**Project\_Overview:**

**Library Management System**

**By, Didhiti Raj Chakraborty**

**Project Overview: Library Management System**

This is a console-based application that allows users to perform basic library management functions like:

1. Add books to the library
2. View all books in the library
3. Search for a book by title
4. Delete a book from the library

This project demonstrates **basic C++ concepts** like:

* Classes and objects
* File handling (to store book records permanently)
* Loops, functions, and conditionals

## ****Step-by-Step Explanation****

**1. Project Structure**

You will create a single file library\_managementSystem.cpp for this project.

**2. Classes and Objects**

We’ll define a Book class to represent books with attributes:

* **ID**: A unique identifier for each book
* **Title**: The title of the book
* **Author**: The name of the book's author

**#include <iostream>**

**#include <fstream>**

**#include <vector>**

**#include <string>**

**using namespace std;**

**class Book {**

**public:**

**int id;**

**string title;**

**string author;**

**Book() {} // Default constructor**

**Book(int i, string t, string a) : id(i), title(t), author(a) {} // Parameterized constructor**

**};**

**3. File Handling**

Books will be saved to a file (books.txt) so that the data persists even after the program ends. We'll create **functions** for:

* Adding a book
* Viewing all books
* Searching a book by title
* Deleting a book

**4. Main Functions**

Let’s define each feature step by step.

**4.1 Add a Book**

This function:

1. Takes input for book ID, title, and author.
2. Saves this information to books.txt.

**void addBook() {**

**ofstream file("books.txt", ios::app); // Open file in append mode**

**int id;**

**string title, author;**

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

**cin >> id;**

**cin.ignore(); // To ignore newline character**

**cout << "Enter Book Title: ";**

**getline(cin, title);**

**cout << "Enter Author Name: ";**

**getline(cin, author);**

**file << id << "," << title << "," << author << endl; // Save to file**

**file.close();**

**cout << "Book added successfully!" << endl;**

**}**

**4.2 View All Book**

This function reads the file and displays all book records.

**void viewBooks() {**

**ifstream file("books.txt");**

**if (!file) {**

**cout << "No books found!" << endl;**

**return;**

**}**

**int id;**

**string title, author;**

**cout << "ID\tTitle\tAuthor" << endl;**

**while (file >> id) {**

**getline(file, title, ','); // Read title until comma**

**getline(file, author); // Read author**

**cout << id << "\t" << title << "\t" << author << endl;**

**}**

**file.close();**

**}**

**4.3 Search for a Book by Tittle**

This function:

1. Reads through the file line by line.
2. Checks if the book's title matches the search input.

**void searchBook() {**

**ifstream file("books.txt"); // Open the file for reading**

**if (!file) {**

**cout << "No books found!" << endl;**

**return;**

**}**

**string searchTitle, title, author;**

**int id;**

**bool found = false;**

**cout << "Enter Book Title to Search: ";**

**cin.ignore(); // Clear input buffer**

**getline(cin, searchTitle);**

**cout << "\nSearching for Book...\n";**

**while (file >> id) {**

**file.ignore(); // Ignore the comma**

**getline(file, title, ',');**

**getline(file, author);**

**if (title == searchTitle) {**

**cout << "\nBook Found!" << endl;**

**cout << "ID: " << id << endl;**

**cout << "Title: " << title << endl;**

**cout << "Author: " << author << endl;**

**found = true;**

**break;**

**}**

**}**

**if (!found) {**

**cout << "Book not found!" << endl;**

**}**

**file.close();**

**}**

**4.4 Delete a Book**

This function:

1. Copies all books except the one to delete into a temporary file.
2. Renames the temporary file back to books.txt.

**void deleteBook() {**

**ifstream file("books.txt");**

**ofstream temp("temp.txt");**

**int id, deleteId;**

**string title, author;**

**cout << "Enter Book ID to Delete: ";**

**cin >> deleteId;**

**bool found = false;**

**while (file >> id) {**

**getline(file, title, ',');**

**getline(file, author);**

**if (id == deleteId) {**

**found = true;**

**continue; // Skip the book to delete**

**}**

**temp << id << "," << title << "," << author << endl;**

**}**

**file.close();**

**temp.close();**

**remove("books.txt"); // Delete old file**

**rename("temp.txt", "books.txt"); // Rename temp file to books.txt**

**if (found)**

**cout << "Book deleted successfully!" << endl;**

**else**

**cout << "Book not found!" << endl;**

**}**

**5. Menu System**

A simple menu for users to choose options:

**int main() {**

**int choice;**

**while (true) {**

**cout << "\nLibrary Management System" << endl;**

**cout << "1. Add Book" << endl;**

**cout << "2. View Books" << endl;**

**cout << "3. Search Book" << endl;**

**cout << "4. Delete Book" << endl;**

**cout << "5. Exit" << endl;**

**cout << "Enter your choice: ";**

**cin >> choice;**

**switch (choice) {**

**case 1: addBook(); break;**

**case 2: viewBooks(); break;**

**case 3: searchBook(); break;**

**case 4: deleteBook(); break;**

**case 5: cout << "Exiting program. Goodbye!" << endl; return 0;**

**default: cout << "Invalid choice! Try again." << endl;**

**}**

**}**

**return 0;**

**}**

**6. Explanation of Code**

* **Classes and Objects**: The Book class is used to structure book details.
* **File Handling**: ofstream, ifstream, and file operations ensure data persistence.
* **Functions**: Each feature is encapsulated in its own function to maintain clarity and modularity.
* **Loops and Conditionals**: The while loop allows continuous interaction until the user exits.
* **User Input and Output**: Input validation and output formatting make the program user-friendly.

**7. How to Compile and Run**

1. Save the code as library\_managementSystem.cpp.
2. Open Code::Blocks IDE and open library\_managementSystem.cpp fill.
3. Click on “Build” then click on “Build on run”.