**LIBRARY**

**MANAGEMENT SYSTEM**

**A PROJECT REPORT**

**Submitted By :- Dheeraj Kumar**

**Date :- 21-08-2024**

**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 an advanced software platform meticulously crafted to optimize the management and operations of libraries. Libraries serve as essential institutions in knowledge dissemination, housing vast collections of books, journals, digital media, and other resources. Effective management of these resources is crucial to ensuring that users—including students, researchers, and the general public—can access the information they need efficiently and systematically. An LMS provides libraries with the tools to automate cataloging, circulation, and resource tracking, ultimately enhancing user experience and operational efficiency.

Library Management System

The transformation of libraries from traditional physical spaces to hybrid environments that include digital resources has driven the need for advanced management systems. A Library Management System (LMS) meets this demand by offering a comprehensive suite of tools and functionalities that cover a broad spectrum of tasks, including cataloging, acquisition, circulation, and user management. By integrating various library functions into a single platform, an LMS minimizes redundancy, enhances accuracy, and significantly improves 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 <fstream>

#include <string>

#include <iomanip>

using namespace std;

class Book {

public:

int bookId;

char title[50];

char author[50];

bool isAvailable;

void addBook();

void displayBook() const;

void getAllBooks() const;

void updateBook();

void deleteBook();

bool isBookExists(int) const;

};

class Patron {

public:

void borrowBook();

void returnBook();

void viewBorrowedBooks() const;

};

bool Book::isBookExists(int id) const {

ifstream checkFile("books.dat", ios::binary);

Book b;

while (checkFile.read((char\*)&b, sizeof(b))) {

if (id == b.bookId) {

checkFile.close();

return true;

}

}

checkFile.close();

return false;

}

void Book::addBook() {

cout << "\nEnter book title: ";

cin.ignore();

cin.getline(title, 50);

cout << "Enter book author: ";

cin.getline(author, 50);

cout << "Enter book ID: ";

while (!(cin >> bookId)) {

cout << "Enter number only: ";

cin.clear();

cin.ignore(123, '\n');

}

if (isBookExists(bookId)) {

cout << "Error: Book with ID " << bookId << " already exists\n";

} else {

isAvailable = true;

ofstream file("books.dat", ios::binary | ios::app);

if (file) {

file.write((char\*)this, sizeof(\*this));

file.close();

cout << "\nBook added successfully...\n";

} else {

cout << "Error: Could not open file to save book data.\n";

}

}

}

void Book::displayBook() const {

cout << setw(5) << bookId << setw(25) << title << setw(25) << author << setw(10) << (isAvailable ? "Available" : "Not Available") << endl;

}

void Book::getAllBooks() const {

ifstream file("books.dat", ios::binary);

if (!file) {

cerr << "Error: Could not open file.\n";

return;

}

Book b;

bool found = false; // To check if any book exists

// Print headings

cout << setw(5) << "ID" << setw(25) << "Title" << setw(25) << "Author" << setw(10) << "Status" << endl;

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

while (file.read((char\*)&b, sizeof(b))) {

found = true;

b.displayBook();

}

if (!found) {

cout << "There are no books.\n";

}

file.close();

}

void Book::updateBook() {

int id, pos;

bool found = false;

cout << "\nEnter book ID to update: ";

cin >> id;

fstream file("books.dat", ios::binary | ios::in | ios::out);

Book b;

while (file.read((char\*)&b, sizeof(b))) {

if (b.bookId == id) {

cout << "Enter new book title: ";

cin.ignore();

cin.getline(b.title, 50);

cout << "Enter new book author: ";

cin.getline(b.author, 50);

pos = -1 \* static\_cast<int>(sizeof(b));

file.seekp(pos, ios::cur);

file.write((char\*)&b, sizeof(b));

found = true;

cout << "\nBook updated successfully...\n";

break;

}

}

if (!found) {

cout << "Invalid Book ID\n";

}

file.close();

}

void Book::deleteBook() {

int id;

bool found = false;

cout << "\nEnter book ID to delete: ";

cin >> id;

ifstream file("books.dat", ios::binary);

ofstream tempFile("temp.dat", ios::binary);

Book b;

while (file.read((char\*)&b, sizeof(b))) {

if (b.bookId != id) {

tempFile.write((char\*)&b, sizeof(b));

} else {

found = true;

}

}

file.close();

tempFile.close();

remove("books.dat");

rename("temp.dat", "books.dat");

if (found) {

cout << "\nBook deleted successfully...\n";

} else {

cout << "Invalid Book ID\n";

}

}

void Patron::borrowBook() {

int id;

bool found = false;

cout << "Enter book ID to borrow: ";

cin >> id;

fstream file("books.dat", ios::binary | ios::in | ios::out);

Book b;

while (file.read((char\*)&b, sizeof(b))) {

if (b.bookId == id) {

if (b.isAvailable) {

b.isAvailable = false;

int pos = -1 \* static\_cast<int>(sizeof(b));

file.seekp(pos, ios::cur);

file.write((char\*)&b, sizeof(b));

found = true;

cout << "\nBook borrowed successfully...\n";

break;

} else {

cout << "Book is not available for borrowing.\n";

break;

}

}

}

if (!found) {

cout << "Invalid Book ID\n";

}

file.close();

}

void Patron::returnBook() {

int id;

bool found = false;

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

cin >> id;

fstream file("books.dat", ios::binary | ios::in | ios::out);

Book b;

while (file.read((char\*)&b, sizeof(b))) {

if (b.bookId == id) {

if (!b.isAvailable) {

b.isAvailable = true;

int pos = -1 \* static\_cast<int>(sizeof(b));

file.seekp(pos, ios::cur);

file.write((char\*)&b, sizeof(b));

found = true;

cout << "\nBook returned successfully...\n";

break;

} else {

cout << "Book is already available.\n";

break;

}

}

}

if (!found) {

cout << "Invalid Book ID\n";

}

file.close();

}

void Patron::viewBorrowedBooks() const {

ifstream file("books.dat", ios::binary);

if (!file) {

cerr << "Error: Could not open file.\n";

return;

}

Book b;

bool found = false; // To check if any book exists

// Print headings

cout << setw(5) << "ID" << setw(25) << "Title" << setw(25) << "Author" << setw(25) << "Status" << endl;

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

while (file.read((char\*)&b, sizeof(b))) {

if (!b.isAvailable) {

found = true;

b.displayBook();

}

}

if (!found) {

cout << "There are no borrowed books.\n";

}

file.close();

}

// Function for admin login

bool adminLogin() {

string username, password;

const string adminUsername = "dhzzraj";

const string adminPassword = "dheeraj167"; // For demonstration purposes

cout << "Enter admin username: ";

cin >> username;

cout << "Enter admin password: ";

cin >> password;

return (username == adminUsername && password == adminPassword);

}

// Function for customer login

bool customerLogin() {

string username, password;

const string customerUsername = "customer";

const string customerPassword = "customer167"; // For demonstration purposes

cout << "Enter customer username: ";

cin >> username;

cout << "Enter customer password: ";

cin >> password;

return (username == customerUsername && password == customerPassword);

}

int main() {

int choice;

bool isAdmin = false;

bool isCustomer = false;

while (true) {

cout << "\n::::::::::||LIBRARY MANAGEMENT SYSTEM||::::::::::\n";

cout << "1. Admin Login\n";

cout << "2. Customer Login\n";

cout << "3. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1: // Admin Login

isAdmin = adminLogin();

if (isAdmin) {

while (true) {

cout << "\n::::::::::||ADMIN MENU||::::::::::\n";

cout << "1. Add Book\n";

cout << "2. Display All Books\n";

cout << "3. Update Book\n";

cout << "4. Delete Book\n";

cout << "5. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

Book book;

switch (choice) {

case 1:

book.addBook();

break;

case 2:

book.getAllBooks();

break;

case 3:

book.updateBook();

break;

case 4:

book.deleteBook();

break;

case 5:

isAdmin = false;

break;

default:

cout << "Invalid choice. Please try again.\n";

}

if (!isAdmin) break;

}

} else {

cout << "Invalid admin credentials.\n";

}

break;

case 2: // Customer Login

isCustomer = customerLogin();

if (isCustomer) {

while (true) {

cout << "\n::::::::::||CUSTOMER MENU||::::::::::\n";

cout << "1. Borrow Book\n";

cout << "2. Return Book\n";

cout << "3. View Borrowed Books\n";

cout << "4. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

Patron patron;

switch (choice) {

case 1:

patron.borrowBook();

break;

case 2:

patron.returnBook();

break;

case 3:

patron.viewBorrowedBooks();

break;

case 4:

isCustomer = false;

break;

default:

cout << "Invalid choice. Please try again.\n";

}

if (!isCustomer) break;

}

} else {

cout << "Invalid customer credentials.\n";

}

break;

case 3:

return 0;

default:

cout << "Invalid choice. Please try again.\n";

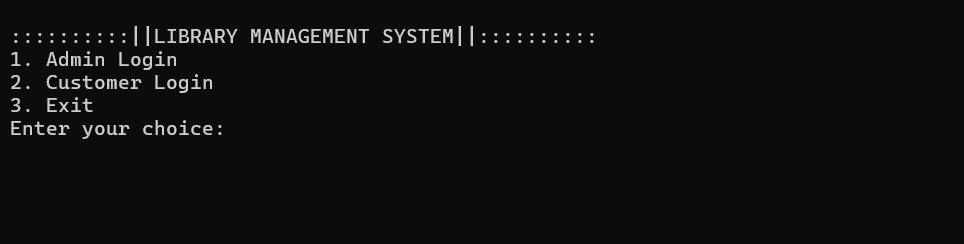
}

}

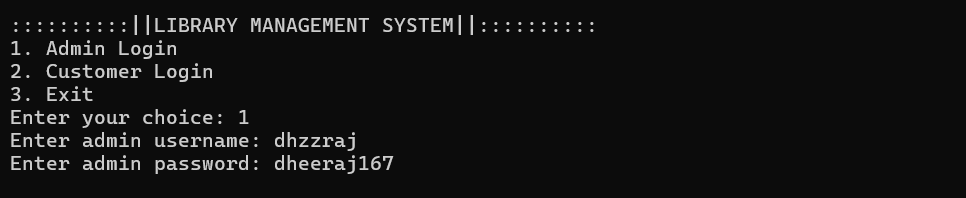
return 0;

}

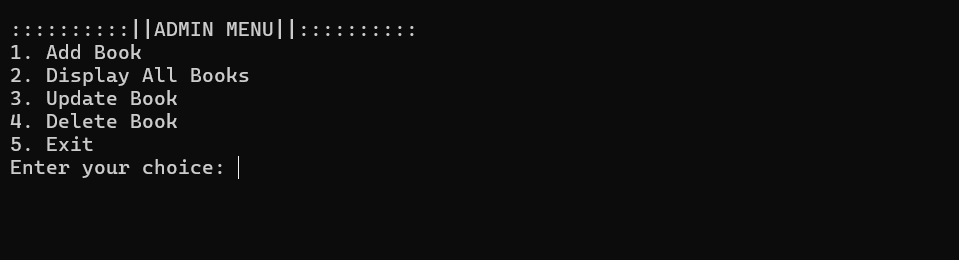
**INPUT / OUTPUT SCREENS**

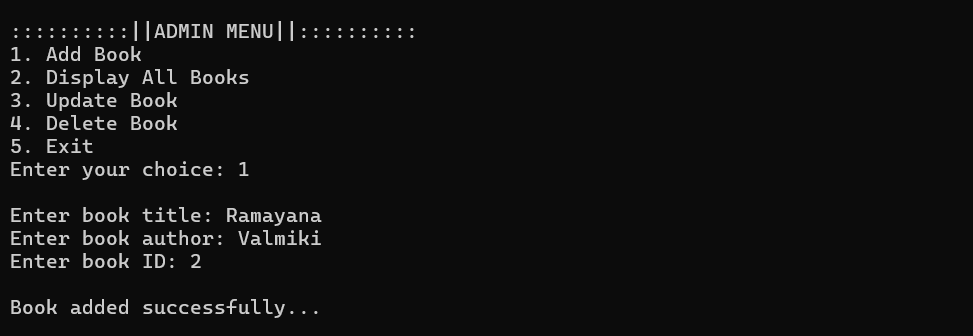
**MAIN MENU**

**ADMIN LOGIN**

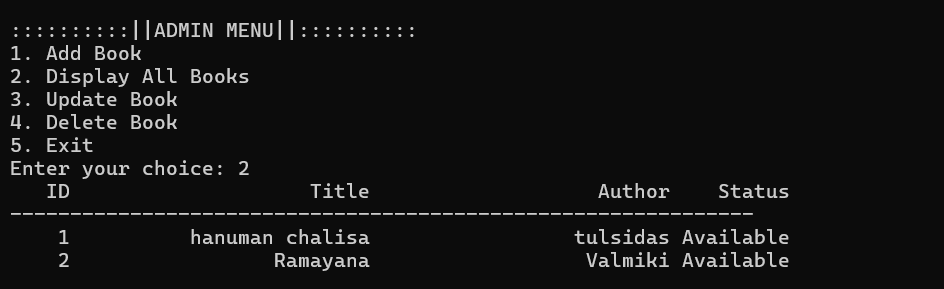


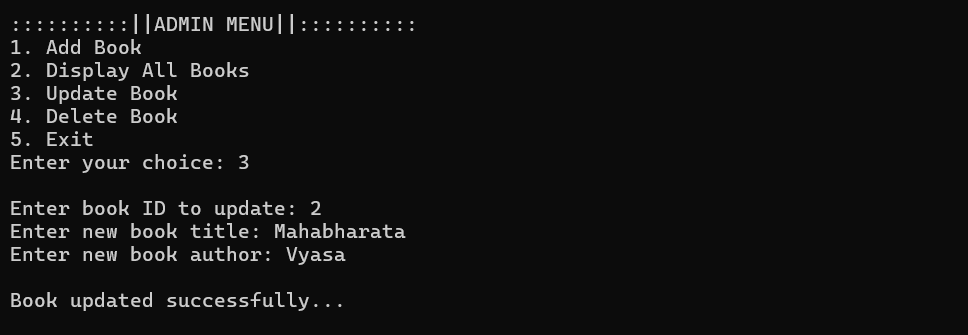
**ADMIN MENU**

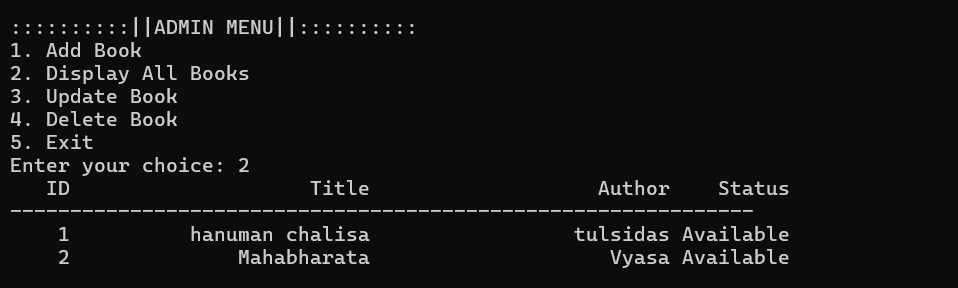


**ADD BOOK**

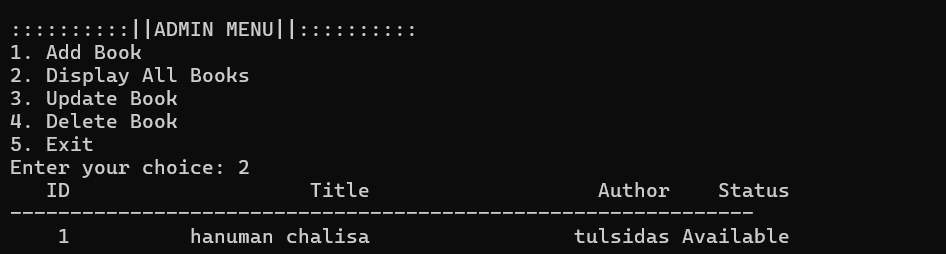
**DISPLAY ALL BOOKS**

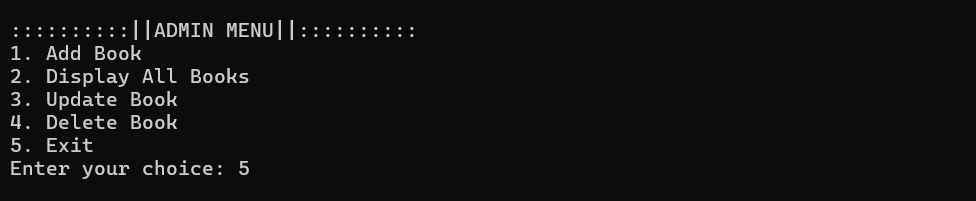


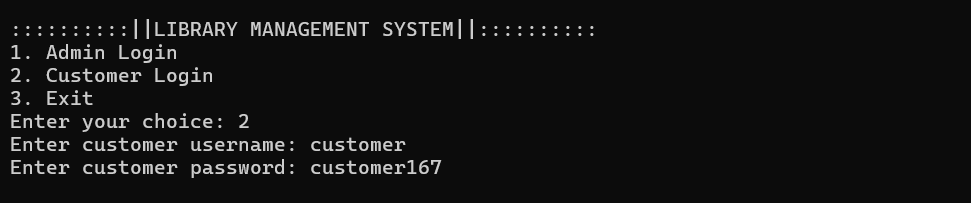
**UPDATE BOOK**

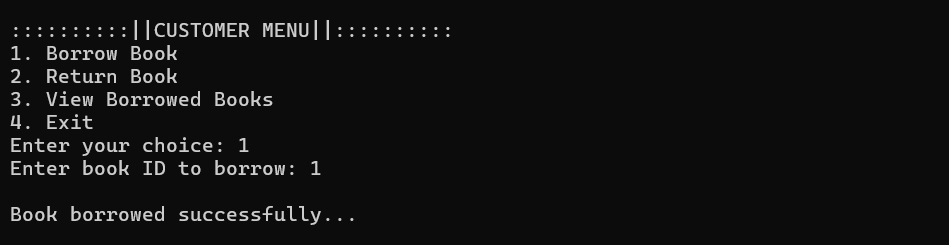
**AFTER UPDATE BOOK**

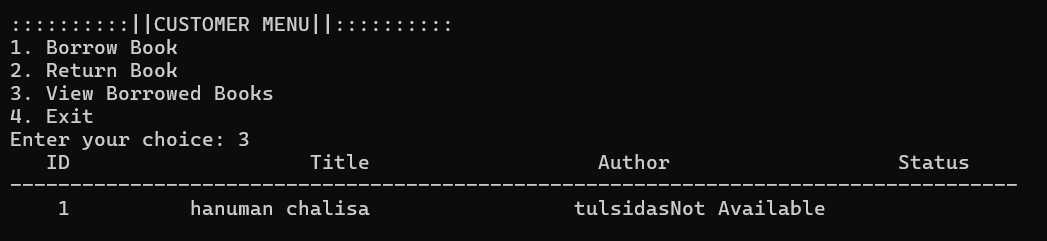
**DELETE BOOK**

**AFTER DELETED BOOK**

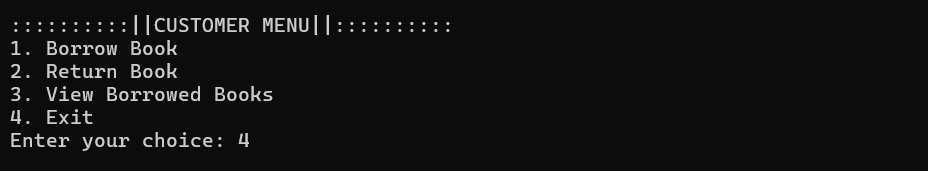
**EXITING FROM ADMIN’S MENU**

**CUSTOMER LOGIN** 

**BORROW BOOK**

**VIEW BORROWED BOOKS**

**RETURN BOOK**

**EXITING FROM CUSTOMER’S MENU**

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