
5. Calculate Fine
6. Display Books
7. Display Borrowers
8. Delete Book
9. Edit Book
10. Delete Borrower
11. Edit Borrower
12. Clear All Records
13. Exit
Enter your choice: 11
Enter the name of the borrower to edit: Taef Ibrahim
Enter new borrower name: Taef
Enter new contact number: 01830589147
Borrower information updated successfully!
```

Clear All Records:



```
--- Library Management System ---
1. Add Book
2. Search Book
3. Issue Book
4. Return Book
5. Calculate Fine
6. Display Books
7. Display Borrowers
8. Delete Book
9. Edit Book
10. Delete Borrower
11. Edit Borrower
12. Clear All Records
13. Exit
Enter your choice: 12
All records have been cleared.
```

Exiting the Program:



```
--- Library Management System ---
1. Add Book
2. Search Book
3. Issue Book
4. Return Book
5. Calculate Fine
6. Display Books
7. Display Borrowers
8. Delete Book
9. Edit Book
10. Delete Borrower
11. Edit Borrower
12. Clear All Records
13. Exit
Enter your choice: 13
Exiting the system. Goodbye!
```

Source Code

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

// Structure for Book records
struct Book
{
    char name[50];
    char author[50];
    char category[30];
    int quantity;
    int issued;
};

// Structure for Borrower records
struct Borrower
{
    char name[50];
    char contact[20];
    char bookNames[100][50]; // Array to store multiple book names
    int issueDates[100];     // Array to store issue dates for each book
    int returnDates[100];    // Array to store return dates for each book
    int bookCount;           // Number of books borrowed
};

// Global variables and file pointers
struct Book books[100];
struct Borrower borrowers[100];
int bookCount = 0, borrowerCount = 0;

// File names
const char *bookFile = "books.txt";
const char *borrowerFile = "borrowers.txt";

// Function prototypes
void loadRecords();
void saveRecords();
void addBook();
void searchBook();
void issueBook();
void returnBook();
void calculateFine();
void displayBooks();
void displayBorrowers();
void deleteBook();
void editBook();
void deleteBorrower();
void editBorrower();
void clearRecords();

// Load records from files
void loadRecords()
{
    FILE *fp;
    fp = fopen(bookFile, "r");
    if (fp != NULL)
    {
        fread(&bookCount, sizeof(int), 1, fp);
        fread(books, sizeof(struct Book), bookCount, fp);
        fclose(fp);
    }

    fp = fopen(borrowerFile, "r");
    if (fp != NULL)
    {
        fread(&borrowerCount, sizeof(int), 1, fp);
        fread(borrowers, sizeof(struct Borrower), borrowerCount, fp);
        fclose(fp);
    }
}

// Save records to files
void saveRecords()
{
    FILE *fp;
    fp = fopen(bookFile, "w");
    fwrite(&bookCount, sizeof(int), 1, fp);
    fwrite(books, sizeof(struct Book), bookCount, fp);
    fclose(fp);

    fp = fopen(borrowerFile, "w");
    fwrite(&borrowerCount, sizeof(int), 1, fp);
    fwrite(borrowers, sizeof(struct Borrower), borrowerCount, fp);
    fclose(fp);
}

// Add a new book
void addBook()
{
    printf("Enter book name: ");
    scanf(" %[^\n]", books[bookCount].name);
    printf("Enter author name: ");
    scanf(" %[^\n]", books[bookCount].author);
    printf("Enter category: ");
    scanf(" %[^\n]", books[bookCount].category);
    printf("Enter quantity: ");
    scanf("%d", &books[bookCount].quantity);
    books[bookCount].issued = 0;
    bookCount++;
    saveRecords();
    printf("Book added successfully!\n");
}

// Search for a book
void searchBook()
{
    char searchKey[50];
    printf("Enter book name, author name, or category to search: ");
    scanf(" %[^\n]", searchKey);
    printf("Search Results:\n");
    for (int i = 0; i < bookCount; i++)
    {
        if (strcmp(books[i].name, searchKey) == 0 || strcmp(books[i].author,
searchKey) == 0 || strcmp(books[i].category, searchKey) == 0)
        {
            printf("Book Name: %s, Author: %s, Category: %s, Available: %d\n",
                   books[i].name, books[i].author, books[i].category,
                   books[i].quantity - books[i].issued);
        }
    }
}

// Issue a book to a borrower
void issueBook()
{
    char borrowerName[50], contact[20], bookName[50];
    int issueDate, returnDate;


   printf("Rules: A fee of 30 taka per day will be charged for late returns of
borrowed books\n");

    printf("Enter borrower's name: ");
    scanf(" %[^\n]", borrowerName);
    printf("Enter contact number: ");
    scanf(" %[^\n]", contact);
    // Check if borrower already exists
    int borrowerIndex = -1;
    for (int i = 0; i < borrowerCount; i++)
    {
        if (strcmp(borrowers[i].name, borrowerName) == 0)
        {
            borrowerIndex = i;
            break;
        }
    }

    if (borrowerIndex == -1)
    {
        // New borrower
        borrowerIndex = borrowerCount++;
        strcpy(borrowers[borrowerIndex].name, borrowerName);
        strcpy(borrowers[borrowerIndex].contact, contact);
        borrowers[borrowerIndex].bookCount = 0;
    }

    printf("Enter book name to issue: ");
    scanf(" %[^\n]", bookName);

    // Find the book
    int found = -1;
    for (int i = 0; i < bookCount; i++) // bookCount - 1;
    {
        if (strcmp(books[i].name, bookName) == 0)
        {
            found = i;
            break;
        }
    }

    if (found == -1 || books[found].issued >= (books[found].quantity – 1))
    {
        printf("Book not available.\n");
        return;
    }

    printf("Enter issue date (YYYYMMDD): ");
    scanf("%d", &issueDate);
    printf("Enter return date (YYYYMMDD): ");
    scanf("%d", &returnDate);

    // Update book and borrower records
    books[found].issued++;
```

```c
    strcpy(borrowers[borrowerIndex].bookNames[borrowers[borrowerIndex].bookCount]
, bookName);
    borrowers[borrowerIndex].issueDates[borrowers[borrowerIndex].bookCount] =
issueDate;
    borrowers[borrowerIndex].returnDates[borrowers[borrowerIndex].bookCount] =
returnDate;
    borrowers[borrowerIndex].bookCount++;
    saveRecords();
    printf("Book issued successfully!\n");
}

// Return a book
void returnBook()
{
    char borrowerName[50], bookName[50];
    int actualReturnDate;
    printf("Enter borrower's name: ");
    scanf(" %[^\n]", borrowerName);

    // Find the borrower
    int borrowerIndex = -1;
    for (int i = 0; i < borrowerCount; i++)
    {
        if (strcmp(borrowers[i].name, borrowerName) == 0)
        {
            borrowerIndex = i;
            break;
        }
    }

    if (borrowerIndex == -1)
    {
        printf("Borrower not found.\n");
        return;
    }

    printf("Enter book name to return: ");
    scanf(" %[^\n]", bookName);

    // Find the book in borrower's record
    int bookIndex = -1;
    for (int i = 0; i < borrowers[borrowerIndex].bookCount; i++)
    {
        if (strcmp(borrowers[borrowerIndex].bookNames[i], bookName) == 0)
        {
            bookIndex = i;
            break;
        }
    }

    if (bookIndex == -1)
    {
        printf("Book not found in borrower's record.\n");
        return;
    }

    printf("Enter actual return date (YYYYMMDD): ");
    scanf("%d", &actualReturnDate);

    // Calculate fine
    int fine = 0;
    if (actualReturnDate > borrowers[borrowerIndex].returnDates[bookIndex])
    {
        fine = (actualReturnDate -
borrowers[borrowerIndex].returnDates[bookIndex]) * 30; // 30 Tk per day
    }
    printf("Fine for %s: %d Tk\n", borrowerName, fine);

    // Update book and borrower records
    for (int i = 0; i < bookCount; i++)
    {
        if (strcmp(books[i].name, bookName) == 0)
        {
            books[i].issued--;
            break;
        }
    }

    // Remove the book from borrower's record
    for (int i = bookIndex; i < borrowers[borrowerIndex].bookCount - 1; i++)
    {
        strcpy(borrowers[borrowerIndex].bookNames[i],
borrowers[borrowerIndex].bookNames[i + 1]);
        borrowers[borrowerIndex].issueDates[i] =
borrowers[borrowerIndex].issueDates[i + 1];
        borrowers[borrowerIndex].returnDates[i] =
borrowers[borrowerIndex].returnDates[i + 1];
    }
    borrowers[borrowerIndex].bookCount--;

    // If borrower has no more books, remove the borrower
    if (borrowers[borrowerIndex].bookCount == 0)
    {
        for (int i = borrowerIndex; i < borrowerCount - 1; i++)
        {
            borrowers[i] = borrowers[i + 1];
```

```c
        }
        borrowerCount--;
    }

    saveRecords();
    printf("Book returned successfully!\n");
}

// Calculate fine for late return
void calculateFine()
{
    char borrowerName[50];
    int actualReturnDate;
    printf("Enter borrower's name: ");
    scanf(" %[^\n]", borrowerName);
    printf("Enter actual return date (YYYYMMDD): ");
    scanf("%d", &actualReturnDate);

    for (int i = 0; i < borrowerCount; i++)
    {
        if (strcmp(borrowers[i].name, borrowerName) == 0)
        {
            for (int j = 0; j < borrowers[i].bookCount; j++)
            {
                int fine = 0;
                if (actualReturnDate > borrowers[i].returnDates[j])
                {
                    fine = (actualReturnDate - borrowers[i].returnDates[j]) * 30;
// 30 Tk per day
                }
                printf("Fine for book '%s': %d Tk\n", borrowers[i].bookNames[j],
fine);
            }
            return;
        }
    }
    printf("Borrower not found.\n");
}

// Display all books
void displayBooks()
{
    printf("Book List:\n");
    for (int i = 0; i < bookCount; i++)
    {
        printf("Name: %s, Author: %s, Category: %s, Quantity: %d, Issued: %d\n",
               books[i].name, books[i].author, books[i].category,
               books[i].quantity, books[i].issued);
    }
}

// Display all borrowers
void displayBorrowers()
{
    printf("Borrower List:\n");
    for (int i = 0; i < borrowerCount; i++)
    {
        printf("Name: %s, Contact: %s, Books: \n", borrowers[i].name,
borrowers[i].contact);
        for (int j = 0; j < borrowers[i].bookCount; j++)
        {
            printf("%s (Issue Date: %d, Return Date: %d)",
borrowers[i].bookNames[j], borrowers[i].issueDates[j],
borrowers[i].returnDates[j]);
            if (j < borrowers[i].bookCount - 1)
            {
                printf(", ");
            }
        }
        printf("\n");
    }
}

// Delete a book from the list
void deleteBook()
{
    char bookName[50];
    printf("Enter the name of the book to delete: ");
    scanf(" %[^\n]", bookName);

    int found = -1;
    for (int i = 0; i < bookCount; i++)
    {
        if (strcmp(books[i].name, bookName) == 0)
        {
            found = i;
            break;
        }
    }

    if (found == -1)
    {
        printf("Book not found.\n");
        return;
    }
    for (int i = found; i < bookCount - 1; i++)
    {
        books[i] = books[i + 1];
    }
    bookCount--;
    saveRecords();
    printf("Book deleted successfully!\n");
}

// Edit a book's information
void editBook()
{
    char bookName[50];
    printf("Enter the name of the book to edit: ");
    scanf(" %[^\n]", bookName);

    int found = -1;
    for (int i = 0; i < bookCount; i++)
    {
        if (strcmp(books[i].name, bookName) == 0)
        {
            found = i;
            break;
        }
    }

    if (found == -1)
    {
        printf("Book not found.\n");
        return;
    }

    printf("Enter new book name: ");
    scanf(" %[^\n]", books[found].name);
    printf("Enter new author name: ");
    scanf(" %[^\n]", books[found].author);
    printf("Enter new category: ");
    scanf(" %[^\n]", books[found].category);
    printf("Enter new quantity: ");
    scanf("%d", &books[found].quantity);
    saveRecords();
    printf("Book information updated successfully!\n");
}

// Delete a borrower from the list
void deleteBorrower()
{
    char borrowerName[50];
    printf("Enter the name of the borrower to delete: ");
    scanf(" %[^\n]", borrowerName);

    int found = -1;
    for (int i = 0; i < borrowerCount; i++)
    {
        if (strcmp(borrowers[i].name, borrowerName) == 0)
        {
            found = i;
            break;
        }
    }

    if (found == -1)
    {
        printf("Borrower not found.\n");
        return;
    }

    for (int i = found; i < borrowerCount - 1; i++)
    {
        borrowers[i] = borrowers[i + 1];
    }
    borrowerCount--;
    saveRecords();
    printf("Borrower deleted successfully!\n");
}

// Edit a borrower's information
void editBorrower()
{
    char borrowerName[50];
    printf("Enter the name of the borrower to edit: ");
    scanf(" %[^\n]", borrowerName);

    int found = -1;
    for (int i = 0; i < borrowerCount; i++)
    {
        if (strcmp(borrowers[i].name, borrowerName) == 0)
        {
            found = i;
            break;
        }
    }

    if (found == -1)
    {
        printf("Borrower not found.\n");
        return;
    }

    printf("Enter new borrower name: ");
    scanf(" %[^\n]", borrowers[found].name);
    printf("Enter new contact number: ");
    scanf(" %[^\n]", borrowers[found].contact);
    saveRecords();
    printf("Borrower information updated successfully!\n");
}

// Clear all records
void clearRecords()
{
    // Reset global variables
    bookCount = 0;
    borrowerCount = 0;

    // Clear the files
    FILE *fp;
    fp = fopen(bookFile, "wb");
    if (fp != NULL)
    {
        fwrite(&bookCount, sizeof(int), 1, fp);
        fclose(fp);
    }

    fp = fopen(borrowerFile, "wb");
    if (fp != NULL)
    {
        fwrite(&borrowerCount, sizeof(int), 1, fp);
        fclose(fp);
    }

    printf("All records have been cleared.\n");
}

// Main function
// Main function
int main()
{
    int choice;

    loadRecords(); // Load existing records from files

    while (1)
    {
        printf("\n--- Library Management System ---\n");
        printf("1. Add Book\n");
        printf("2. Search Book\n");
        printf("3. Issue Book\n");
        printf("4. Return Book\n");
        printf("5. Calculate Fine\n");
        printf("6. Display Books\n");
        printf("7. Display Borrowers\n");
        printf("8. Delete Book\n");
        printf("9. Edit Book\n");
        printf("10. Delete Borrower\n");
        printf("11. Edit Borrower\n");
        printf("12. Clear All Records\n");
        printf("13. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            addBook();
            break;
        case 2:
            searchBook();
            break;
        case 3:
            issueBook();
            break;
        case 4:
            returnBook();
            break;
        case 5:
            calculateFine();
            break;
        case 6:
            displayBooks();
            break;
        case 7:
            displayBorrowers();
            break;
        case 8:
            deleteBook();
            break;
        case 9:
            editBook();
            break;
        case 10:
            deleteBorrower();
            break;
        case 11:
            editBorrower();
            break;
        case 12:
            clearRecords();
            break;
        case 13:
            printf("Exiting the system. Goodbye!\n");
            saveRecords(); // Save records before exiting
            exit(0);
            break;
        default:
            printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}
```

Limitations of Program

- **Fine Calculation:** The program fails to calculate fines correctly if the month changes between the issue date and the return date.

- **Case-Sensitive Search:** The program's search functionality is case-sensitive. It cannot handle dynamic searches where variations in capitalization (e.g., "Mahin", "mahin", "mAhin", "maHin") should be considered equivalent.

- **Limited User Interface:** The program uses a text-based interface, which may not be as user-friendly as a graphical user interface (GUI).

- **Input Validation:** The program crashes if incorrect data types are entered (e.g., entering a character where an integer is expected), requiring the terminal to be killed and the code to be rerun.

Extra Features

- **Display Books:** This feature allows users to view a list of all the books available in the library, including details like the book's name, author, category, quantity, and the number of copies issued.
- **Display Borrowers:** This feature provides a list of all the borrowers registered in the library system, along with their contact information and the details of the books they have borrowed.
- **Delete Book:** This feature enables users to remove a book record from the library system. It helps in keeping the library's inventory up-to-date by removing outdated or unavailable books.
- **Edit Book:** This feature allows users to update the details of an existing book in the library system, such as changing the book's name, author, category, quantity, or issued count.
- **Delete Borrowers:** This feature allows users to remove a borrower record from the library system. It helps in maintaining an accurate list of active borrowers.
- **Edit Borrowers:** This feature enables users to update the details of an existing borrower, such as their name, contact information, or the list of books they have borrowed.
- **Clear All Records:** This feature provides an option to delete all book and borrower records from the library system. It is useful for resetting the system or starting fresh.
- **Showing the Rules While Issuing the Book:** This feature displays the library's rules and guidelines to the user when they are issuing a book. It ensures that borrowers are aware of the policies regarding book borrowing and returns.
- **Single Copy Restriction:** If only one copy of a book is left in stock, it cannot be borrowed. The system will notify the user that the book is not available for borrowing.

Conclusion

This Library Management System (LMS) project successfully achieved its objectives by providing a comprehensive solution for managing library operations. The system includes essential features such as displaying books and borrowers, adding, editing, and deleting records, and issuing books with clear rules. These functionalities ensure efficient management of library resources and enhance user experience. The project demonstrates effective use of data structures and file handling in C, resulting in a robust and user-friendly application. Overall, the LMS project is a valuable tool for libraries to streamline their operations and improve service delivery.