**Documentation About,**

**Library Management System**

**Author: Didhiti Raj Chakraborty**

**Table of Contents**

1. [Introduction](#introduction)
2. [Objective](#objective)
3. [Features](#features)
4. [Tools and Technologies Used](#tools-and-technologies-used)
5. [System Design and Code Walkthrough](#system-design-and-code-walkthrough)
   * 5.1 Class Diagram
   * 5.2 Functionality
   * [5.3 Class Definition](#51-class-definition)
   * [5.4 Add Book Functionality](#52-add-book-functionality)
   * [5.5 View Books Functionality](#53-view-books-functionality)
   * [5.6 Search Book Functionality](#54-search-book-functionality)
   * [5.7 Delete Book Functionality](#55-delete-book-functionality)
   * [5.8 Menu System](#56-menu-system)
6. [Project Execution](#project-execution)
7. [Testing](#testing)
8. [Future Enhancements](#future-enhancements)
9. [Conclusion](#conclusion)

**1. Introduction**

The **Library Management System** is a console-based C++ application designed to manage library books. It allows users to **add, view, search, and delete books**, demonstrating practical applications of **Object-Oriented Programming (OOP)** and **file handling** in C++.

## ****2. Objective****

The objective of this project is to:

* Implement a simple C++ program to manage book records efficiently.
* Demonstrate object-oriented programming principles.
* Practice file handling to save and retrieve data.

This project is suitable for beginners to intermediate-level developers who want to understand how to work with files, classes, and modular code in C++.

## ****3. Features****

The Library Management System includes the following features:

1. **Add Book**: Allows the user to add book details (ID, title, and author) to the library.
2. **View Books**: Displays a list of all books currently available in the library.
3. **Search Book**: Searches for a book in the library by title.
4. **Delete Book**: Removes a specific book from the library using its ID.
5. **Data Persistence**: Book records are saved in a file (books.txt) so that they are retained even after the program is closed.

## ****4. Tools and Technologies Used****

* **Programming Language**: C++
* **Compiler**: GCC (GNU Compiler Collection)
* **IDE/Editor**: Visual Studio Code, Code::Blocks, or any text editor
* **Operating System**: Windows/Linux/MacOS
* **File Handling**: Used to save and retrieve book records

## ****5.**** System Design and Code Walkthrough

## ****5.1 Class Diagram****

+----------------------------------+

| Book |

+----------------------------------+

| - id: int |

| - title: string |

| - author: string |

+----------------------------------+

| + Book() |

| + Book(int, string, string) |

+----------------------------------+

## ****5.2 Functionality****

| **Function** | **Purpose** | **Input** | **Output** |
| --- | --- | --- | --- |
| addBook() | Adds a new book to the library and stores it in a file. | Book ID, Title, Author | Confirmation of book addition |
| viewBooks() | Displays all books in the library. | None | List of books |
| searchBook() | Searches for a book by title. | Book Title | Details of the book (if found) |
| deleteBook() | Deletes a book from the library using its ID. | Book ID | Confirmation of book deletion |

## ****5.3 Class Definition****

The Book class represents the structure of a book record.

**#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**

**};**

**Explanation:**

* **Attributes**:
  + id - Unique identifier for the book.
  + title - Title of the book.
  + author - Author of the book.
* **Constructors**:
  + **Default Constructor** - Initializes an empty object.
  + **Parameterized Constructor** - Allows creating objects with values.

## ****5.4 Add Book Functionality****

The addBook function adds a new book to the system and saves it to a file.

**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;**

**}**

## ****Explanation:****

1. **File Handling**:Opens books.txt in **append mode** (ios::app)

to add new data without overwriting.

1. **User Input**: Takes ID, title, and author from the user.
2. **Saving to File**:Appends the book details as a single line

in the format: id,title,author.

## ****5.5**** View Books Functionality

The viewBooks function reads and displays all book records from the file.

**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();**

**}**

**Explanation:**

1. **File Reading**: Opens books.txt in **input mode**.
2. **Display**: Reads and prints each book record line-by-line.

## ****5.6**** Search Book Functionality

## The searchBook function allows searching for a book by its title.

**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();**

**}**

**Explanation:**

1. Searches for the user-specified title.
2. Prints book details if found; otherwise, displays "Book not found!".

## ****5.7**** Delete Book Functionality

## The deleteBook function removes a book by its ID.

**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;**

**}**

## ****5.8**** Menu System Or Main Function

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

**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;**

**}**

## ****Project Execution****

**Steps to Run the Project**

1. **Setup**
   * Install a C++ compiler (GCC).
   * Create a file library\_managementSystem.cpp
2. **Write Code**
   * Copy the code provided into the file.
3. **Compile the Code**
   * Use the terminal/command prompt to compile:

g++ library\_management.cpp -o library\_management

1. **Run the Program**
   * Execute the compiled file:

./library\_management

1. **Interaction**
   * Follow the menu options to add, view, search, or delete books.

## ****Testing****

| **Test Case** | **Input** | **Expected Output** | **Status** |
| --- | --- | --- | --- |
| Add a book | ID=1, Title=Book1, Author=Author1 | "Book added successfully!" | Pass |
| View all books | |  | | --- | |  |  |  | | --- | | N/A | | List with all book details | Pass |
| Search for an existing book | Title=Book1 | Book details are displayed | Pass |
| Search for a non-existing book | Title=Unknown | "Book not found!" | Pass |
| Delete a book | ID=1 | "Book deleted successfully!" | Pass |

**8. Future Enhancements**

* Add functionality to **update book details**.
* Implement a **GUI** for better user experience.

**9. Conclusion**

This project demonstrates key concepts of **C++ programming** and is a solid addition to your portfolio. It showcases **file handling**, **OOP**, and a user-friendly menu system.