
# ST. MARY'S UNIVERSITY

## PROGRAMMING PROJECT

## LIBRARY MANAGEMENT SYSTEM (LMS)

## GROUP 1 SECTION B


| Group Members | ID No |
|---|---|
| 1. Dina Daniel | RCD/2494/2016 |
| 2. Dagimawi Mekonnen | RCD/0182/2017 |
| 3. Eyerusalem Hailu | RCD/0190/2017 |
| 4. Dagimawi G/Mariam | RCD/0183/2017 |
| 5. Meareg Abrha | RCD/0201/2017 |
| 6. Yeabsira Sinye | RCD/0224/2017 |
| 7. Hafsa Shamil | RCD/0194/2017 |
| 8. Mikiyas Endale | RCD/0206/2017 |
| 9. Abdulhafiz Kedir | RCD/2634/2016 |


**Submitted to: Mr. Dawit**

**Submission date: Friday, July 11, 2025 G.C**

# Library Management System (LMS)

## Introduction

In this project, we are going to see how Library Management System of St. Mary's University library enables user to add, view and issue books specifically for computer science department students using C++ program and we will try to see their functionalities and key points of each code snippets.

## Overview

-**Library Management System (LMS)** is a software application used by libraries to manage their resources efficiently.

-It helps librarians organize, catalogue, and circulate books, journals, media, and other materials available in the library.

- In a library system, librarians can perform various tasks such as adding new books to the catalogue, updating existing records, checking items in and out, and managing user accounts.

## Functionalists of LMS

**1) Catalog Management**: function allows users to search for books, journals, media, and other resources. It usually supports advanced search filters, and categorization by genres, authors, and topics.

**2) Circulation Management**: handles book loans, returns, renewals, and reservations. It helps track which items are in circulation and which are available, allowing both the library and its users to manage resources effectively.

**3) Acquisition and Inventory**: helps us track new purchases, donations, and other acquisitions while ensuring that all items are correctly inventoried.

**4) Member Management**: Libraries serve members, and the LMS keeps track of membership details, fines, borrowing history, and communication with patrons. Whether it's sending due date reminders or promotional emails about upcoming events, the Library Management System simplifies member interactions.

**5) Reports and Analytics**: To run efficiently, libraries need data. This feature provides detailed reports on circulation, member activity, inventory status, and more, helping you make informed decisions based on real-time information.

## 1) Header Includes and Namespace

```
1   #include <iostream>
2   #include<cstring>
3   using namespace std;
```

**Functionality**

-<iostream>: Input/output (cin, cout)

-<cstring>: For C-style string manipulation like strcpy()

-using namespace std: Removes the need to write std:: before every command

## 2) Book Structure and Initialization

```
struct Book {
    char title[50];
    char author[50];
    bool isIssued;
};
```

**Functionality**

-Stores basic book information using fixed-size character arrays

- Title: Holds the book title (max 49 chars + null terminator)
- Author: Stores author name (same size limit)
- isIssued: Boolean flag (false=available, true=checked out)

-Uses C-style strings (char []) instead of C++ string class

## 3) Global Variables

```
Book books[MAX_BOOKS];
int bookCount = 0;
```

**Functionality**

-Fixed-capacity array (MAX_BOOKS = 100)

-BookCount acts as both counter and next available index

-Potential improvement: Use dynamic container like vector<Book>

## 4) Core Functions

a) Pre-load books:

### a) Add Book

```
void addBook() {
    if (bookCount < MAX_BOOKS) {
        Book newBook;
        cin>> ignore();  // Clear newline
        cout << "Enter Book Title: ";
        cin>> getline(newBook.title, 50);
        cout << "Enter Book Author: ";
        cin>> getline(newBook.author, 50);
        newBook.isIssued = false;

        books[bookCount++] = newBook;
        cout << "Book added successfully: " << end
    } else {
        cout << "Book list is full: "<< endl;
    }
```

### Functionality

- Checks array capacity first

-Uses cin.ignore() to handle leftover newline characters

-Gets input with cin.getline() for multi-word titles/authors

-New books are automatically marked available

-Increments bookCount after successful addition

b) Viewbooks

```
void viewBooks() {
    if (bookCount == 0) {
        cout << "No books available.\n";
        return;
    }
    for (int i = 0; i < bookCount; i++) {
        cout << "Index: " << i
             << ", Title: " << books[i].title
             << ", Author: " << books[i].author;
        cout << (books[i].isIssued ? " (Issued)" : " (Available)") << endl;
    }
```

## Functionality

-Handles empty library case

-Displays all books in formatted list

-Shows array index (used for issuing books)

-Uses ternary operator for availability status

-Lists books in insertion order (no sorting)

## c) Issue book

```
void issueBook(int index) {
    if (index >= 0 && index < bookCount && !books[index].
        books[index].isIssued = true;
        cout << "Book issued successfully." << endl;
    } else {
        cout << "Book cannot be issued." << endl;
```

## Functionality

Index ≥ 0

Index < bookCount

Book is available

Minimal state change (only flips isIssued flag)

Generic error message (could be more specific)

Doesn't track who borrowed the bookrwise, shows an error message

## 5) Main Function

```
int main() {
    preloadBooks();  // Load 20 books at program start

cout << "ST.MARY UNIVERSITY" << endl << "COMPUTER SCIENCE DEPARTMENT";

    int choice;
    do {
        cout << "\nLibrary Management System\n";
        cout << "1. View Books\n";
        cout << "2. Add Book\n";
        cout << "3. Issue Book\n";
        cout << "4. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                viewBooks();
                break;
            case 2:
                addBook();
                break;
            case 3: {
                int index;
                cout << "Enter book index to issue: ";
                cin >> index;
                issueBook(index);
                break;
            }
            case 4:
                cout << "Exiting...\n";
                break;
            default:
                cout << "Invalid choice.\n";
```

## Functionality

-Initializes with sample data via preloadBooks()

-Simple text-based menu system

-Uses do-while loop to keep running until exit

-Switch-case handles user choices

-Minimal input validation for menu choices

-Linear flow with no submenus