NATIONAL UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE
BS CYBER SECURITY

PROGRAMMING FUNDAMENTAL PROJECT

"Library Management System"

Group Members:

Mahnoor Fatima	F24609043
Hajra Shehzad	F24609025

ACKNOWLEDGMENT

First of all, we are thankful to Allah Almighty the Merciful, the Most Beneficent, and the source of all Knowledge, for granting us the courage, understanding, and knowledge to complete this Project.We are thankful to our parents, who supported us wholeheartedly in our studies and the position in which we are standing today is only possible because of their efforts. I express my appreciation to our supervisor **SIR RIZWAN YOUSAF** for supporting and providing us with the opportunity to enhance our learning and knowledge. I would like to mention my siblings and Friends, here who were there to help us whenever we got stuck somewhere in the development or later phases.

MAHNOOR FATIMA
F-24609043
HAJRA SHEHZAD
F-24609025

Project Title:	**LIBRARY MANAGEMENT SYSTEM**
Objective:	This project aims to provide automate library management system.
Undertaken By:	Mahnoor Fatima F24609043 Hajra Shehzad F24609025
Supervised By:	**Rizwan Yousaf**
Date Started:	Nov-2024
Date Completed:	Dec-2024
Tools, Technologies And Language Used:	Front End: SFML Back End: C++ Other Tools: Dev c++
Operating System Used:	Microsoft Window 11
Systems Used:	HP Elitebook Dell

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

This chapter introduces the requirements and scope of the Library Management System. It outlines the need for such a system and its potential benefits for the library and its users.

1.1 INTRODUCTION OF PROJECT

A Library Management System is a software application designed to automate and streamline various library operations. It manages the entire lifecycle of library resources, including books, journals, and other materials. This system helps to improve efficiency, accuracy, and accessibility of library resources for both staff and patrons.

1.2 PURPOSE

The primary purpose of a Library Management System is to streamline library operations, enhance user experience, and improve overall efficiency. By automating tasks like book cataloging, circulation, and acquisitions, the system reduces manual effort, minimizes human error, and saves valuable time and resources. It provides a centralized platform for managing library resources, tracking user information, and generating reports, enabling librarians to focus on providing better services to their patrons.

- Automate Library Services
- Improve Resource management
- Enhance User Experience

1.3 PROJECT MOTIVATION

Traditional library management systems often involve manual processes, which can be time consuming, prone to errors, and inefficient. This project aims to address these challenges by developing a computerized system that streamlines library operations and enhances the overall library experience for both staff and patrons.

1.4 SCOPE

This Library Management System will focus on the following core functionalities:

1.4.1 Book Management:

- Add, edit, and delete book records.
- Search for books by title, author, ISBN, and keywords.
- Generate reports on book availability and circulation.

1.4.2 User Management:

- Register new library members.
- Manage user accounts and information.
- Track user borrowing history.

CHAPTER 2

PROBLEM ANALYSIS

2. PROBLEM ANAYLSIS

This chapter analyzes the existing manual library operations and identifies the challenges associated with them. It then presents the proposed computerized system as a solution to these challenges

2.2 EXISITING SYSTEM

Traditional library operations often involve manual processes. Issuing and returning books, recording borrower information, and calculating fines manually. These include few drawbacks:

- **Manual cataloging:** Cataloging new books, assigning Dewey Decimal numbers, and creating card catalogs.
- **Manual circulation:** Issuing and returning books, recording borrower information, and calculating fines manually.
- **Manual record keeping:** Maintaining physical records of book inventory, user information, and circulation history.
- **Manual searches:** Conducting manual searches for books in card catalogs or physical indexes.

2.3 PROPOSED SYSTEM

Our proposed Library Management System is a computerized solution designed to address the limitations of manual systems. It will automate various library operations, including:

- **Automated Circulation:** Issuing and returning books, calculating fines, and generating circulation reports.
- **Online Catalog:** Providing online access to the library catalog for easy searching and browsing.
- **User Management:** Managing user accounts, tracking borrowing history, and providing online services.
- **Inventory Management:** Maintaining an accurate and up-to-date inventory of library resources.

2.4 STALK HOLDOERS

- Admin
- Students

2.5 USER-GOAL LIST

User Roles	**Description**
Admin	• Admin can add staff. • Admin can display staff. • Admin can add book. • Admin can search book.
Student	• Student can register themselves. • Student can view registered students list. • Student can issue book. • Student can return book.

CHAPTER 03

SYSTEM ANALYSIS

3. SYSTEM ANALYSIS

System Analysis is the task that links and connects the system level requirement and designing of the project. It ensures the careful consideration of various factors to ensure the success of Library Management project. It helps in designing the document based on analysis gathered by requirements.

3.1 PROBLEM OVERVIEW

As more and more of our life is transacted online, expectations grow about what services are offered from a college or university and how they are delivered. Whether you are enrolling a new student or dealing with a current system is a cornerstone technology for online access to library resources and enabling self-service options.

3.2 FUNCTIONAL REQUIREMENTS

Functional requirements define the specific tasks and operations that the Library Management System must perform. The system will be divided into the following modules:

3.2.1 STUDENT MANAGEMENT MODULES

3.2.1.1 Add student

Requirements	Description
Name	User has to provide his name that is on his registration card.
ID	User has to provide his registered Id (F-897374)

3.2.1.2 Display Student

Requirement	Desription
Student Id	Id of registered student will display.
Student name	Name of registered student will display.
Book status	Status that either book is borrowed by student or not is displayed.

3.2.1.2 Check-in Book

Requirements	Description
ID	Student has to enter registered ID.
Book no.	If book number entered by user is available the book will issue otherwise No book found message is display.

3.2.1.3 Check-out Book

Requirements	Description
ID	User will simply add Id number and if Book was issued by this ID then it will successfully returned itherwise message will display "User has not borrowed any book".

3.2.2 ADMIN MANAGEMENT MODULES

3.2.2.1 Registering Staff

Requirement	Description
ID	Staff has to enter his Id number that is registered on his name.
Name	Staff has to enter his name to register as library management staff.

3.2.2.2 Display Staff

Requirement	Desription
Staff name	Id of registered student will display.
Book status	Status that either book is borrowed by staff or not is displayed.(staff is also allowed to borrow books.)

3.2.2.3 Search Book

Requirement	Description
Book no	Staff member has to enter book number to search it in library catalog and message will display either it is present or not.

3.2.2.4 Add Book

Requirement	Description
Book no.	Staff has to enter book number to add book in library.
Book name	Staff has to enter book name to add book.
Author name	Staff has to enter author name of book.

CHAPTER 04

SYSTEM DESIGN

4. SYSTEM DESIGN

This chapter outlines the core system design considerations for the library management project. We will explore the utilization of arrays and functions to efficiently manage the various components of the library's operations.

4.1DATA STRUCTURES

4.1.1 ARRAYS:

4.2 Used to store data for students, books, and staff.

4.3 Provides efficient storage and access for basic operations.

4.4 Arrays for student data: studentAdmissionNo, studentNames, studentBorrowed Books.

4.5 Arrays for book data: bookNo, bookNames, bookAuthors, bookIssued.

4.6 Arrays for staff data: staffID, staffNames.

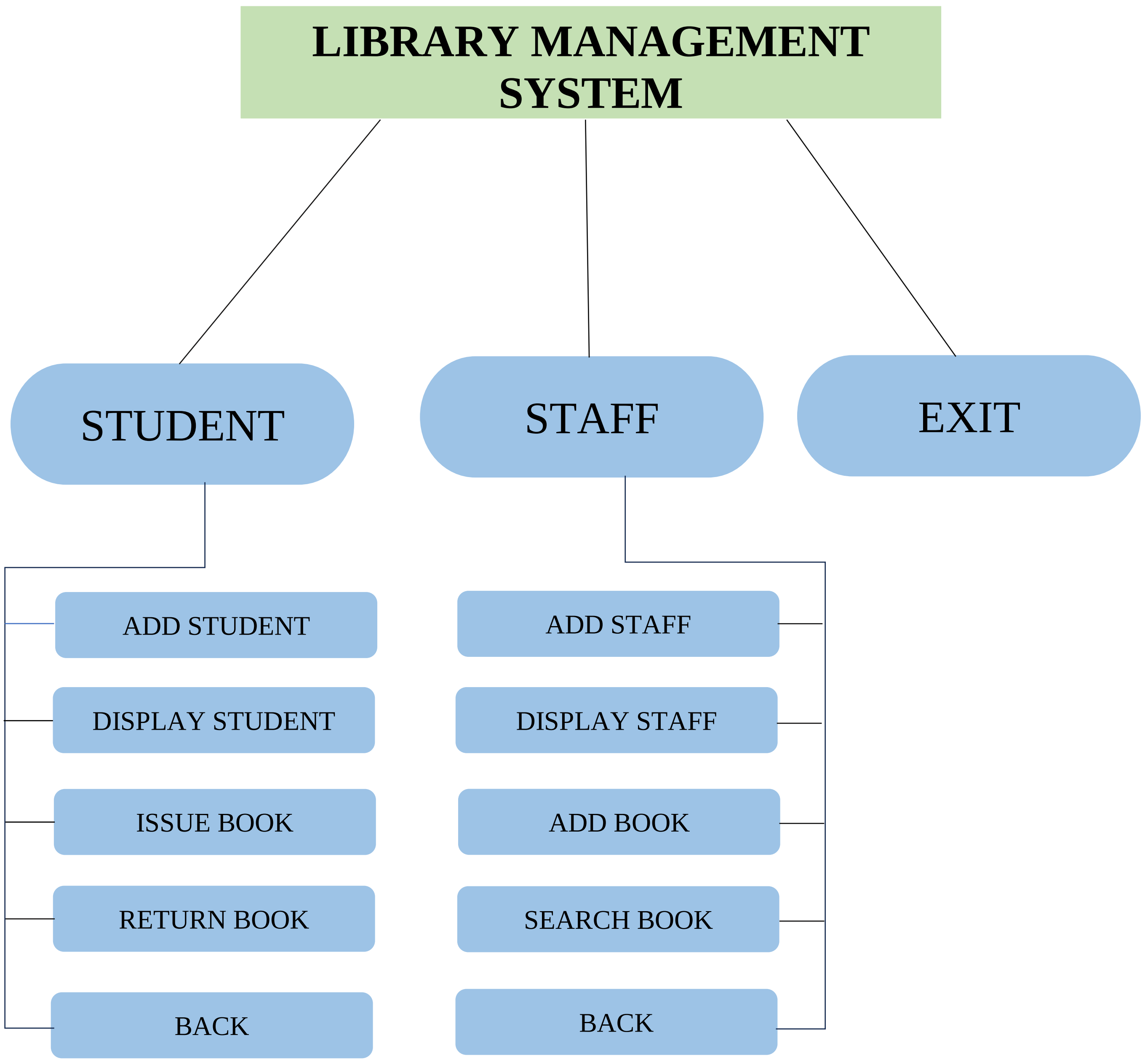
4.2 SYSTEM ARCHITECTURE

4.2.1 MODULAR DESIGN:

- Main Module: Handles user input, displays the main menu, and coordinates interactions between other modules.
- Student Module: Includes functions for student-related operations (adding, displaying information, issuing/returning books).
- Staff Module: Includes functions for staff-related operations (adding staff, adding books, searching books).

4.2.2 SYSTEM FLOWCHART

- The flowchart visually represent the main steps and decision points in the program's execution.
- It can start with the main menu, branch into student/staff options, and then further detail the actions within each option (e.g., add student, issue book, search book).
- General Functions: Contains utility functions like searchStudent searchBook.


LIBRARY MANAGEMENT SYSTEM
STUDENT
STAFF
EXIT
ADD STUDENT
DISPLAY STUDENT
ISSUE BOOK
RETURN BOOK
BACK
ADD STAFF
DISPLAY STAFF
ADD BOOK
SEARCH BOOK
BACK

CHAPTER 5

IMPLEMENTATION

5. IMPLEMENTATION

This chapter details the implementation of the library management system, focusing on how the code translates the design into a functional program.

5.1 DATA STRUCTURES

5.1.1 USER STRUCTURE:

id: Stores the unique ID of the user (student or staff).
name: Stores the user's full name.
borrowedBook: Stores the ID of the book currently borrowed by the user.

```
struct User {
    string id;
    string name;
    string borrowedBook;
};
```

5.1.2 BOOK STRUCTURE:

bookNo: Stores the unique ID of the book.

name: Stores the title of the book.

author: Stores the author's name.

isIssued: A Boolean flag indicating whether the book is currently issued or available.

```
struct Book {
    string bookNo;
    string name;
    string author;
    bool isIssued;
};
```

5.2 ARRAY-BASED IMPLEMENTATION

5.2.1 USER ARRAYS:

Students: An array of User structures to store student information.

Staff: An array of User structures to store staff information.

5.2.2 BOOK ARRAY:

Books: An array of Book structures to store book information.

```
User students[MAX_USERS];
User staff[MAX_USERS];
Book books[MAX_BOOKS];
```

5.3 FUNCTIONS

Adds a new user (student or staff) to the respective array.

Takes the user array, the current number of users, and the user type (student or staff) as input.

```
void addUser(User users[], int& numUsers, const string& userType)
{

    if (numUsers >= MAX_USERS) {

    cout << "Maximum number of " << userType << "s reached.\n";
    return;
    }
```

- Prompts the user to enter the ID and name of the new user.
- Initializes the borrowedBook field to an empty string.
- Increments the numUsers counter.
- Prints a success message.

```
    cout << "Enter the " << userType << " ID: ";
    cin >> users[numUsers].id;
    cout << "Enter the " << userType << "'s name: ";
    cin.ignore();
    getline(cin, users[numUsers].name);
    users[numUsers].borrowedBook = "";

    numUsers++;
    cout << userType << " record created.\n";
    }
```

- Displays the list of users (students or staff).
- Takes the user array, the number of users, and the user type as input.
- Iterates through the array and prints the name and borrowed book (if any) for each user.

```
void displayUsers(const User users[], int numUsers, const string&
userType)
 {
    cout << "\nList of " << userType << "s:\n";
    for (int i = 0; i < numUsers; i++) {
        cout << "\n" << userType << ": " << users[i].name << endl;
        if (!users[i].borrowedBook.empty()) {
            cout << "Borrowed Book: " << users[i].borrowedBook <<
endl;
        } else {
            cout << "No book borrowed.\n";
        }
```

- Searches for a user in the array based on their ID.
- Takes the user array, the number of users, and the user ID as input.
- Returns the index of the user in the array if found, otherwise returns -1.

```
int searchUser(const User users[], int numUsers, const string&
userID) {

    for (int i = 0; i < numUsers; i++) {
        if (users[i].id == userID) {
            return i;
        }
    }
    return -1;
}
```

- Issues a book to a student.
- Prompts the user to enter their ID and the book number.
- Searches for the student and the book in their respective arrays.
- Checks if the student has already borrowed a book and if the book is already issued.
- If the book is available, updates the student's borrowedBook field and sets the isIssued flag of the book to true.

```
void issueBook(User users[], int numStudents, Book books[], int&
numBooks) {
    string userID, bookNo;

    cout << "Enter user's ID: ";
    cin >> userID;

    int userIndex = searchUser(students, numStudents, userID);
    if (userIndex == -1) {
        cout << "User not found.\n";
        return;
    }

    if (!students[userIndex].borrowedBook.empty()) {
        cout << "User already has a book.\n";
        return;
    }
    cout << "Enter book number to issue: ";
     cin >> bookNo;

     for (int i = 0; i < numBooks; i++) {
         if (books[i].bookNo == bookNo) {
             if (books[i].isIssued) {
                 cout << "Book is already issued.\n";
                 return;
             }
             students[userIndex].borrowedBook = bookNo;
             books[i].isIssued = true;
             cout << "Book issued successfully.\n";
             return;
         }
     }

     cout << "Book not found.\n";
}
```

- Returns a borrowed book.
- Prompts the user to enter their ID.
- Searches for the student in the array.
- Checks if the student has borrowed a book.
- Searches for the borrowed book in the book array.
- If the book is found, unsets the isIssued flag of the book and clears the borrowedBook field of the student.

```
void returnBook(User users[], int numStudents, Book books[], int&
numBooks) {
    string userID;
    cout << "Enter user's ID: ";
    cin >> userID;

    int userIndex = searchUser(students, numStudents, userID);
    if (userIndex == -1) {
        cout << "User not found.\n";
        return;
    }

    if (students[userIndex].borrowedBook.empty()) {
        cout << "User has not borrowed any book.\n";
        return;
    }
    string bookNo = students[userIndex].borrowedBook;

    for (int i = 0; i < numBooks; i++) {
        if (books[i].bookNo == bookNo) {
            books[i].isIssued = false;
            students[userIndex].borrowedBook = "";
            cout << "Book returned successfully.\n";
            return;
        }
    }

    cout << "Book not found in the library.\n";
}
```

- Adds a new book to the book array.
- Prompts the user to enter the book number, name, and author.
- Sets the isIssued flag to false.
- Increments the numBooks counter.

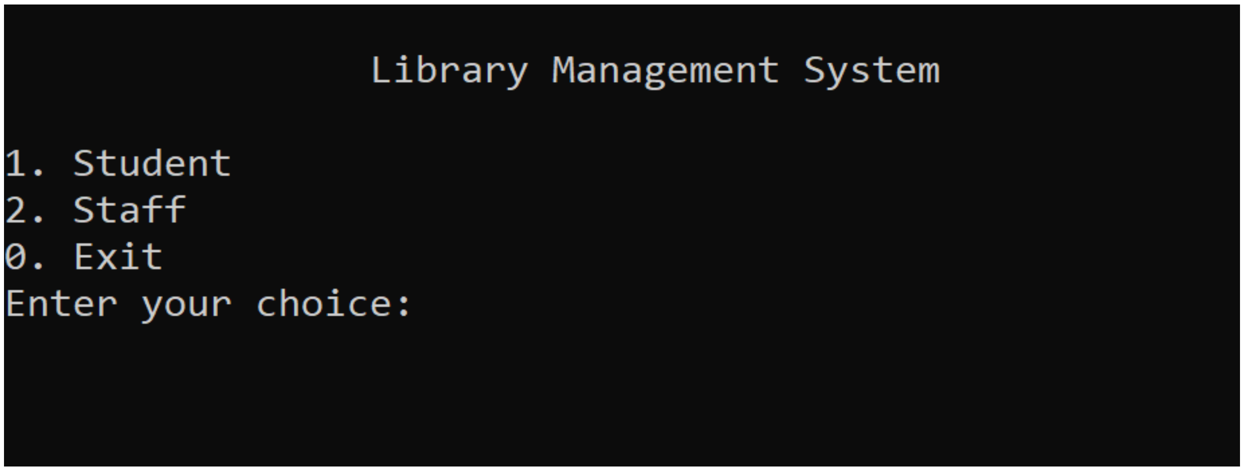
```
void addBook(Book books[], int& numBooks) {
    if (numBooks >= MAX_BOOKS) {
        cout << "Maximum number of books reached.\n";
        return;
    }

    cout << "\nNEW BOOK ENTRY...\n";
    cout << "Enter the book number: ";
    cin >> books[numBooks].bookNo;
    cout << "Enter the book name: ";
    cin.ignore();
    getline(cin, books[numBooks].name);
    cout << "Enter the author's name: ";
    getline(cin, books[numBooks].author);
    books[numBooks].isIssued = false;

    numBooks++;
    cout << "\nBook record created.\n";
}
```

- Searches for a book in the array based on its ID.
- Takes the book array, the number of books, and the book ID as input.
- Returns the index of the book in the array if found, otherwise returns -1.

```
int searchBook(const Book books[], int numBooks, const string&
bookNo) {
    for (int i = 0; i < numBooks; i++) {
        if (books[i].bookNo == bookNo) {
            return i;
        }
    }
    return -1;
}
```

5.4 USER INTERFACE

- The program presents a main menu with options for "Student" and "Staff."
- Each option leads to a submenu with specific actions:
- Student Menu: Add Student, Display Students, Issue Book, Return Book.
- Staff Menu: Add Staff, Display Staff, Add Book, Search Book.
- The program provides clear prompts and informative messages to guide the user.

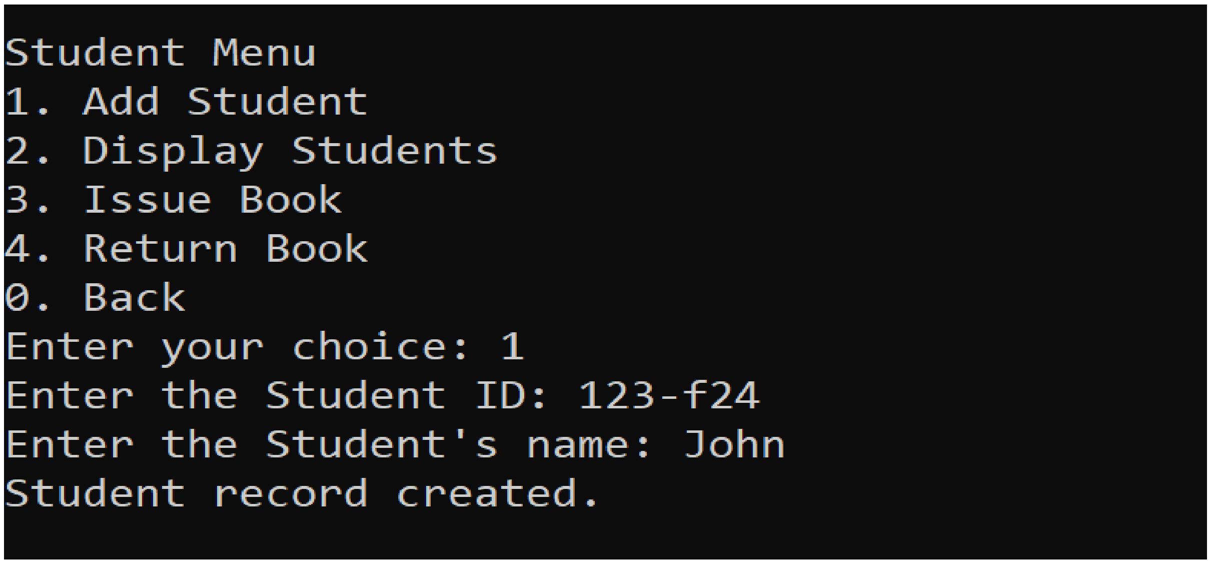


```
                Library Management System

1. Student
2. Staff
0. Exit
Enter your choice:
```

5.5 OUTPUT

5.5.1 ADDING A STUDENT

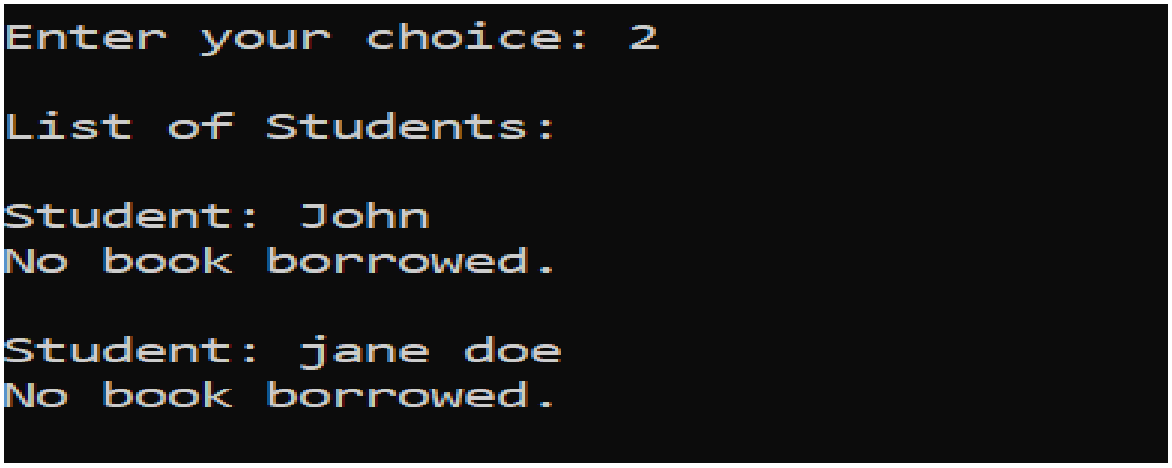
- Enter the Student ID: S123
- Enter the Student's name: John Doe
- Student record created.



```
Student Menu
1. Add Student
2. Display Students
3. Issue Book
4. Return Book
0. Back
Enter your choice: 1
Enter the Student ID: 123-f24
Enter the Student's name: John
Student record created.
```

5.5.2 DISPLAYING STUDENTS

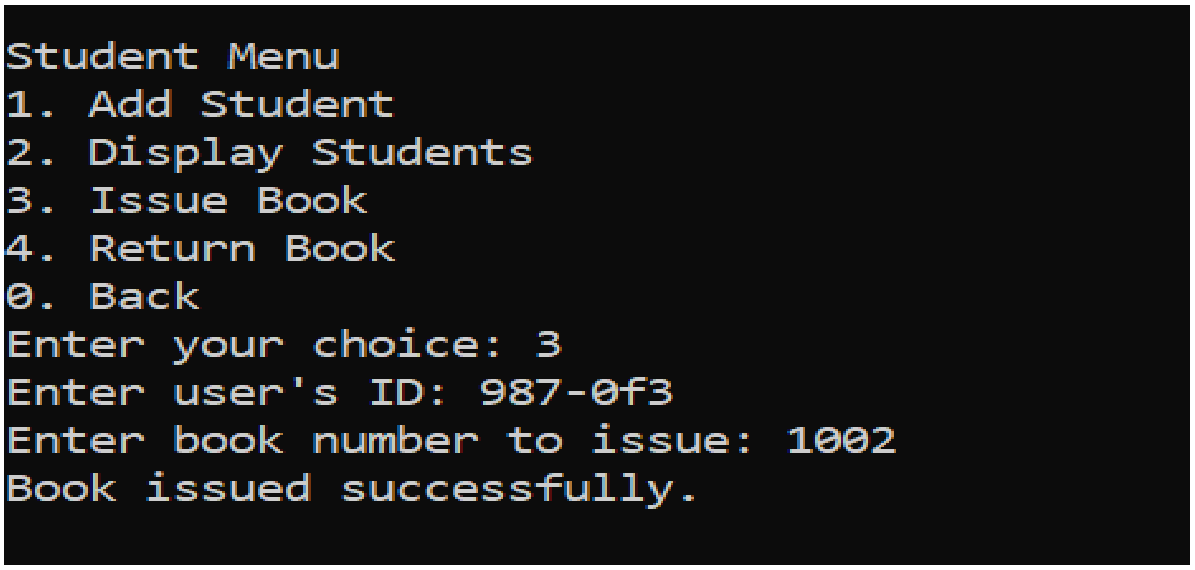
- List of Students:
- Student: John Doe
- Borrowed Book: 1004
- Student: Jane Doe
- Borrowed Book: B001



```
Enter your choice: 2

List of Students:

Student: John
No book borrowed.

Student: jane doe
No book borrowed.
```

5.2.3 ISSUING A BOOK

- Enter user's ID: S123
- Enter book number to issue: B002
- Book issued successfully.



```
Student Menu
1. Add Student
2. Display Students
3. Issue Book
4. Return Book
0. Back
Enter your choice: 3
Enter user's ID: 987-0f3
Enter book number to issue: 1002
Book issued successfully.
```

5.6 EXPLANATION

- The code uses arrays to store the data for students, staff, and books.
- The functions handle user interactions, data manipulation, and system logic.
- The user interface provides a menu-driven approach for easy navigation and interaction.
- The code includes input validation and error handling to ensure data integrity.

CHAPTER 6

CONCLUSION

6. CONCLUSION

This project successfully implements a basic Library Management System using C++. The system effectively automates several library operations, including:

- **User Management:**
  - Adding and displaying student and staff records.
- **Book Management:**
  - Adding new books to the library.
  - Searching for books by their book number.
- **Book Issuance and Return:**
  - Issuing books to students and tracking which books they have borrowed.
  - Returning borrowed books to the library.

6.1 KEY FEATURES:

- **Array-based Data Structures:** Utilizes arrays to efficiently store and manage student, staff, and book data.
- **Modular Design:** The system is organized into well-defined modules for better code organization and maintainability.
- **User-friendly Interface**: Provides a simple and intuitive command-line interface with clear prompts and messages.
- **Core Library Functions:** Includes essential functions for adding, displaying, searching, and managing users and books.

6.2 LIMITATIONS AND FUTURE ENHANCEMENTS:

- **Data Persistence:** Currently, the data is stored in memory. Implementing file I/O capabilities would allow the system to save and retrieve data between sessions, making it more robust and practical for real-world use.
- **User Authentication:** Adding user authentication and authorization features would enhance security and restrict access to certain functionalities based on user roles.
- **Advanced Search**: Implementing more advanced search functionalities, such as searching by book title, author, or keywords, would improve user experience.

- **Overdue Book Handling:** Incorporating features to track overdue books and generate reminders for users would improve library operations.
- **Graphical User Interface (GUI):** Developing a GUI using a library like SFML or Qt would make the system more user-friendly and visually appealing.

Overall, this project serves as a foundation for a more comprehensive library management system. By addressing the mentioned limitations and implementing further enhancements, the system can be expanded to better meet the needs of a real-world library.