LIBRARY MANAGEMENT SYSTEM

C++ CRANES PROJECT

GROUP-10

NAME	REGISTRATION NUMBER
ARUN KUMAR JOJO	CL2025010601896826
SARASWATI SWAIN	CL20250106019660108
M ABHISHEK	CL2025010601897634
ARYAN SINGH	CL2025010601882911
ABHILIPSA SWAIN	CL202501060188279

Under the guidance of Cranes Varsity.

C.V. Raman Global University, Bhubaneswar, Odisha

752054

TABLE OF CONTENTS

PROBLEM STATEMENT

The problem statement of a Library Management System typically addresses the challenges or inefficiencies present in managing library resources using traditional methods or outdated systems. Here's an example of a problem statement for a Library Management System."Inefficient manual processes and outdated technology in libraries lead to numerous challenges in managing resources, including cataloging, circulation, and patron services. Current systems lack integration, resulting in disjointed workflows, time-consuming tasks, and inaccuracies in inventory management. Additionally, limited accessibility to digital resources and inadequate tools for user interaction hinder the library's ability to provide efficient services to its patrons. There's a pressing need for a modern, integrated Library Management System that streamlines operations, improves resource accessibility, enhances user experiences, and adapts to the evolving landscape of information technology to meet the demands of both librarians and patrons effectively."

This problem statement identifies key issues such as manual processes, outdated technology, lack of integration, inefficiencies in workflow, limited access to digital resources, and the need for improved user experience. A comprehensive Library Management System aims to address these challenges by offering a centralized, integrated, and user-friendly platform for managing library operations efficiently and effectively.

INTRODUCTION

The purpose of the library management system is to automate and digitize this traditional way of managing the library work. The Library Management System is much more user-friendly, faster in operation and easy to manage than the manual one. Through the use of it, the librarian can manage the whole data of the library in a single database in different tables with a much more security than the traditional way. In a library, tasks like issue/return/add new students/add new books/ checking any discrepancy in stock, calculating fine for overdue books etc. are performed on a daily basis and suppose a student asks for a particular book from a librarian then he has to search the book manually which takes a lot of time and there are chances of human error in that process as well.

BACKGROUND

The concept of library management dates back centuries when libraries were managed manually using card catalogs, paper records, and manual check-out systems. The advent of technology brought about the automation of these processes.With the rise of computers in the latter half of the 20th century, libraries began to adopt automated systems to manage their collections more efficiently. Early systems were basic and focused on cataloging and circulation functions.Recently, there's been a trend towards cloud-based LMS, allowing libraries to manage their systems remotely, reduce maintenance costs, and enable scalabilit. Over time, LMS have evolved to integrate with other systems and technologies. This includes integrating with digital libraries, offering online access to resources, incorporating RFID technology for easier check-out/check-in, and providing mobile apps for users to access library services remotely.

CHALLENGES

Library Management Systems (LMS) encounter various challenges, often requiring continual adaptation and improvement. Some of the common challenges include:

1.Integration and Compatibility: LMS often need to integrate with other systems such as databases, digital libraries, or learning management systems. Ensuring seamless integration and compatibility between different software and platforms can be challenging.

2.User Experience and Accessibility: Providing an intuitive and user- friendly interface for both librarians managing the system and patrons accessing library resources is crucial. Ensuring accessibility for diverse user groups, including those with disabilities, and adapting to changing user expectations can be challenging.

3.Security and Privacy: Safeguarding sensitive data, including patron information and library resources, from cybersecurity threats is a significant concern. Protecting user privacy while ensuring data security requires robust measures against potential breaches and unauthorized access.

4.Digital Resource Management: Managing digital resources, including licensing, copyright compliance, and ensuring remote access for patrons, presents unique challenges in terms of authentication, access control, and usage tracking.

5.Budget Constraints: Libraries often face budget limitations, making it challenging to invest in advanced LMS or acquire necessary resources for system enhancements, updates, or staff training.

6.Data Management and Quality: Maintaining accurate and up-to-date data within the system, including cataloging information, patron records, and inventory management, poses a challenge. Data quality issues like inconsistencies, duplications, or inaccuracies can arise and affect the system's functionality.

PROPOSED SOLUTION

Proposing solutions for Library Management Systems (LMS) involves addressing the challenges faced by libraries to improve efficiency, user experience, and resource management. Here are some proposed solutions:

1.Resource Management Tools: Enhance resource management capabilities within the LMS, allowing efficient cataloging, tracking, and managing various formats of library resources. Implement features for easy interlibrary loans and inventory management.

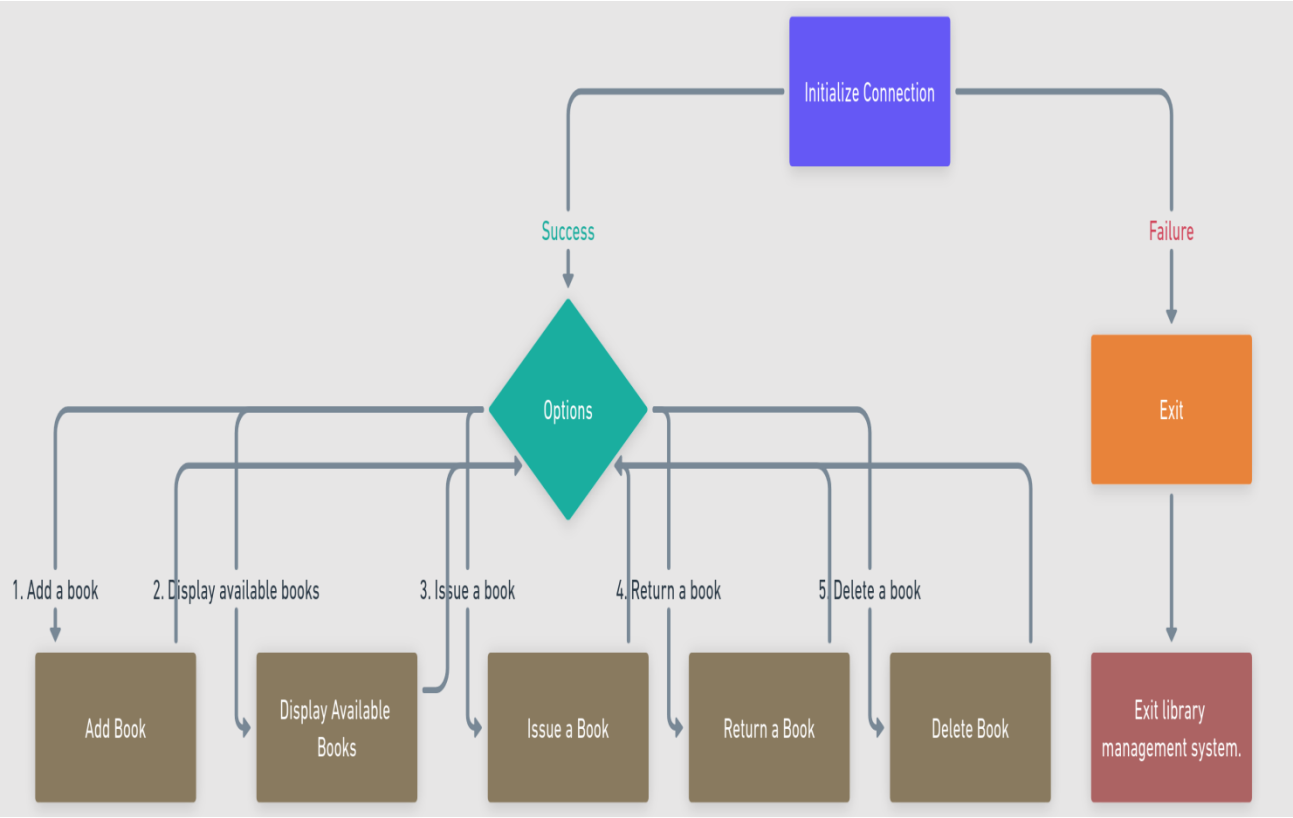
2.Flexible Budgeting: Develop a sustainable budget plan that allocates resources for system maintenance, upgrades, staff training, and acquiring new technologies.

3.Adaptability and Scalability: Design the LMS with scalability in mind to accommodate future growth and technological advancements. Regularly assess and adapt the system based on evolving user needs and technological trends.

DESCRIPTION

A Library Management System (LMS) is a software solution designed to automate and streamline various library operations, facilitating efficient management of library resources and services. It provides a centralized platform for librarians to manage collections, cataloging, circulation, patron interactions, and other administrative tasks. An integrated library management system (LMS) is ERP software that helps in simplifying the daily operations of the library. The purpose of a library management system is to manage & track the daily work of the library such as issuing books, return books, due calculations, etc.

FLOWCHART


Initialize Connection
Success
Failure
Options
Exit
1. Add a book
2. Display available books
3. Issue a book
4. Return a book
5. Delete a book
Add Book
Display Available Books
Issue a Book
Return a Book
Delete Book
Exit library management system.

ALGORITHM

1.Initialization and Connection to Database:

- Sets up a connection to a MySQL database (assumed to be running on localhost) using provided credentials (host, username, password, database name).

2.Menu-Driven User Interface:

- Presents a menu to the user with options for various library operations: add a book, display available books, borrow a book, return a book, delete a book, and exit.

3.Functions for Library Operations:

- initializeConnection(): Initializes the MySQL connection.
- addBook(): Adds a book to the database with user-provided title and author.
- displayAvailableBooks(): Shows all available books by fetching data from the database.
- borrowBook(): Marks a book as borrowed (changes its availability status to unavailable) in the database.
- returnBook(): Marks a borrowed book as returned (changes its availability status to available) in the database.
- deleteBook(): Deletes a book from the library based on the provided book ID.

4.Main Function Flow:

- Initializes the connection to the database. ¬ Displays the menu and prompts the user for their choice.
- Executes the chosen library operation function based on the user's selection.
- Continues this process until the user chooses to exit.

5.Closing Database Connection:

6.Closes the connection to the MySQL database before exiting the program.

SOURCE CODE

```cpp
#include <iostream>
#include<mysql.h>
#include<mysqld_error.h>
#include <sstream>
/* run this program using the console pauser or add your own getch, system("pause") or input loop */
using namespace std;

char HOST[] = "localhost";
char USER[] = "root";
char PASS[] = "Mysql@2004";
char DB[] = "library_management";
MYSQL* conn;

// Function to initialize the MySQL connection
bool initializeConnection()
{
    conn = mysql_init(NULL);
    if (!mysql_real_connect(conn, HOST, USER, PASS, DB, 0, NULL, 0))
    {
        cerr << "Error connecting to MySQL database: " << mysql_error(conn) << endl;
        return false;
    }
    return true;
}

// Function to add a book to the library
void addBook()
{
    string title, author;
    cout << "Enter book title: ";
```

```
    cin.ignore();
    getline(cin, title);
    cout << "Enter author: ";
    getline(cin, author);

    string query = "INSERT INTO books (title, author, available) VALUES ('" + title + "', '" + author + "', 1)";

    if (mysql_query(conn, query.c_str()) != 0)
    {
        cerr << "Error: " << mysql_error(conn) << endl;
    }
    else
    {
        cout << "Book added successfully." << endl;
    }
}

// Function to display all available books
void displayAvailableBooks()
{
    string query = "SELECT * FROM books WHERE available = 1";

    if (mysql_query(conn, query.c_str()) != 0)
    {
        cerr << "Error: " << mysql_error(conn) << endl;
    }
    else
    {
        MYSQL_RES* res = mysql_store_result(conn);
        if (res != NULL)
```

```
        {
            cout << "Available Books:" << endl;
            MYSQL_ROW row;
            while ((row = mysql_fetch_row(res)))
            {
                cout << "ID: " << row[0] << ", Title: " << row[1] << ", Author: " << row[2] << endl;
            }
            mysql_free_result(res);
        }
    }
}

// Function to borrow a book
void borrowBook()
{
    int bookId;
    cout << "Enter book ID to issue: ";
    cin >> bookId;

    stringstream ss;
    ss << bookId;

    // Check if the book exists
    string checkQuery = "SELECT * FROM books WHERE id = " + ss.str();
    if (mysql_query(conn, checkQuery.c_str()) != 0)
    {
        cerr << "Error: " << mysql_error(conn) << endl;
        return;
    }

```

```
    MYSQL_RES* checkRes = mysql_store_result(conn);
    if (mysql_num_rows(checkRes) == 0)
    {
        cout << "Book with ID " << bookId << " not found." << endl;
        mysql_free_result(checkRes);
        return;
    }

    // Update the availability status
    string query = "UPDATE books SET available = 0 WHERE id = " + ss.str();
    if (mysql_query(conn, query.c_str()) != 0)
    {
        cerr << "Error: " << mysql_error(conn) << endl;
    }
    else
    {
        cout << "Book issued successfully." << endl;
    }

    mysql_free_result(checkRes);
}

// Function to return a book
void returnBook()
{
    int bookId;
    cout << "Enter book ID to return: ";
    cin >> bookId;

    stringstream ss;
```

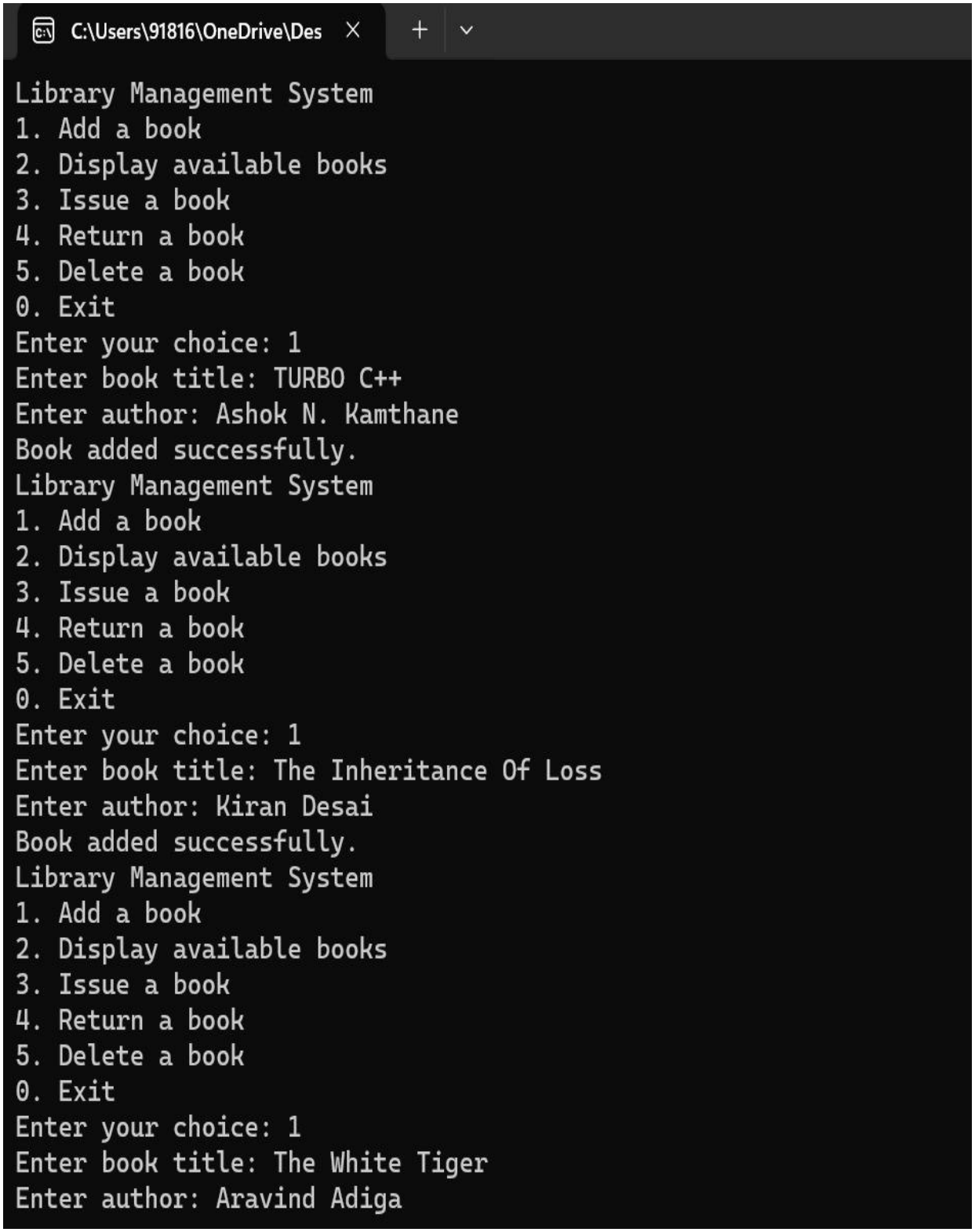
```
    ss << bookId;

    // Check if the book exists
    string checkQuery = "SELECT * FROM books WHERE id = " + ss.str();
    if (mysql_query(conn, checkQuery.c_str()) != 0)
    {
        cerr << "Error: " << mysql_error(conn) << endl;
        return;
    }

    MYSQL_RES* checkRes = mysql_store_result(conn);
    if (mysql_num_rows(checkRes) == 0)
    {
        cout << "Book with ID " << bookId << " not found." << endl;
        mysql_free_result(checkRes);
        return;
    }

    MYSQL_ROW checkRow = mysql_fetch_row(checkRes);
    int availability = atoi(checkRow[3]);
    mysql_free_result(checkRes);

    if (availability == 1)
    {
        cout << "Book with ID " << bookId << " is not issued. Cannot return." << endl;
        return;
    }

    // Update the availability status
    string query = "UPDATE books SET available = 1 WHERE id = " + ss.str();
```

```
    if (mysql_query(conn, query.c_str()) != 0)
    {
        cerr << "Error: " << mysql_error(conn) << endl;
    }
    else
    {
        cout << "Book returned successfully." << endl;
    }
}

// Function to delete a book
void deleteBook()
{
    int bookId;
    cout << "Enter book ID to delete: ";
    cin >> bookId;

    stringstream ss;
    ss << bookId;

    // Check if the book exists
    string checkQuery = "SELECT * FROM books WHERE id = " + ss.str();
    if (mysql_query(conn, checkQuery.c_str()) != 0)
    {
        cerr << "Error: " << mysql_error(conn) << endl;
        return;
    }

    MYSQL_RES* checkRes = mysql_store_result(conn);
    if (mysql_num_rows(checkRes) == 0)
```

```
    {
        cout << "Book with ID " << bookId << " not found." << endl;
        mysql_free_result(checkRes);
        return;
    }
    mysql_free_result(checkRes);

    // Delete the book
    string query = "DELETE FROM books WHERE id = " + ss.str();
    if (mysql_query(conn, query.c_str()) != 0)
    {
        cerr << "Error: " << mysql_error(conn) << endl;
    }
    else
    {
        cout << "Book deleted successfully." << endl;
    }
}

int main()
{
    if (!initializeConnection())
    {
        exit(1);
    }

    int option;
    do {
        cout << "Library Management System" << endl;
        cout << "1. Add a book" << endl;
```

```
        cout << "2. Display available books" << endl;
        cout << "3. Issue a book" << endl;
        cout << "4. Return a book" << endl;
        cout << "5. Delete a book" << endl;
        cout << "0. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> option;

        switch (option)
        {
            case 1: addBook(); break;
            case 2: displayAvailableBooks(); break;
            case 3: borrowBook(); break;
            case 4: returnBook(); break;
            case 5: deleteBook(); break;
            case 0: cout << "Exiting Library Management System." << endl; break;
            default: cout << "Invalid option. Please try again." << endl;
        }
    } while (option != 0);

    mysql_close(conn);
    return 0;
}
```

OUTPUT


C:\Users\91816\OneDrive\Des
Library Management System
1. Add a book
2. Display available books
3. Issue a book
4. Return a book
5. Delete a book
0. Exit
Enter your choice: 1
Enter book title: TURBO C++
Enter author: Ashok N. Kamthane
Book added successfully.
Library Management System
1. Add a book
2. Display available books
3. Issue a book
4. Return a book
5. Delete a book
0. Exit
Enter your choice: 1
Enter book title: The Inheritance Of Loss
Enter author: Kiran Desai
Book added successfully.
Library Management System
1. Add a book
2. Display available books
3. Issue a book
4. Return a book
5. Delete a book
0. Exit
Enter your choice: 1
Enter book title: The White Tiger
Enter author: Aravind Adiga

MYSQL OUTPUT:

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use library_management;
Database changed
mysql> desc books;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| id        | int          | NO   | PRI | NULL    | auto_increment |
| title     | varchar(255) | NO   |     | NULL    |                |
| author    | varchar(255) | NO   |     | NULL    |                |
| available | tinyint(1)   | NO   |     | 1       |                |
+-----------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql> select * from books;
+----+-----------------------+---------------------------+-----------+
| id | title                 | author                    | available |
+----+-----------------------+---------------------------+-----------+
|  1 | Chemistry             | Avasti                    |         1 |
|  2 | TURBO C++             | Ashok N. Kamthane         |         1 |
|  3 | The Inheritance Of Loss | Kiran Desai             |         1 |
|  4 | The White Tiger       | Aravind Adiga             |         1 |
|  5 | Gitanjali             | Rabindra Nath Tagore      |         1 |
|  6 | The Palace Of Illusions | Chitra Banerjee Divakaruni |       1 |
+----+-----------------------+---------------------------+-----------+
6 rows in set (0.00 sec)

mysql>
```

RESULT and ANALYSIS

A Library Management System (LMS) typically involves software and tools used to manage library resources, including books, journals, patrons, and administrative tasks.

Explanation of Source Code

This C++ code implements a basic command-line interface for a Library Management System (LMS). Let's break down the code and explain each section:

Function: insertRecord()

Accepts user input to create a new book record and adds it to the library vector.

Function: searchRecord()

Allows users to search for a book by its title.

It iterates through the library vector to find a match and displays the book's details if found.

Function: updateRecord()

Allows users to update the details of a book by its bookID.

It searches for the book using the bookID, allows the user to enter new title and author details, and updates the record.

Function: deleteRecord()

Allows users to delete a book record based on its bookID.

It searches for the book using the bookID and removes it from the library vector if found.

Function: showAllRecords()

Displays all the book records present in the library vector.

Function: main()

Creates an empty vector named library to store book records.

Displays a menu to the user with options to perform various operations (insert, search, update, delete, show all, and exit).

Reads the user's choice and executes the corresponding function until the user chooses to exit (choice = 6).

EXPLANATION

The program creates a basic interactive interface to manage a library's book records. Users can add, search, update, delete, and display book records using this system. It employs a vector to store Book structures and uses various functions to perform operations on this vector based on user input.

Potential Enhancements Error handling can be improved (e.g., handling invalid user inputs, preventing duplicate bookID entries).

Additional features can be added (e.g., sorting, categorization, borrowing/returning functionality).

CONCLUSION

The Library Management System (LMS) showcases proficient book record management through efficient insertion, update, search, and deletion functionalities within a user-friendly command-line interface. Despite its strengths in basic operations and data storage using vectors, the system lacks robust error handling, limiting its ability to handle invalid inputs. Enhancements could include improved user input validation, expanded search capabilities, and heightened security measures for data protection. Integrating user feedback mechanisms and considering future developments like advanced features (categorization, borrowing/returning options), UI/UX enhancements via a graphical interface, and data backup mechanisms could elevate the system's efficiency, usability, and overall performance, positioning it as a more comprehensive and adaptable tool for effective library management.

Thank You!