**DSA ASSIGNMENT**

**On**

# “LIBRARY MANAGEMENT SYSTEM”

**Project Report**

Submitted by :

NAME : BHUMIREDDY SUDARSHAN REDDY

Email.ID : reddysudharshan060@gmail.com

**Project Overview*:***

This project involves building a simple library management system in C++. This system would allow the librarian to manage books and keep track of issued books. The librarian should be able to add new books, search for books, issue a book, and return a book.

The project will provide students an opportunity to apply knowledge of various data structures such as arrays, linked lists, trees, and more in a practical

scenario. It will also involve implementing essential algorithms for searching and sorting.

**Code:**

#include <iostream>

#include <string>

#include <vector> #include <queue>

#include <algorithm>

using namespace std;

struct Student {

string name; int id;

};

class Book { private: int id; string title; string author; bool isAvailable; Student\* borrower; public:

Book(int \_id, string \_title, string \_author)

: id(\_id), title(\_title), author(\_author), isAvailable(true), borrower(nullptr) {}

int getId() { return id; } string getTitle() { return title; } string getAuthor() { return author; } bool getAvailability() { return isAvailable; } void setAvailability(bool availability) { isAvailable = availability; } void setBorrower(Student\* student) { borrower = student; }

Student\* getBorrower() { return borrower; }

};

class Library { private:

vector<Book> books; int nextId;

queue<pair<Book\*, Student\*>> issuedBooks; public:

Library() : nextId(1) {}

void addBook(const string& title, const string& author) { books.push\_back(Book(nextId, title, author));

nextId++;

}

void searchBook() { string searchTerm; cout << "Enter book title or ID to search: "; cin.ignore();

getline(cin, searchTerm);

bool found = false;

try {

int searchId = stoi(searchTerm); for (size\_t i = 0; i < books.size(); i++) { if (books[i].getId() == searchId) { displayBookDetails(books[i]);

found = true;

break;

}

}

} catch (invalid\_argument&) { for (size\_t i = 0; i < books.size(); i++) { if (books[i].getTitle() == searchTerm) { displayBookDetails(books[i]);

found = true;

}

}

}

if (!found) {

cout << "Book not found." << endl;

}

}

void issueBook() { int bookId;

cout << "Enter the ID of the book to issue: "; cin >> bookId; for (size\_t i = 0; i < books.size(); i++) { if (books[i].getId() == bookId) { if (books[i].getAvailability()) { Student\* student = new Student(); cout << "Enter student name: "; cin.ignore();

getline(cin, student->name); cout << "Enter student ID: "; cin >> student->id; books[i].setAvailability(false); books[i].setBorrower(student);

issuedBooks.push(make\_pair(&books[i], student));

cout << "Book '" << books[i].getTitle() << "' has been issued to " << student->name << endl;

} else {

cout << "The book '" << books[i].getTitle() << "' is already issued." << endl;

} return;

}

}

cout << "Book with ID " << bookId << " not found." << endl;

}

void returnBook() {

int bookId;

cout << "Enter the ID of the book to return: "; cin >> bookId; for (size\_t i = 0; i < books.size(); i++) { if (books[i].getId() == bookId) { if (!books[i].getAvailability()) { books[i].setAvailability(true);

Student\* borrower = books[i].getBorrower();

books[i].setBorrower(nullptr); queue<pair<Book\*, Student\*>> temp; while (!issuedBooks.empty()) {

if (issuedBooks.front().first != &books[i]) { temp.push(issuedBooks.front());

}

issuedBooks.pop();

}

issuedBooks = temp;

cout << "Book '" << books[i].getTitle() << "' has been returned successfully." << endl; cout << "It was borrowed by " << borrower->name << " (ID: " << borrower->id << ")" << endl; delete borrower;

} else {

cout << "Error: The book '" << books[i].getTitle() << "' is not currently issued." << endl;

} return;

}

}

cout << "Error: Book with ID " << bookId << " not found." << endl;

}

void listAllBooks() { if (books.empty()) {

cout << "The library is empty." << endl; return;

}

int choice;

cout << "Sort by:\n1. ID\n2. Title\nEnter your choice (1 or 2): "; cin >> choice;

vector<Book> sortedBooks = books; if (choice == 1) {

sort(sortedBooks.begin(), sortedBooks.end(),

[](Book& a, Book& b) { return a.getId() < b.getId(); });

} else if (choice == 2) {

sort(sortedBooks.begin(), sortedBooks.end(),

[](Book& a, Book& b) { return a.getTitle() < b.getTitle(); });

} else {

cout << "Invalid choice. Sorting by ID by default." << endl; sort(sortedBooks.begin(), sortedBooks.end(),

[](Book& a, Book& b) { return a.getId() < b.getId(); });

}

cout << "\nList of all books in the library:\n" << endl; cout << "ID\tTitle\tAuthor\tStatus" << endl; for (size\_t i = 0; i < sortedBooks.size(); i++) { cout << sortedBooks[i].getId() << "\t"

<< sortedBooks[i].getTitle() << "\t"

<< sortedBooks[i].getAuthor() << "\t"

<< (sortedBooks[i].getAvailability() ? "Available" : "Issued") << endl;

}

}

void deleteBook() { int bookId;

cout << "Enter the ID of the book to delete: "; cin >> bookId; for (size\_t i = 0; i < books.size(); i++) { if (books[i].getId() == bookId) { if (!books[i].getAvailability()) {

cout << "Error: Cannot delete book '" << books[i].getTitle()

<< "' because it is currently issued." << endl; return;

}

string title = books[i].getTitle(); books.erase(books.begin() + i); queue<pair<Book\*, Student\*>> temp; while (!issuedBooks.empty()) {

if (issuedBooks.front().first->getId() != bookId) { temp.push(issuedBooks.front());

}

issuedBooks.pop();

}

issuedBooks = temp;

cout << "Book '" << title << "' with ID " << bookId

<< " has been successfully deleted from the library." << endl; return;

}

}

cout << "Error: Book with ID " << bookId << " not found." << endl;

}

private:

void displayBookDetails(Book& book) { cout << "Book Details:" << endl; cout << "ID: " << book.getId() << endl; cout << "Title: " << book.getTitle() << endl; cout << "Author: " << book.getAuthor() << endl;

cout << "Status: " << (book.getAvailability() ? "Available" : "Issued") << endl; if (!book.getAvailability() && book.getBorrower()) {

cout << "Borrowed by: " << book.getBorrower()->name << " (ID: " << book.getBorrower()->id << ")" << endl;

}

cout << endl;

}

};

int main() { Library library; // Adding some predefined books library.addBook("Midnight's Children", "Salman Rushdie"); library.addBook("The God of Small Things", "Arundhati Roy"); library.addBook("A Suitable Boy", "Vikram Seth");

library.addBook("The Sadhu", "Gotham Chopra and Jeevan J. Kang");

int choice;

while (true) {

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

cout << "1. Add a book" << endl; cout << "2. Search for a book" << endl; cout << "3. Issue a book" << endl; cout << "4. Return a book" << endl; cout << "5. List all books" << endl; cout << "6. Delete a book" << endl; cout << "7. Exit" << endl; cout << "Enter your choice: ";

cin >> choice; switch (choice) { case 1: { string title, author; cout << "Enter book title: ";

cin.ignore();

getline(cin, title); cout << "Enter book author: "; getline(cin, author); library.addBook(title, author);

break; } case 2: library.searchBook();

break; case 3: library.issueBook();

break; case 4: library.returnBook();

break; case 5: library.listAllBooks();

break; case 6: library.deleteBook();

break; case 7:

cout << "Thank you...Have a nice day!!!" << endl; return 0; default:

cout << "Invalid choice. Please try again." << endl;

}

}

return 0;

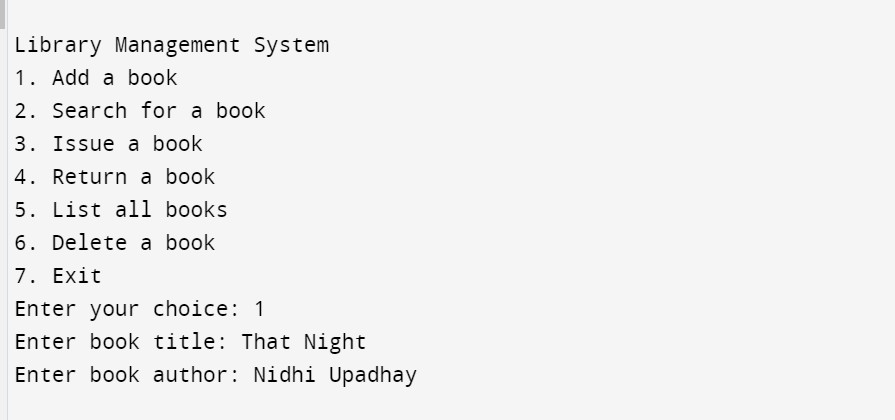
}

**Output:**

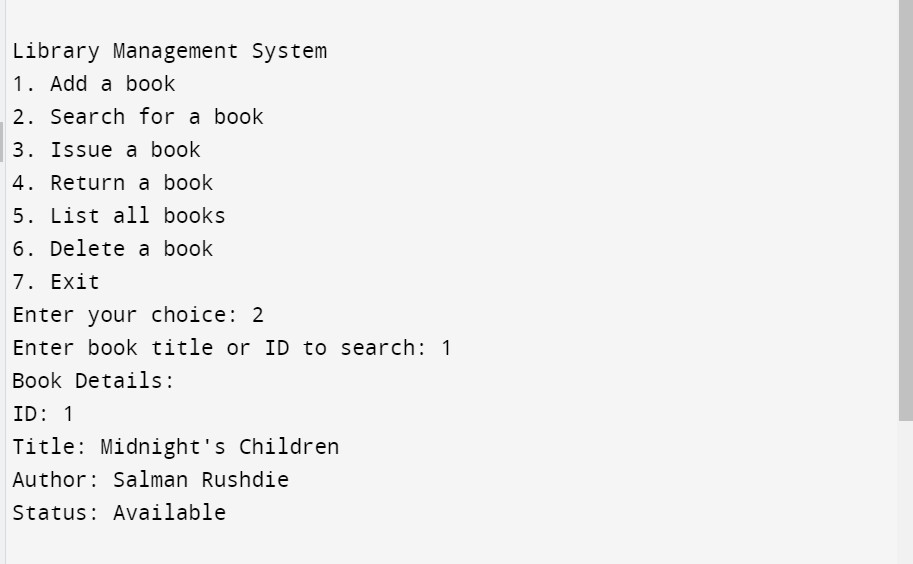
**For output complier used online c++ complier**

Website Link :[- https://www.programiz.com/cpp-programming/online-compiler/](https://www.programiz.com/cpp-programming/online-compiler/)

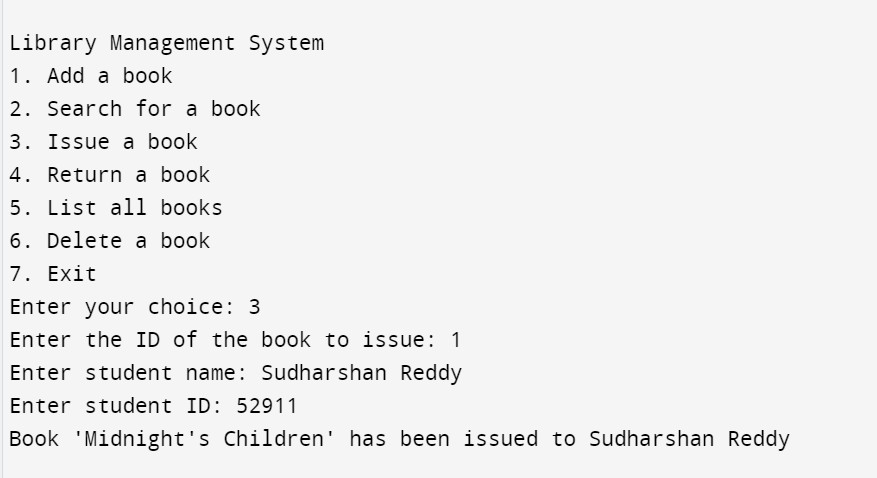
1. To Add Book :-



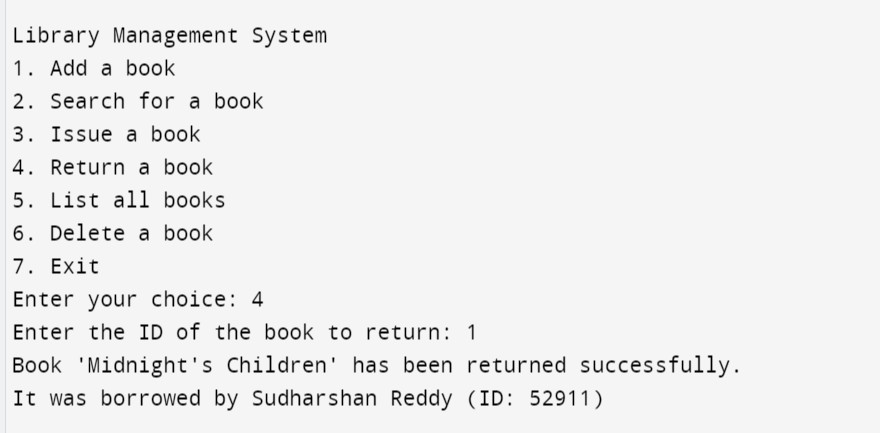
1. To Search for a Book :-



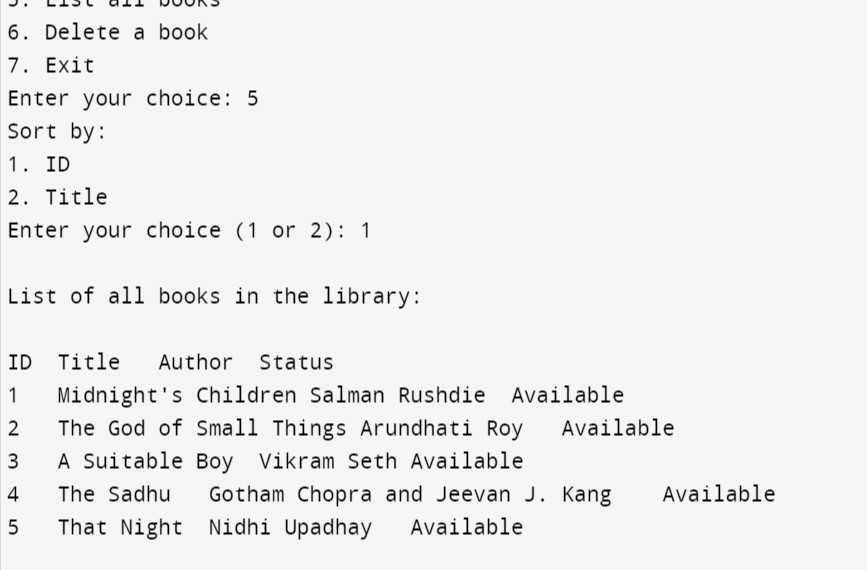
1. To Issue a Book :-



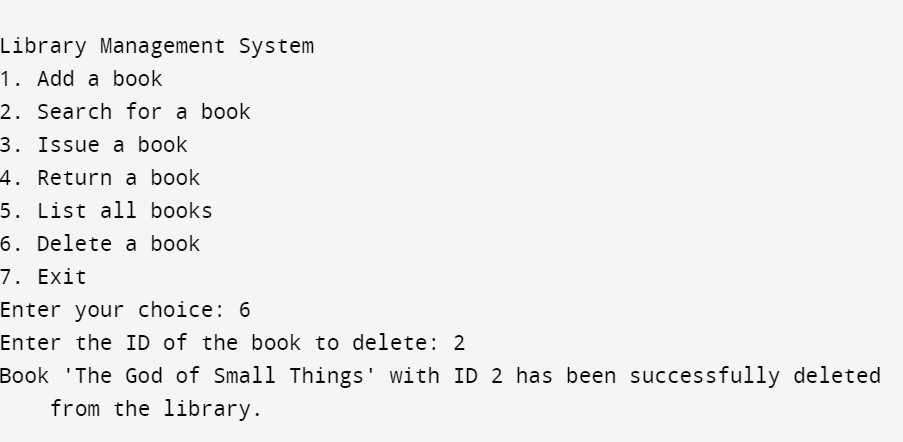
1. To Return a Book :-



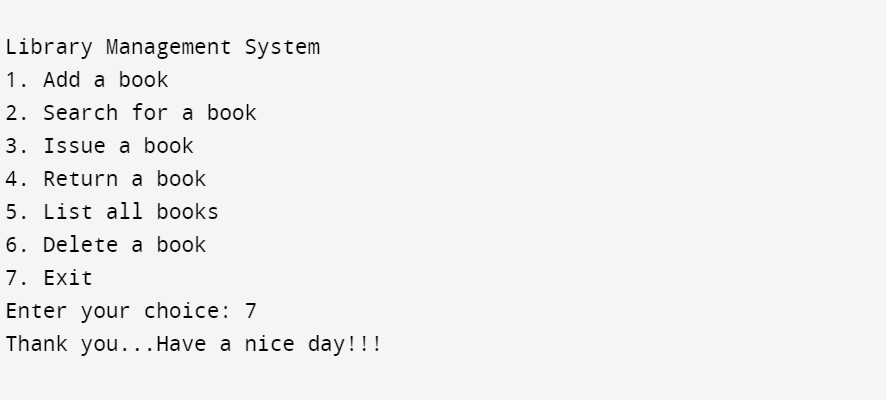
1. To List the Books :-



1. To Delete the book :-



1. To Exit the LMS :-



**Conclusion:**

This Library Management System helps you add, search, issue, return, list, and delete books easily. It tracks who borrowed which book and updates availability. You can see all books sorted by ID or title. The system reduces manual work and keeps the library organized. It's simple, efficient, and user-friendly.

# Thank You