

**MEHRAN UNIVERSITY**

**OF**

**ENGINEERING & TECHNOLOGY**

## DEPARTMENT

## OF

## SOFTWARE ENGINEERING

#### PROJECT TITLE : LIBRARY MANAGEMENT SYSTEM

#### ASSIGNED BY : SIR SAJJAD ALI

#### UMAIR AHMED

#### Roll No: 24SW045

**Documentation for Library Management System Code**

**1. Introduction**

This program is a simple library management system in C++ that allows the user to:

* Add details about books (such as Book ID, Book Name, Author Name, Student Name, Price, and Number of Pages).
* Display the details of all books entered so far.
* Exit the program.

The system can store information for up to 20 books. It uses an array of structures (library[]) to store the details.

**2. Program Structure**

The program is divided into several sections:

1. **Class Definition**: A library class to store book details.
2. **Main Function**: Contains the program logic, including input, display, and exit options.
3. **Menu and Options**: A simple menu for the user to interact with.

**3. Class Definition: library**

cpp

Copy code

class library {

public:

int id; // Book ID

string Bookname; // Name of the book

string Authorname; // Name of the author (who wrote the book)

string Studentname; // Name of the student borrowing the book

int price; // Price of the book

int pages; // Number of pages in the book

};

**Explanation:**

* The library class is used to define a book object. Each book has the following properties:
  + id: An integer that uniquely identifies the book.
  + Bookname: A string representing the title of the book.
  + Authorname: A string representing the author of the book.
  + Studentname: A string representing the name of the student who borrows the book.
  + price: An integer indicating the price of the book.
  + pages: An integer that represents the number of pages in the book.

**4. Main Function: int main()**

**4.1 Declaring Variables:**

cpp

Copy code

library lib[20]; // Array to store up to 20 books

int input = 0; // Stores user input to select the action (1, 2, or 3)

int count = 0; // Keeps track of how many books have been entered

* lib[20] is an array of library objects that will store the details of up to 20 books.
* input holds the user's choice from the menu.
* count keeps track of how many books have been entered.

**4.2 While Loop to Keep Program Running:**

cpp

Copy code

while(input != 3) { // Loop continues until input is 3 (to quit)

* The while loop keeps the program running until the user enters 3 to exit.

**4.3 Display Menu:**

cpp

Copy code

cout << "Enter 1 to input details (id, Bookname, Authorname, Studentname, price, pages)" << endl;

cout << "Enter 2 to display details" << endl;

cout << "Enter 3 to quit" << endl;

cin >> input; // Read the user's choice

* This part displays the menu options (1, 2, or 3) to the user. The user's choice is stored in input.

**4.4 Switch Statement to Handle User Input:**

cpp

Copy code

switch(input) {

case 1: // Option to enter book details

break;

case 2: // Option to display all books' details

break;

case 3: // Exit the program

exit(0);

break;

default: // If the user enters an invalid option

cout << "Invalid choice. Try again!" << endl;

break;

}

* The switch statement checks the value of input and performs actions based on the user's choice:
  + **Case 1**: Calls the code to input book details.
  + **Case 2**: Displays the entered book details.
  + **Case 3**: Exits the program using exit(0).
  + **Default**: If the user enters an invalid number, the program will prompt the user to try again.

**5. Case 1: Inputting Book Details**

cpp

Copy code

case 1:

cout << "Enter Book ID: " << endl;

cin >> lib[count].id; // Store Book ID

cout << "Enter Book name: " << endl;

cin.ignore(); // Clear input buffer before reading a string

getline(cin, lib[count].Bookname); // Store Book Name

cout << "Enter Author name: " << endl;

getline(cin, lib[count].Authorname); // Store Author Name

cout << "Enter Student name: " << endl;

getline(cin, lib[count].Studentname); // Store Student Name

cout << "Enter Price: " << endl;

cin >> lib[count].price; // Store Price of Book

cout << "Enter Pages: " << endl;

cin >> lib[count].pages; // Store Number of Pages

count++; // Increment book count after entering the details

break;

**Explanation:**

* **Input Prompt**: The program asks the user to enter various details about the book one by one.
  + cin.ignore() is used to clear any leftover input in the buffer before reading strings like Book Name, Author Name, and Student Name. This ensures that the input is correctly read.
  + getline(cin, lib[count].Bookname) allows the user to enter the full name of the book, even if it has spaces (e.g., "Harry Potter").
* After all details are entered, count++ increments the book counter.

**6. Case 2: Displaying All Book Details**

cpp

Copy code

case 2:

for(int i = 0; i < count; i++) { // Loop to display all entered books

cout << "Book ID: " << lib[i].id << endl;

cout << "Book Name: " << lib[i].Bookname << endl;

cout << "Author Name: " << lib[i].Authorname << endl;

cout << "Student Name: " << lib[i].Studentname << endl;

cout << "Book Price: " << lib[i].price << endl;

cout << "Book Pages: " << lib[i].pages << endl;

}

break;

**Explanation:**

* **Displaying Details**: A loop runs from i = 0 to count-1, printing the details of each book entered so far.
* It prints out the ID, name, author, student who borrowed the book, price, and pages for each book stored in the array lib[].

**7. Case 3: Exiting the Program**

cpp

Copy code

case 3:

exit(0); // Exit the program

break;

* If the user chooses option 3, the exit(0) function is called, which stops the program.

**8. Error Handling (Default Case)**

cpp

Copy code

default:

cout << "Invalid choice. Try again!" << endl;

break;

* If the user enters an invalid option (something other than 1, 2, or 3), the program tells them to try again and waits for a valid input.

**9. Conclusion**

This program allows users to interact with a simple library system by:

1. Adding book details.
2. Displaying all entered book details.
3. Exiting the program.

By using an array of library objects, the program stores up to 20 books. It uses basic C++ input/output functions and control structures such as loops and switch cases to manage user interactions.

**10. Enhancements and Future Improvements**

* **Validation**: Add input validation to ensure that the user enters valid data (e.g., positive numbers for price and pages).
* **Dynamic Storage**: Instead of a fixed array size (20), use dynamic memory allocation to allow more flexible storage.
* **Sorting and Searching**: Implement sorting by Book ID or name, and searching for a specific book based on certain criteria (e.g., student name or book name).