**PROJECT TITLE:**

**LIBRARY MANAGEMENT SYSTEM**

**NAME: YATIN SAINI, TANISH ARYA , ABHISHEK YADAV**

**UID: 24BCA10103,10105,10128**

**CLASS: 24BCA2(B)**

**SUBMITTED TO: MR. ARVINDER SINGH**

CERTIFICATE

**Chandigarh University**

**Certificate of Completion**

This is to certify that **YATIN SAINI , TANISH ARYA , ABHISHEK**  
has successfully completed the project titled  
**"LIBRARY MANAGEMNT SYSTEM"**

**Submitted By:** YATIN SAINI, TANISH ARYA , ABHISHEK YADAV

**Supervised To:** Mr. ARVINDER SINGH

Congratulations on your accomplishment and dedication to excellence.

Signature: ­­­­­­­­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Acknowledgment**

I would like to express my sincere gratitude to **Chandigarh University** and the **Department of UIC** for providing me with the opportunity and resources to work on this project, titled **"LIBRARY MANAGEMENT SYSTEM”**

I am deeply grateful to my project supervisor, **Mr. Arvinder Singh** for their valuable guidance, encouragement, and insight throughout this project. Their expertise and mentorship have been crucial in completing this project successfully.

I would also like to thank the **lab staff and faculty members** who provided support and technical assistance. Their resources and timely help allowed me to work more efficiently and learn new skills.

Additionally, I extend my gratitude to my family and friends for their constant encouragement, patience, and motivation during the project’s duration.

Finally, I would like to acknowledge everyone who directly or indirectly contributed to the successful completion of this project.

Thank you all for your support and guidance.

Abstract

This project presents a **Library Management System (LMS)** implemented in **C** programming language, aimed at streamlining and simplifying the process of managing library resources. The system is designed to assist librarians in organizing, cataloging, and maintaining records for books and library users. The LMS supports core functionalities such as **adding new books**, **deleting or updating book records**, **searching for books by title or author**, and **issuing books to users**. Additionally, it provides features for **returning books** and managing fines for overdue returns.

By integrating a **structured database** within the C program, the system ensures efficient storage and retrieval of data. The use of simple **file handling techniques** enables persistent data storage, allowing the program to manage large amounts of information without the need for external databases. The user interface is text-based, providing an intuitive command-line experience for library staff and users alike.

This Library Management System enhances productivity by reducing manual record-keeping and minimizing errors, ensuring a reliable and effective method for tracking and managing library inventory. The implementation showcases modular programming techniques, allowing for future scalability and integration with other systems. The project demonstrates the potential of C for developing efficient management solutions and offers a foundation for further enhancements, such as networked library systems and multi-user access control.

Library management system in C :

Creating a Library Management System in C involves several components, including structuring the data, implementing the required functionalities (like adding, removing, searching, and displaying books), and handling user interactions. Here is a step-by-step approach to build a simple console-based Library Management System.

**Overview of the Library Management System**

The Library Management System will have these primary functionalities:

1. Add a new book
2. Search for a book by title, author, or ID
3. Display all books in the library
4. Remove a book by ID
5. Exit the program

The program will use a file to store the book information so that data persists even after the program terminates.

**Structure of the Program**

1. **Data structure**: Define a structure to hold book information.
2. **File handling**: Use file operations to save, retrieve, and delete book records.
3. **Menu-based system**: Provide a user-friendly menu for easy navigation.
4. **Functions**: Implement functions for each operation.

**Step 1: Define the Book Structure**

A book can have attributes like:

* ID: A unique identifier for each book
* Title: Name of the book
* Author: Name of the author
* Quantity: Number of copies available

Here is the structure definition in C:



**Step 2: Function Prototypes**

List down the required functions:

* void addBook()
* void displayBooks()
* void searchBook()
* void removeBook()

**Step 3: Implement Each Function**

**1. Function to Add a Book**

This function prompts the user to input book details and writes them to a file.

A computer screen shot of text

Description automatically generated

**2. Function to Display All Books**

This function reads the books from the file and displays them.

A computer code with many colorful text

Description automatically generated with medium confidence

**3. Function to Search for a Book**

This function allows searching by book title. Other search criteria (like ID or author) could be added similarly.

A screenshot of a computer program

Description automatically generated

**4. Function to Remove a Book**

This function removes a book based on its ID by copying all books except the one to be removed to a new file, then replacing the original file

A screenshot of a computer program

Description automatically generated

**Step 4: Menu and Main Function**

The main function provides a menu-driven interface for the user.

A screenshot of a computer program

Description automatically generated

**Case Studies and Explanation**

**Case Study 1: Adding a Book**

**Scenario: A librarian wants to add a new book to the library’s collection.**

* **Input:**
  + **Book ID: 101**
  + **Title: "C Programming"**
  + **Author: "Dennis Ritchie"**
  + **Quantity: 5**
* **Process:**

**1.The librarian selects the option to add a book from the menu.**

**2.Inputs the details as prompted.**

**3. The system validates the ID to ensure no duplicate entry exists.**

* **Expected Outcome:**
  + **The book details are written to the file library.dat, and a confirmation message is displayed: “Book added successfully.”**
  + **The system confirms the book was stored correctly by retrieving the data immediately afterward.**
* **Issues Encountered:**
  + **The librarian accidentally enters a duplicate ID. The system should ideally check for existing IDs and prompt the user to enter a unique ID.**
* **Lessons Learned:**
  + **Implement input validation to enhance user experience and prevent errors.**
  + **Ensure the system gracefully handles exceptions, such as when a file cannot be accessed.**

**Case Study 2: Searching for a Book**

**Scenario: A user wants to find a specific book by its title.**

* **Input: Title: "C Programming"**
* **Process:**
  1. **The user selects the search option.**
  2. **Inputs the title of the book.**
  3. **The system reads through the library.dat file.**
* **Expected Outcome:**
  1. **The program finds and displays the book details: ID, Title, Author, Quantity.**
  2. **If found, a message confirms: “Book Found: ID: 101, Title: C Programming, Author: Dennis Ritchie, Quantity: 5.”**
* **Issues Encountered:**
  1. **The title entered by the user is case-sensitive. The user enters “c programming” instead of “C Programming”.**
  2. **The system doesn’t find the book, leading to user frustration.**
* **Lessons Learned:**
  1. **Implement case-insensitive search functionality to improve user experience.**
  2. **Include search suggestions or fuzzy search algorithms for better results.**

**Case Study 3: Removing a Book**

**Scenario: The librarian needs to remove a book that is no longer in circulation.**

* **Input: Book ID: 101**
* **Process:**
  1. **The librarian selects the remove option.**
  2. **Inputs the book ID to be removed.**
  3. **The system checks for the ID and copies all other books to a temporary file, excluding the specified ID.**
* **Expected Outcome:**
  1. **The system confirms the book is removed and updates the file accordingly.**
  2. **A message is displayed: “Book removed successfully.”**
* **Issues Encountered:**
  1. **The librarian mistakenly enters an ID that does not exist, leading to a message: “Book with ID 101 not found.”**
  2. **The system could be improved to show the current list of available books before removal.**
* **Lessons Learned:**
  1. **Providing a preview of books before removal can prevent mistakes.**
  2. **Enhance error messages to suggest valid IDs or actions based on the current library inventory.**

**Case Study 4: Displaying All Books**

**Scenario: A user wants to view the complete list of books available in the library.**

* **Process:**
  1. **The user selects the display option.**
  2. **The system reads from the library.dat file and displays all records.**
* **Expected Outcome:**
  1. **The program outputs a neatly formatted list of all books, showing ID, Title, Author, and Quantity.**
* **Issues Encountered:**
  1. **If the file is empty or does not exist, the system returns an error message: “No books available.”**
* **Lessons Learned:**
  1. **A user-friendly message should provide guidance on how to add books if the list is empty.**
  2. **Ensure formatting is consistent and easy to read, especially if the list becomes long.**

**Explanation of Key Concepts**

* **File Handling**: Reading from and writing to a binary file ensures persistence.
* **Structure Usage**: Structs provide an efficient way to group data.
* **Menu-driven Interface**: Provides an intuitive way for users to interact with the system.

This code can be compiled and run in a C environment to create a basic, functional library management system.

**Summary of Enhancements**

By adding specific scenarios, detailed processes, expected outcomes, potential issues, and lessons learned for each case, the case studies now provide a more comprehensive understanding of the Library Management System's functionality. This approach not only emphasizes the capabilities of the system but also highlights the importance of user experience, error handling, and data integrity.