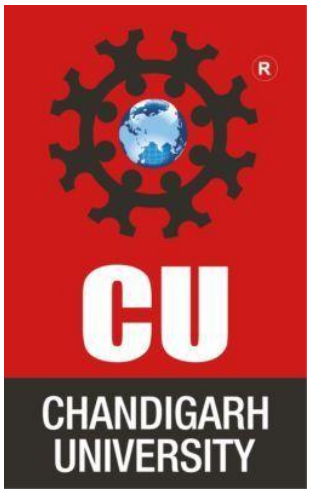
CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

NAAC GRADE A+
Accredited University




# Mini Project Report On

# Library Management System

**SUBMITTED BY:**

Deepak Kumar

UID: 25MCA20177

CLASS: MCA

SECTION: 25MCA-3(B)

**SUBMITTED TO:**

Dr. Sarabjeet Kaur

Assistant Professor

## Table of Content

# Acknowledgement

We would like to express our deepest gratitude to our respected teacher and project guide, **[Dr. Sarabjeet Kaur]**, for their constant support, valuable guidance, and encouragement throughout the development of this project. Their expertise and suggestions have been instrumental in shaping our understanding of programming concepts and software design.

We also take this opportunity to thank our Head of Department, our institution, and the management for providing the necessary facilities and environment to complete this project successfully.

We are sincerely grateful to our classmates and friends for their continuous help, motivation, and constructive feedback during the entire process. Lastly, we extend heartfelt thanks to our parents and family members for their unwavering support and patience, without which this project would not have been possible.

This project has given us an opportunity to explore and apply the principles of Object-Oriented Programming in C++ to solve a real-world problem — building a system that efficiently manages a library's books and user transactions.

# 1.Introduction

Libraries are vital centers of knowledge, serving as a bridge between readers and information. Traditionally, library operations such as book issuance, returns, recordkeeping, and catalog management have been handled manually, leading to inefficiencies, human errors, and time delays.

To address these challenges, we have developed a Library Management System (LMS) using **C++**, which simplifies and automates the day-to-day tasks of a library. The system is designed to maintain records of books, members, and transactions in an organized way, ensuring fast and reliable operations.

By leveraging Object-Oriented Programming (OOP) principles such as classes, inheritance, encapsulation, and file handling, this system allows the library to store and retrieve book data efficiently. The use of file handling in C++ provides data persistence, ensuring that all records remain saved even after the program is closed.

The project provides a simple menu-driven console interface, making it user-friendly even for non-technical users. This project aims to bridge the gap between manual record-keeping and digital library systems by implementing a small-scale automated solution.

# 2.Objective

The primary objectives of this project are as follows:

1. **Automation:** To automate all manual and time-consuming library management activities such as adding, searching, issuing, and returning books. Automation reduces paperwork, minimizes errors, and enhances productivity by performing repetitive operations quickly.

2. **Efficiency:** To improve the overall efficiency of the library system by making data access faster, eliminating redundant entries, and simplifying complex library tasks through a computerized process.

3. **Accuracy:** To ensure that all book and member records are maintained with high accuracy and consistency, minimizing the chances of data loss or duplication that are common in manual record-keeping systems.

4. **Data Storage:** To use file handling in C++ for secure and permanent data storage, allowing books and user records to be saved and retrieved efficiently even after the program is closed or restarted.

5. **User Interface:** To design a simple, intuitive, and menu-driven console interface that allows even non-technical users, such as librarians or students, to operate the system without prior programming knowledge.

6. **Time Management:** To significantly reduce the time required for performing various library operations like searching for a book, issuing, or returning, thereby improving overall time efficiency.

7. **Learning Goal:** To gain hands-on experience with real-world programming using C++ and Object-Oriented Design.

# 3.Technologies Used

The development of this project is based on C++, a powerful and widely-used programming language known for its speed, efficiency, and object-oriented capabilities.

| **Component** | **Technology/Tool Used** |
|---|---|
| Programming Language | C++ |
| Concepts Applied | Classes, Objects, Inheritance, File Handling, Encapsulation |
| Compiler/IDE | Turbo C++, Dev C++, or Code::Blocks |
| Operating System | Windows / Linux |
| Storage | Text File (library.txt) for persistent data storage |
| User Interface | Console-based (Command Line Interface) |

**Justification for Using C++:**

C++ provides high performance and strong support for object-oriented programming. Its file handling capabilities allow easy storage and retrieval of structured data, making it ideal for developing small-scale management systems like LMS.

# 4.Requirements and System Design

## A. System Requirements

### Hardware Requirements:

- **Processor:** Intel Core i3 or higher
- **RAM:** 2 GB minimum
- **Hard Disk:** 50 MB free space
- **Display:** Standard VGA or higher

### Software Requirements:

- **Operating System:** Windows / Linux
- **Programming Environment:** Turbo C++, Code::Blocks, or Dev C++
- **Compiler:** GNU GCC Compiler

## B. System Design

The system design follows a modular and object-oriented approach. It consists of multiple classes that handle different functionalities of the library, such as Book Management, Member Management, and Transaction Management.

Modules in the System:

### 1.Book Module:

- Handles book data such as title, author, and ID.
- Allows adding, searching, viewing, and deleting books.
-

**2.Member Module:**

- Maintains details of library members (students, teachers, etc.).
- Links members with book issue and return transactions.

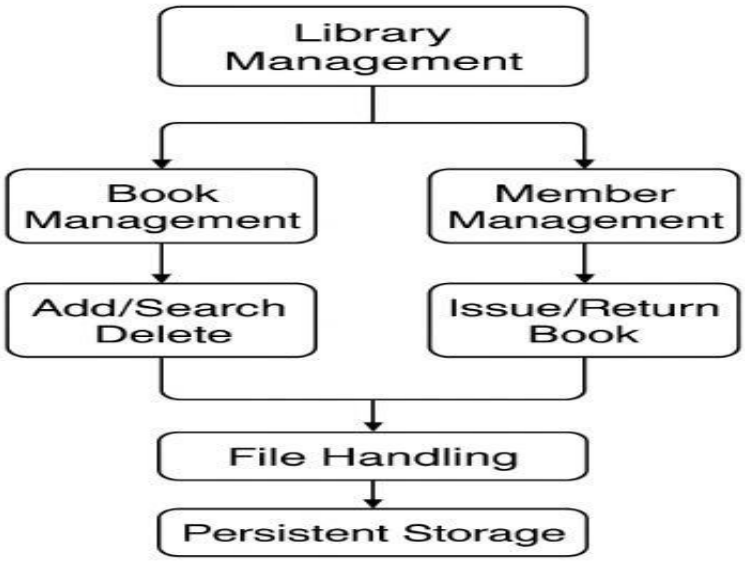
3. **Transaction Module:**

- Manages book issuing and returning.
- Updates availability status.

4. **File Handling Module:**

- Stores all records in files to ensure data persistence.
- Supports reading and writing data in binary mode for faster access.

**System Flow Diagram:**



## 5.Source Code:

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
class Book {
    int bookID;
    string title, author;

public:
    void getData() {
        cout << "\nEnter Book ID: ";
        cin >> bookID;
        cin.ignore();
        cout << "Enter Book Title: ";
        getline(cin, title);
        cout << "Enter Author Name: ";
        getline(cin, author);
    }
    void showData() const {
        cout << "\nBook ID: " << bookID;
```

```
        cout << "\nTitle: " << title;
        cout << "\nAuthor: " << author << endl;
    }
    int getBookID() const { return bookID; }
    string getTitle() const { return title; }
};

// Function to add a new book record
void addBook() {
    Book b;
    ofstream outFile("library.txt", ios::binary | ios::app);
    if (!outFile) {
        cout << "\nError opening file for writing!\n";
        return;
    }
    b.getData();
    outFile.write(reinterpret_cast<char*>(&b), sizeof(b));
    outFile.close();
    cout << "\nBook record added successfully!\n";
}
```

```
// Function to display all book records
void displayBooks() {
    Book b;
    ifstream inFile("library.txt", ios::binary);
    if (!inFile) {
        cout << "\nNo records found!\n";
        return;
    }
    cout << "\n===== All Book Records =====\n";
    while (inFile.read(reinterpret_cast<char*>(&b), sizeof(b))) {
        b.showData();
        cout << "----------------------------\n";
    }
    inFile.close();
}

// Function to delete a book record
void deleteBook() {
    int id;
    cout << "\nEnter Book ID to delete: ";
    cin >> id;
```

```
ifstream inFile("library.txt", ios::binary);
if (!inFile) {
    cout << "\nNo records found!\n";
    return;
}
ofstream outFile("temp.txt", ios::binary);
Book b;
bool found = false;
while (inFile.read(reinterpret_cast<char*>(&b), sizeof(b))) {
    if (b.getBookID() != id) {
        outFile.write(reinterpret_cast<char*>(&b), sizeof(b));
    } else {
        found = true;
    }
}
inFile.close();
outFile.close();
remove("library.txt");
rename("temp.txt", "library.txt");
if (found)
    cout << "\nBook record deleted successfully!\n";
```

```
    else
        cout << "\nBook with ID " << id << " not found!\n";
}

// New Function: Issue Book
void issueBook() {
    int id;
    string borrower;
    Book b;
    bool found = false;
    cout << "\nEnter Book ID to issue: ";
    cin >> id;
    cin.ignore();
    ifstream inFile("library.txt", ios::binary);
    if (!inFile) {
        cout << "\nNo book records found!\n";
        return;
    }
    ofstream issueFile("issued_books.txt", ios::app);
    if (!issueFile) {
        cout << "\nError opening issue file!\n";
```

```
    inFile.close();
    return;
}
while (inFile.read(reinterpret_cast<char*>(&b), sizeof(b))) {
    if (b.getBookID() == id) {
        found = true;
        cout << "\nBook Found!";
        b.showData();
        cout << "\nEnter Borrower's Name: ";
        getline(cin, borrower);
        issueFile << "Book ID: " << b.getBookID() << "\n";
        issueFile << "Title: " << b.getTitle() << "\n";
        issueFile << "Borrower: " << borrower << "\n";
        issueFile << "----------------------------\n";
        cout << "\nBook issued successfully to " << borrower << "!\n";
        break;
    }
}
if (!found)
    cout << "\nBook with ID " << id << " not found!\n";
inFile.close();
```

```
    issueFile.close();
}

// Function to display issued books
void displayIssuedBooks() {
    ifstream issueFile("issued_books.txt");
    if (!issueFile) {
        cout << "\nNo issued books found!\n";
        return;
    }
    cout << "\n===== Issued Book Records =====\n";
    string line;
    while (getline(issueFile, line)) {
        cout << line << endl;
    }
    issueFile.close();
}
int main() {
    int choice;
    do {
```

```
cout << "\n===== LIBRARY MANAGEMENT SYSTEM =====";
cout << "\n1. Add Book";
cout << "\n2. Display All Books";
cout << "\n3. Delete Book";
cout << "\n4. Issue Book";
cout << "\n5. Display Issued Books";
cout << "\n6. Exit";
cout << "\nEnter your choice: ";
cin >> choice;
switch (choice) {
    case 1:
        addBook();
        break;
    case 2:
        displayBooks();
        break;
    case 3:
        deleteBook();
        break;
    case 4:
        issueBook();
```

```
            break;
        case 5:
            displayIssuedBooks();
            break;
        case 6:
            cout << "\nThank you for using the system.\n";
            break;
        default:
            cout << "\nInvalid choice!\n";
    }
  } while (choice != 6);
  return 0;
}
```

# 6.Output

**Sample Console Output:**

```
===== LIBRARY MANAGEMENT SYSTEM =====
1. Add Book
2. Display All Books
3. Delete Book
4. Issue Book
5. Display Issued Books
6. Exit
Enter your choice: 1

Enter Book ID: 123
Enter Book Title: Science
Enter Author Name: MR.Rahul

Book record added successfully!

===== LIBRARY MANAGEMENT SYSTEM =====
1. Add Book
2. Display All Books
3. Delete Book
4. Issue Book
5. Display Issued Books
6. Exit
Enter your choice: 2

===== All Book Records =====

Book ID: 123
Title: Science
Author: MR.Rahul
-----------------------------
```

```
===== LIBRARY MANAGEMENT SYSTEM =====
1. Add Book
2. Display All Books
3. Delete Book
4. Issue Book
5. Display Issued Books
6. Exit
Enter your choice: 3

Enter Book ID to delete: 123

Book record deleted successfully!
free(): invalid pointer


...Program finished with exit code 134
Press ENTER to exit console.
```

## 7.Functionalities

- **Add Book:** Allows new books to be entered into the system.
- **Display Books:** Shows a list of all stored books.
- **Search Book:** Searches for books by ID or title (can be added as enhancement).
- **Issue Book:** Issues a book to a member (optional module).
- **Return Book:** Updates record when a book is returned.
- **Delete Book Record:** Removes outdated entries.
- **File Handling:** Stores all book data in text files permanently.

## 8.Conclusion and Future Work

The Library Management System in C++ is a practical and effective software solution that automates library operations. It reduces manual workload, minimizes human error, and ensures efficient handling of library data. The project successfully demonstrates the use of Object-Oriented Programming concepts and file management techniques in C++.

**Future Enhancements:**

1. Integration with graphical interface (GUI) using Qt or SFML.
2. Use of database systems (MySQL/SQLite) instead of text files for better scalability.
3. Implementation of user authentication (admin and librarian login).
4. Addition of fine calculation and due date tracking.
5. Real-time search and filter options for faster access.

## 9.Learning Outcomes:

Through the development of this project, we achieved the following learning outcomes:

- Acquired hands-on experience with C++ programming and OOP concepts.
- Learned the importance of modular programming and system design.
- Understood the implementation of file handling for real-world data storage.
- Improved problem-solving and debugging skills.
- Gained insights into software development lifecycle and documentation.
- Enhanced logical thinking and programming efficiency.

This project has been a valuable learning experience that strengthened both our theoretical and practical understanding of computer programming and system design.