**LIBRARY**

**MANAGEMENT SYSTEM**

**A PROJECT REPORT**

**Submitted By :- Shubham Gupta**

**Date :- 15-08-2024**

**TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **TOPIC** | **PAGE NO** |
| *1* | Introduction | 1 |
| *2* | Objective | 3 |
| *3* | Tools/Environment | 4 |
| *4* | Program Code | 7 |
| *5* | Input and Output Screens | 17 |
| *6* | Limitations of the Project | 23 |
| *7* | Future Applications of the Project | 24 |
| *8* | Bibliography | 25 |

**INTRODUCTION**

A Library Management System (LMS) is a comprehensive and dynamic software platform designed to facilitate and streamline the day-to-day operations of a library. Libraries, whether public, academic, or specialized, play a crucial role in the dissemination of knowledge and information, serving as repositories of countless books, journals, digital media, and other resources. Managing these resources efficiently is vital to ensure that users—whether they be students, researchers, or the general public—can access the information they need in a timely and organized manner.

**Library Management System ..**

The evolution of libraries from purely physical spaces to hybrid environments that also encompass digital resources has necessitated the development of sophisticated management systems. A Library Management System addresses this need by providing tools and functionalities that cover a wide array of tasks, from cataloging and acquisition to circulation and user management. It integrates various library functions into a single platform, reducing redundancy, improving accuracy, and enhancing the overall user experience.



**Key Features of the Library Management System**

1. **Cataloging and Classification**: The LMS enables libraries to organize their collections systematically. Each resource is cataloged with detailed metadata, including title, author, subject, and classification number, making it easy to locate specific items within the library’s collection. The system supports various classification schemes, such as Dewey Decimal Classification or Library of Congress Classification, ensuring compatibility with global standards.
2. **Acquisition Management**: Acquiring new materials is a critical function of any library. The LMS automates the acquisition process, allowing librarians to manage budgets, track orders, and maintain supplier records efficiently. The system can generate purchase orders, receive invoices, and update the catalog with new acquisitions in real-time.
3. **Circulation Management**: The circulation module is one of the most utilized features of an LMS. It manages the lending of materials, tracking which items are checked out, who has borrowed them, and when they are due for return. The system also handles renewals, reservations, and fines for overdue items, ensuring smooth and efficient operation of the lending process.
4. **User Management**: Libraries serve diverse user groups, each with unique needs and privileges. The LMS allows libraries to create and manage user profiles, assigning different levels of access and borrowing rights based on user categories such as students, faculty, researchers, or the public. The system also tracks user activity, providing valuable insights into usage patterns and preferences.
5. **Search and Discovery**: One of the primary goals of a library is to make information accessible. The LMS includes advanced search capabilities, enabling users to search the library’s catalog using various criteria such as keyword, author, subject, or ISBN. Some systems also offer federated search options, allowing users to search across multiple databases and resources simultaneously.

**OBJECTIVES**

**1. Efficient Resource Management**

* **Cataloging and Classification:** Organize and classify library materials systematically to facilitate easy retrieval and efficient management.
* **Acquisition Management:** Streamline the process of acquiring new materials, from ordering to cataloging, ensuring that the library's collection is up-to-date and well-maintained.

**2. Enhanced User Experience**

* **Search and Discovery:** Provide users with intuitive search capabilities to easily find and access both physical and digital resources.
* **Personalized Services:** Offer personalized user profiles and services, such as tailored recommendations, borrowing histories, and easy access to reserved items.

**3. Optimized Circulation**

* **Automated Check-in/Check-out:** Manage the lending of materials efficiently through automated check-in/check-out processes, tracking borrowed items, due dates, and returns.
* **Overdue Management:** Automatically handle overdue notices, renewals, and fines, ensuring that materials are returned promptly and fairly.

**4. Comprehensive Reporting and Analytics**

* **Collection Development:** Use data-driven insights to guide the acquisition of new materials, ensuring that the library’s collection remains relevant and diverse.

**5. Integration and Interoperability**

* **System Integration:** Ensure seamless integration with other systems such as student information systems, financial management platforms, and online public access catalogs (OPAC).

**TOOLS AND ENVIORNMENT**

**HARDWARE REQUIREMENTS**

**Processor:** Minimum Pentium IV 2.4 GHZ

**RAM:** At Least 100 MB

**Disk Space:** At Least 500 MB

# **SOFTWARE REQUIREMENTS**

**Operating System:** Windows,IOS,LINUX,Etc.

**Code Compiler :** Visual Code Studio / Dev C++/ Turbo C++/Etc.

**ENTITY RELATIONSHIP DIAGRAM**

**MANAGES**

**LIBRARY**

**ADMIN**

**HAS**

**BOOKS**

**CUSTOMER**

**ISSUED BY**

**DATA FLOW DIAGRAM**

LIBRARY MANAGEMENT SYSTEM

VIEW AVAILABILTIYRETURN BOOKS

ADMIN

ISSUE BOOKS

CUSTOMER

ADD BOOKS

VIEW AVAILABLITY

DATA DATA

DATABASE

**PROGRAM CODE**

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

class Book {

public:

int id;

string title;

string author;

int availableCopies;

void addBook(int bookId, string bookTitle, string bookAuthor, int copies) {

id = bookId;

title = bookTitle;

author = bookAuthor;

availableCopies = copies;

}

void displayBook() {

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

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

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

if (availableCopies > 0) {

cout << "Status: Available (" << availableCopies << " copies)" << endl;

} else {

cout << "Status: Not Available" << endl;

}

}

};

class Library {

public:

Book books[100];

int bookCount;

void saveToFile() {

ofstream outFile("library\_data.txt");

if (outFile.is\_open()) {

for (int i = 0; i < bookCount; i++) {

outFile << books[i].id << endl;

outFile << books[i].title << endl;

outFile << books[i].author << endl;

outFile << books[i].availableCopies << endl;

}

outFile.close();

} else {

cout << "Unable to open file for writing." << endl;

}

}

void loadFromFile() {

ifstream inFile("library\_data.txt");

if (inFile.is\_open()) {

while (inFile >> books[bookCount].id && bookCount < 100) {

inFile.ignore(); // Ignore newline character after id

getline(inFile, books[bookCount].title);

getline(inFile, books[bookCount].author);

inFile >> books[bookCount].availableCopies;

inFile.ignore(); // Ignore newline character after availableCopies

bookCount++;

}

inFile.close();

}

}

Library() {

bookCount = 0;

loadFromFile();

}

void addBook() {

if (bookCount < 100) {

int id, copies;

string title;

string author;

cout << "Enter Book ID: ";

cin >> id;

cin.ignore(); // To ignore the newline character after entering the ID

cout << "Enter Book Title: ";

getline(cin, title);

cout << "Enter Book Author: ";

getline(cin, author);

cout << "Enter number of copies: ";

cin >> copies;

books[bookCount].addBook(id, title, author, copies);

saveToFile();

bookCount++;

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

} else {

cout << "Library is full. Cannot add more books." << endl;

}

}

void displayBooks() {

if (bookCount == 0) {

cout << "No books available in the library." << endl;

} else {

for (int i = 0; i < bookCount; i++) {

books[i].displayBook();

cout << "----------------------" << endl;

}

}

}

void issueBook() {

int id;

cout << "Enter Book ID to issue: ";

cin >> id;

for (int i = 0; i < bookCount; i++) {

if (books[i].id == id) {

if (books[i].availableCopies > 0) {

books[i].availableCopies--;

saveToFile(); // Save changes to file

cout << "Book issued successfully!" << endl;

} else {

cout << "Book is not available." << endl;

}

return;

}

}

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

}

void returnBook() {

int id;

cout << "Enter Book ID to return: ";

cin >> id;

for (int i = 0; i < bookCount; i++) {

if (books[i].id == id) {

books[i].availableCopies++;

saveToFile();

cout << "Book returned successfully!" << endl;

return;

}

}

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

}

void adminMenu() {

int choice;

do {

cout << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Admin Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

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

cout << "2. Display All Books" << endl;

cout << "3. Exit Admin Menu" << endl;

cout << endl;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

addBook();

break;

case 2:

displayBooks();

break;

case 3:

cout << "Exiting Admin Menu..." << endl;

break;

default:

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

}

} while (choice != 3);

}

void customerMenu() {

int choice;

do {

cout << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Customer Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "1. Issue Book" << endl;

cout << "2. Return Book" << endl;

cout << "3. Check Book Availability" << endl;

cout << "4. Exit Customer Menu" << endl;

cout << endl;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

issueBook();

break;

case 2:

returnBook();

break;

case 3:

displayBooks();

break;

case 4:

cout << "Exiting Customer Menu..." << endl;

break;

default:

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

}

} while (choice != 4);

}

};

int main() {

Library lib;

int userchoice;

string admin\_password;

string admin\_id;

string customer\_mobile\_no;

string customer\_password;

do {

cout << endl;

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

cout << "1. Customer Menu" << endl;

cout << "2. Admin Menu" << endl;

cout << "3. Exit" << endl;

cout << "Enter your choice: ";

cin >> userchoice;

switch (userchoice) {

case 1: {

do {

cout << endl << "Enter your mobile number: ";

cin >> customer\_mobile\_no;

if (customer\_mobile\_no.length() != 10) {

cout << "Mobile number must be exactly 10 digits long. Please try again." << endl;

}

} while (customer\_mobile\_no.length() != 10);

cout << "Enter your password: ";

cin >> customer\_password;

if (customer\_mobile\_no == "8595632225" && customer\_password == "209") {

lib.customerMenu();

} else {

cout << "Invalid mobile number or password!" << endl;

}

break;

}

case 2: {

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

cin >> admin\_id;

cout << "Enter admin password: ";

cin >> admin\_password;

if (admin\_id == "shubham" && admin\_password == "209") {

lib.adminMenu();

} else {

cout << "Invalid admin ID or password!" << endl;

}

break;

}

case 3:

cout << "Exiting the program..." << endl;

break;

default:

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

}

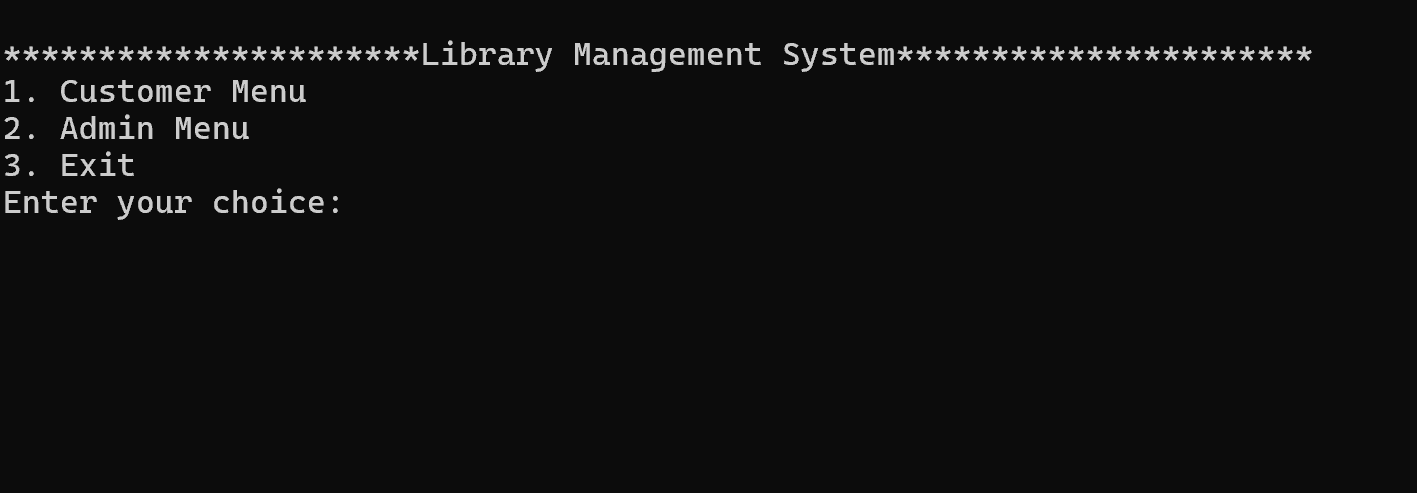
} while (userchoice != 3);

return 0;

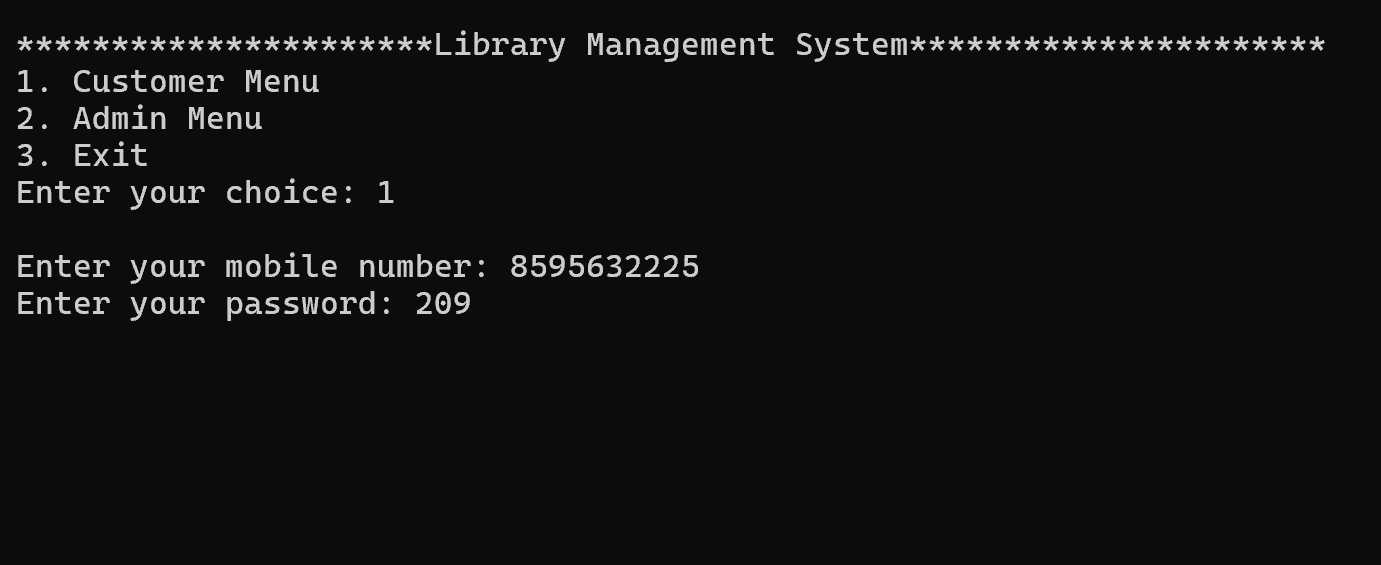
}

**INPUT / OUTPUT SCREENS**

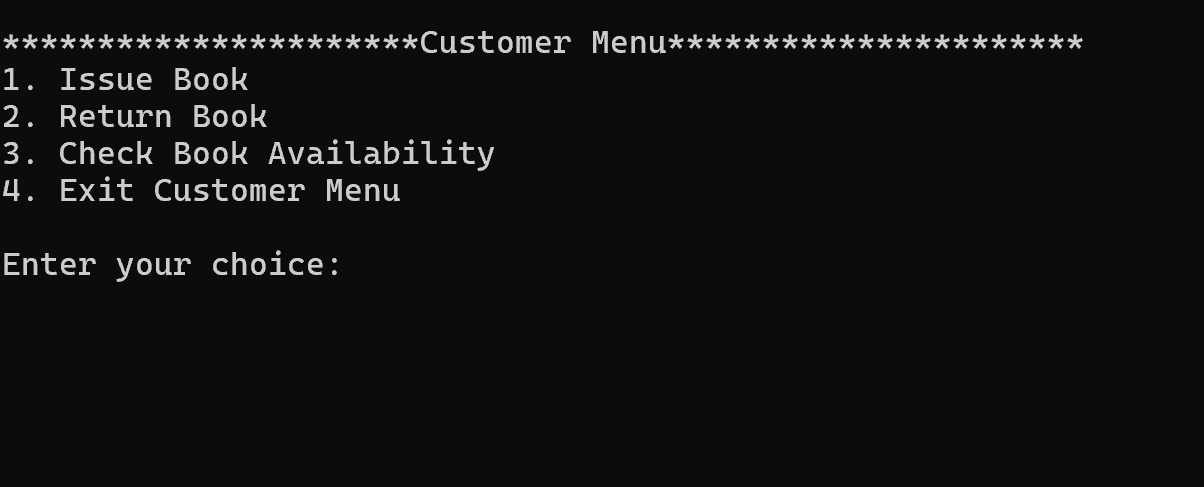
MAIN MENU



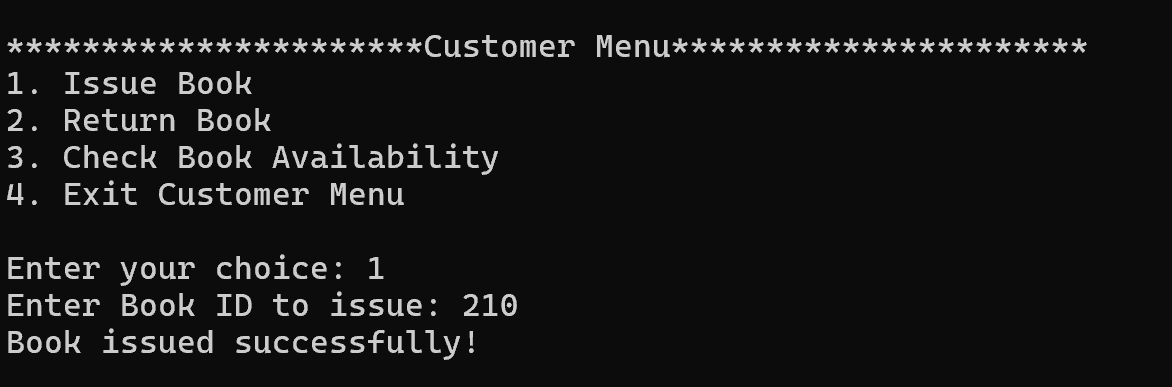
CUSTOMER LOGIN



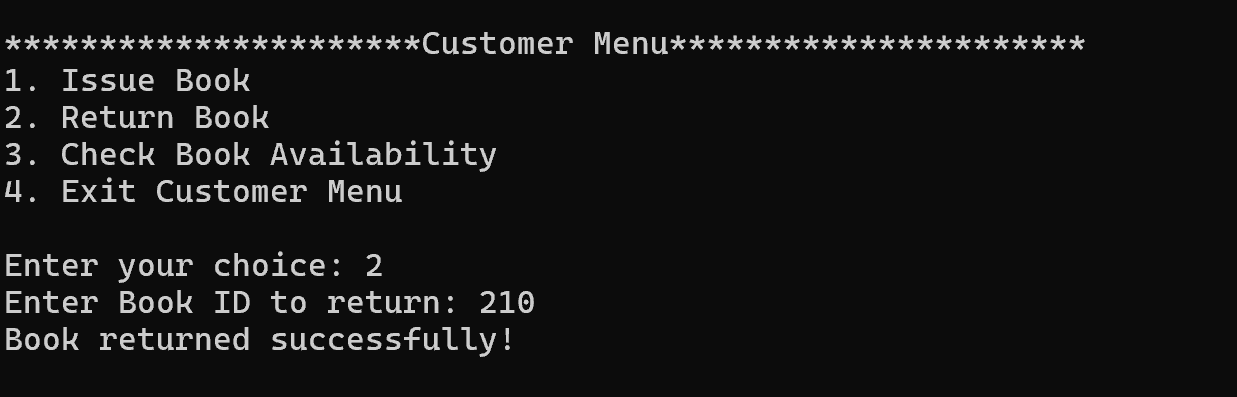
CUSTOMER MENU



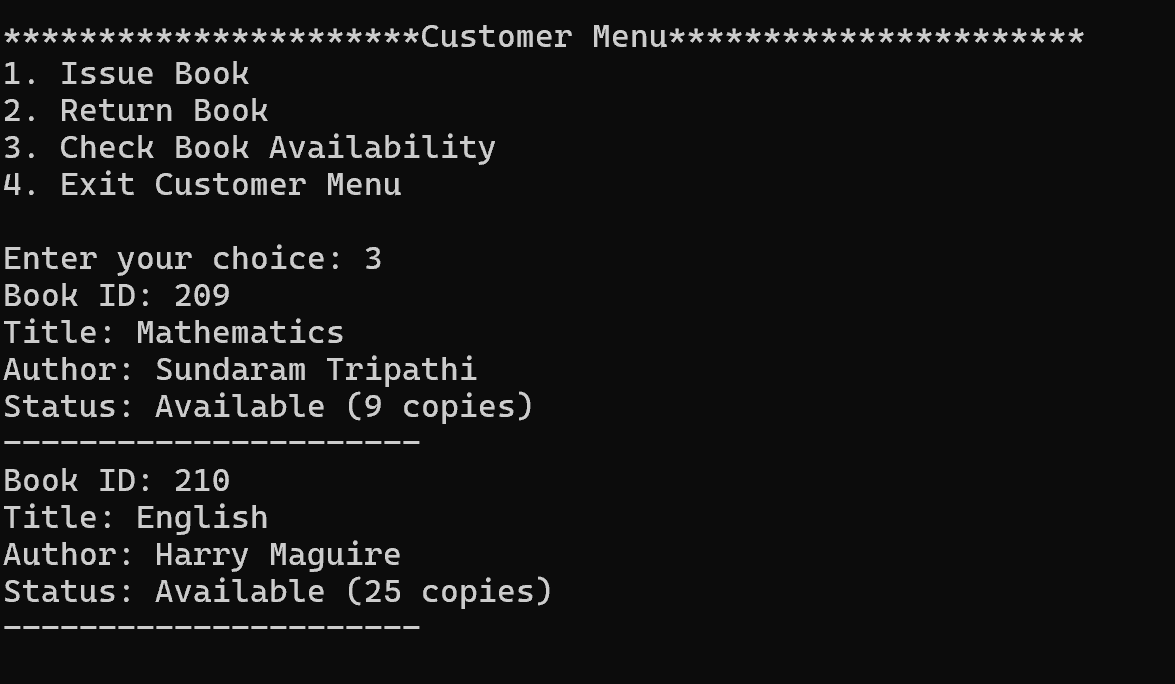
ISSUE BOOK



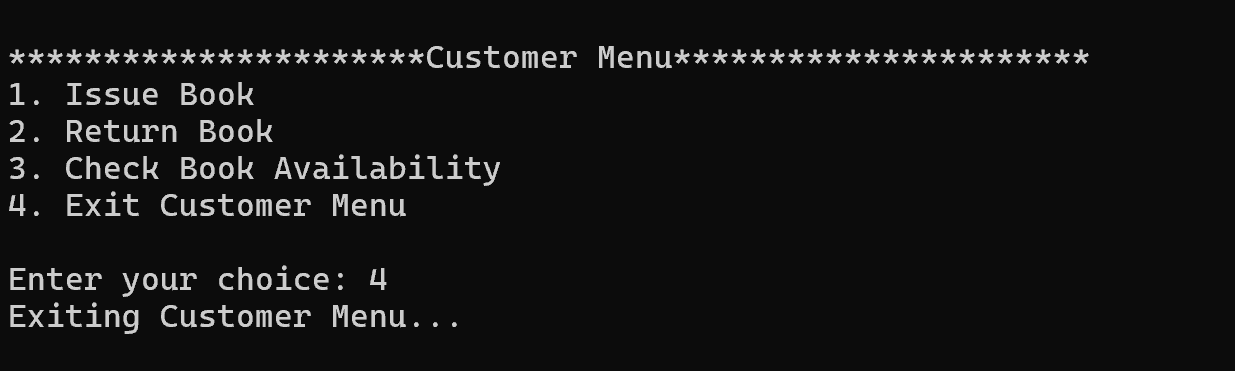
RETUEN BOOK



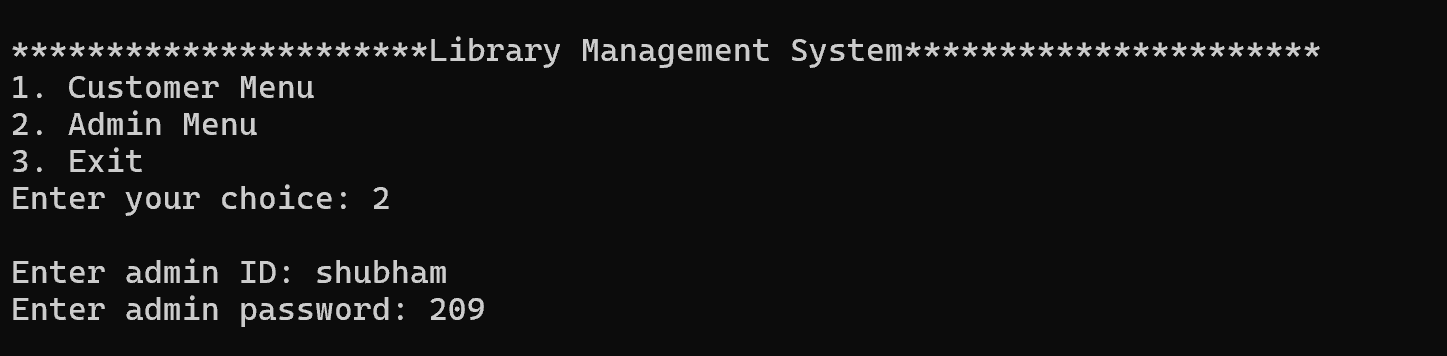
CHECK AVAILABILITY



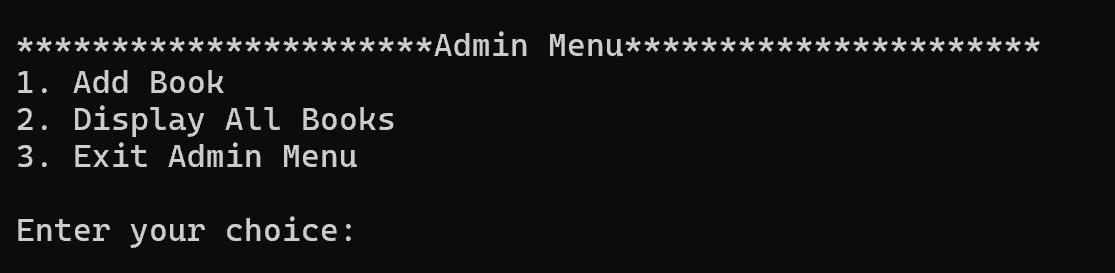
EXITING AS CUSTOMER



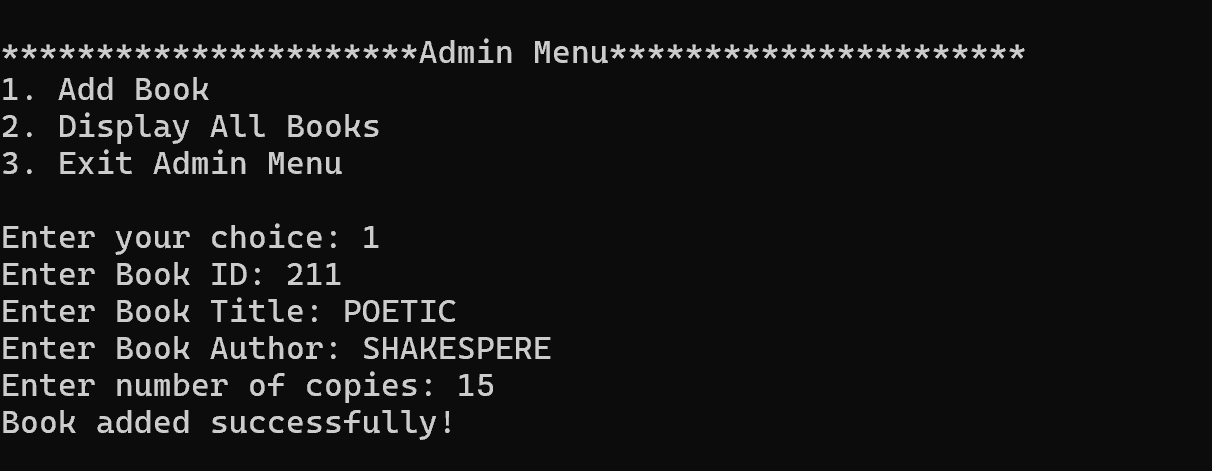
ADMIN LOGIN



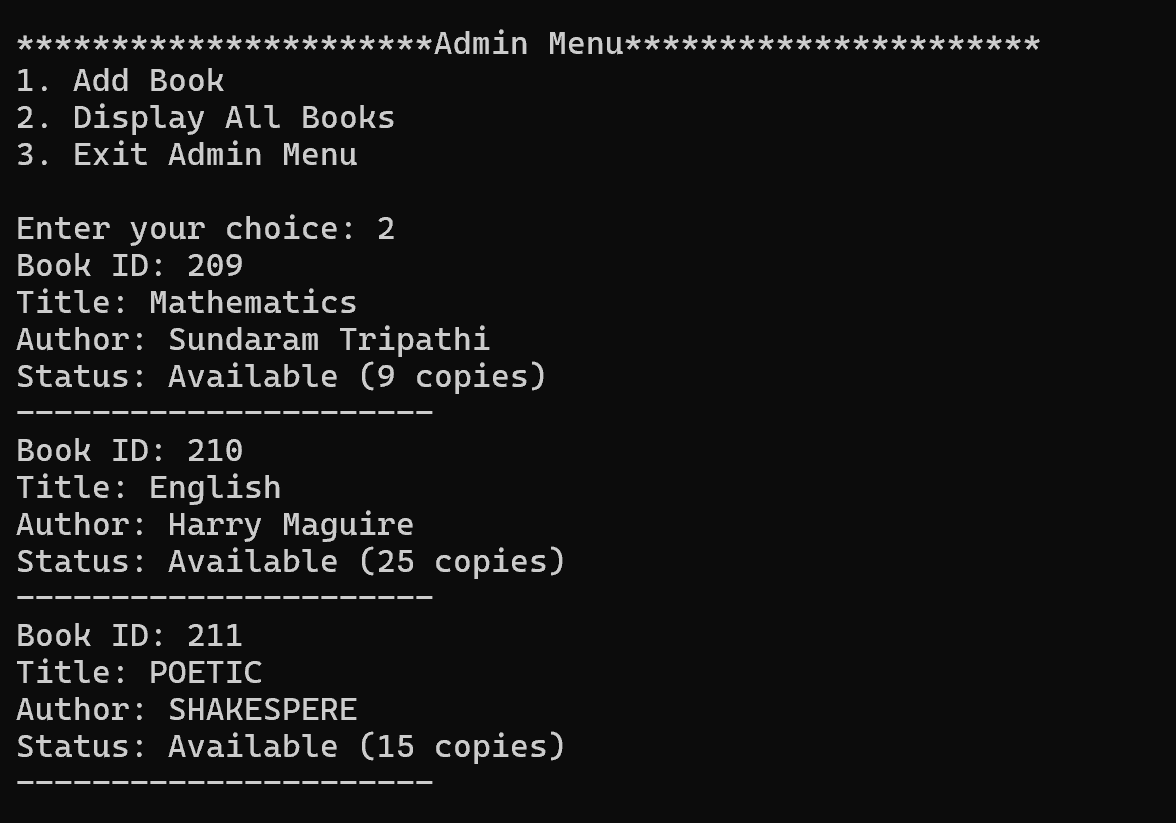
ADMIN MENU



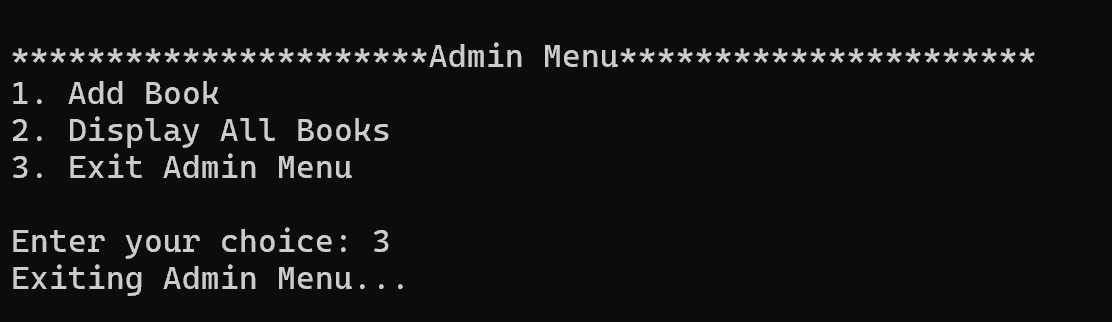
ADD BOOKS



DISPLAY ALL BOOKS



EXITING AS ADMIN



**LIMITATIONS OF LIBRARY MANAGEMENT SYSTEM**

**1.** **High Initial Cost**

* Implementation Costs: The initial cost of purchasing and implementing an LMS can be high, including expenses related to software licensing, hardware, and setup.

**2. Complexity and Learning Curve**

* Training Requirements: Library staff may require significant training to effectively use the system, especially if the LMS is complex or feature-rich.

**3. Dependence on Internet Connectivity**

* Internet Dependency: For cloud-based systems, a reliable internet connection is essential. Any disruption in connectivity can impact the availability and functionality of the LMS.

**4. Data Migration Challenges**

* Legacy Systems: Migrating data from legacy systems to a new LMS can be complex and time-consuming. There is also a risk of data loss or corruption during the migration process.

**5. Limited Flexibility**

* Customization Constraints: Some LMS platforms may have limited options for customization, making it difficult to adapt the system to specific library needs or workflows.
* Vendor Lock-In: Libraries might become dependent on a specific vendor for updates, support, and customization, limiting their flexibility to switch providers or make independent changes.

**FUTURE APPLICATION OF THE PROJECT**

The future applications of a Library Management System (LMS) project are driven by the evolving needs of libraries, advancements in technology, and the growing demand for seamless access to information.

**1. Integration with Artificial Intelligence (AI)**

* **Personalized Recommendations:** AI-driven algorithms could analyze user behavior and preferences to offer personalized resource recommendations, similar to how streaming services suggest content based on viewing history.
* **Automated Cataloging:** AI could assist in automating the cataloging process by analyzing and classifying new materials with minimal human intervention, speeding up the process and ensuring consistency.

**2. Enhanced User Experience through Augmented Reality (AR)**

* **Interactive Library Navigation:** AR could be used to guide users through physical library spaces, providing an interactive map or overlays that highlight specific sections, new arrivals, or themed collections.
* **Digital Resource Visualization:** Users could interact with digital content in new ways, such as viewing 3D models of historical artifacts or exploring virtual exhibitions that complement the library’s physical collection.

**3. Blockchain for Enhanced Security and Authentication**

* **Secure Transactions:** Blockchain technology could be utilized to secure user transactions, such as borrowing and returning items, ensuring that records are tamper-proof and reliable.
* **Digital Rights Management:** Blockchain could help manage digital rights for e-books and online resources, providing a transparent and secure way to track ownership and usage rights.

**4. Expanded Digital Resource Management**

* **Integration with Open Access Resources:** As open access content grows, the LMS could integrate more seamlessly with open educational resources (OER), providing users with broader access to freely available academic materials.

**BIBLIOGRAPHY**

**Books**

* Let Us C by Yashavant Kanetkar.
* Let us C++ by Yashavant Kanetkar.
* C in Depth by S.K Srivastava.
* The C++ Programming Language By Bjarne Stroustrup.

**Websites**

* [www.google.com](http://www.google.com)
* [www.youtube.com](http://www.youtube.com)
* [www.w3schools.com](http://www.w3schools.com)
* [www.geeksforgeeks.com](http://www.geeksforgeeks.com)