



NATIONAL ENGINEERING COLLEGE, K.R.NAGAR, KOVILPATTI - 628 503

(An Autonomous Institution – Affiliated to Anna University, Chennai)

Department of Computer Science and Engineering

23CS24C – OBJECT ORIENTED PROGRAMMING USING C++

Experiential Learning

A Report on

Complex Library Management System

Team Members:

Suruthika J - 2312040

Haritha R - 2312064

Himani Gupta - 2312114

Dharani S -2312116

(I st Year / CSE)

Course Instructor

Ms.Rathi Pathi

Assistant Professor/CSE

SCENARIO:

You are tasked with developing a complex library management system for a large university. The system should be implemented in C++ and should incorporate various object-oriented programming constructs and principles.

Task: Design and implement the library management system with the following features:

- a. Create a base class 'LibraryItem' that represents an item in the library. It should have attributes like 'title', 'author', 'itemID', and 'availability'. Implement constructors, accessors, and mutators as needed.
- b. Create derived classes Book and AudioCD that inherit from the "LibraryItem" class. These derived classes should have their own unique attributes and methods. For example, a Book should have attributes like 'genre' and 'pages', while an AudioCD should have attributes like 'duration' and 'artist'.
- c. Implement a generic structure to store library items (e.g., using templates) so that it can handle both books and audio CDs. The structure should support adding, removing, and searching for items.
- d. Implement dynamic binding and polymorphism by defining virtual functions in the base class and overriding them in the derived classes. For example, create a virtual function 'displayDetails()' that displays item-specific details.
- e. Implement exception handling to manage runtime errors. For instance, handle exceptions when trying to remove a non-existent item from the collection or when a user attempts to borrow an already borrowed item.
- f. Implement a file handling system that allows you to read and write library item data to/from files. Create functions for saving the library's data to a file and loading data from a file.

Ensure that your program demonstrates the effective use of C++ features, adheres to object-oriented programming principles, and meets the specified requirements. You can add your own additional class, methods, Templates wherever needed Your code should be well- structured, maintainable, capable of handling real-world scenarios and error-tolerant

INTRODUCTION:

In a large university, managing a vast collection of library resources efficiently is crucial for ensuring that students and faculty can access the materials they need for their academic and research activities. Traditional methods of managing library items, such as books and audio CDs, often lead to inefficiencies and inaccuracies. To address these challenges, a comprehensive library management system is essential.

In the evolving landscape of educational institutions, efficient management of resources is paramount. A pivotal component of any academic institution is its library—a repository of knowledge that requires meticulous organization and accessibility. To address these needs, we propose the development of a comprehensive Complex Library Management System needed for a large university.

Designing a library management system for a large university involves creating a sophisticated C++ software solution to manage various library items, such as books and audio CDs. The system must utilize object-oriented programming principles to ensure modularity, extensibility, and maintainability. This can be achieved through the use of OOP concepts such as Inheritance, Polymorphism, Abstraction, and Encapsulation.

To address these challenges, this project proposes the development of a sophisticated library management system. The design includes the creation of a base class `LibraryItem` and derived classes `Book` and `AudioCD`, each encapsulating specific attributes and behaviors. By employing OOP principles, therefore this introduction outlines the scope, key functionalities, and technological foundation of the Library Management System, emphasizing its importance in modern educational environments.

PROBLEM STATEMENT:

Implementing a C++ library management system for a university using OOP principles. The system should manage books and audio CDs with features for adding, removing, searching, polymorphism, and file handling. Ensure error handling and real-world scenario management. The system should be scalable to handle a large number of library items efficiently. Ensure compatibility with various operating systems and file systems for the file handling functionalities. By fulfilling these requirements, the Library Management System will enable efficient management of library resources, ensuring a robust and user-friendly experience for the university's staff and students.

OBJECTIVES:

- To make the library's operations run efficiently, benefiting both the staff and students who use the library.
- To implement functions to efficiently add and remove books and audio CDs from the library collection.
- To give transparency in library operations by implementing secured processes for managing fines, fees, and patron information within the library Management System.
- To enable file handling and exception management for data integrity and error handling.
- To ensure maintainability and real world applicability and design the system to be well-structured and maintainable and capable of handling real-world scenarios.

USE CASE DIAGRAM :

A use case diagram shows how users (student, librarian) interact with a system by illustrating the specific tasks or actions the system can perform in response to user inputs or events. Here, there are many connections between librarian and users of the library.

Oval shapes represent functionalities or services provided by the system to its actors. Each use case describes a specific way the system interacts with its users (actors) to achieve a particular goal.

The below figure (Fig.1) represents the use case diagram of the Complex Library Management system

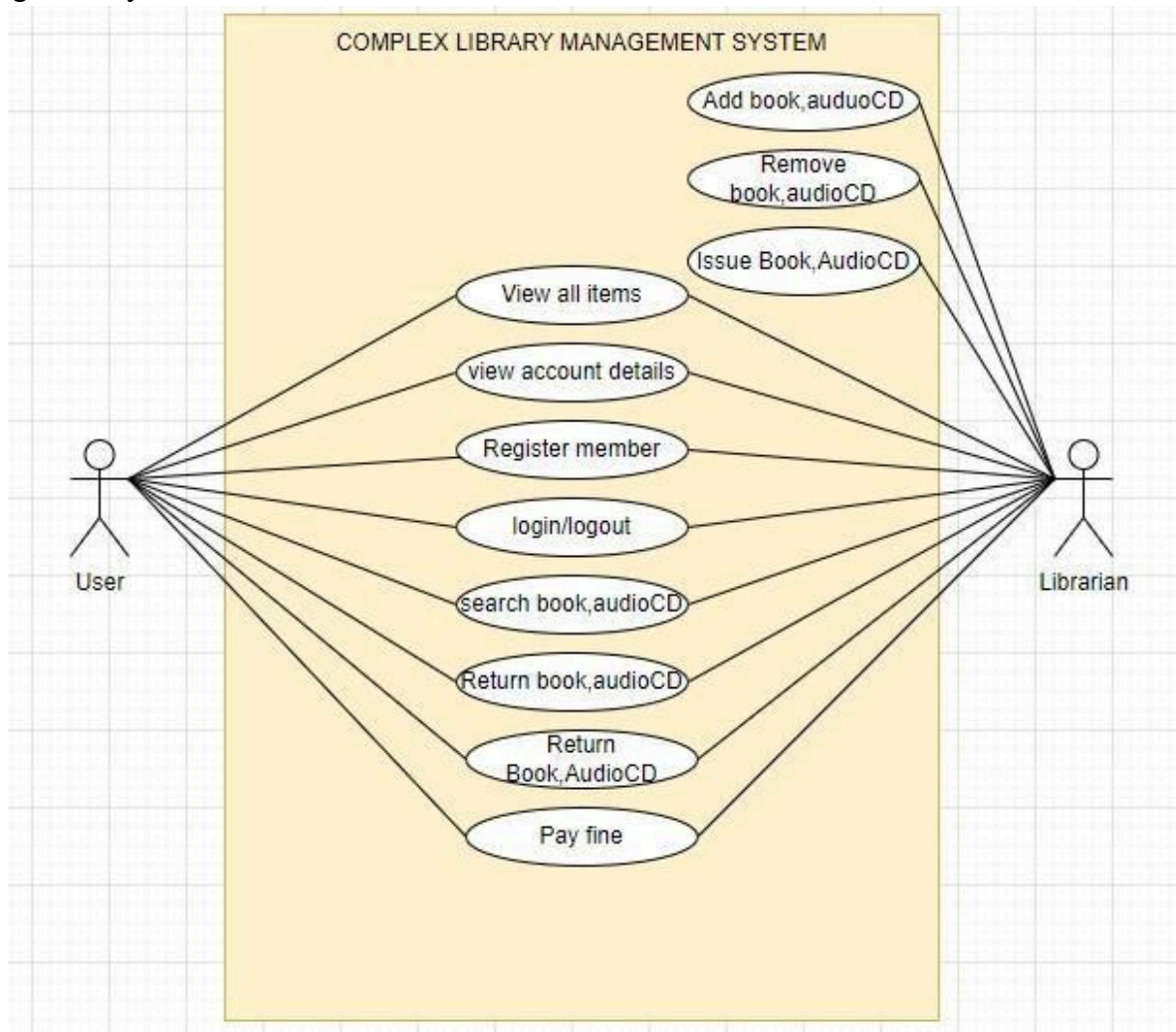


Fig.1 - Use case diagram - Complex Library Management system

CLASS DIAGRAM

A class diagram is a type of UML diagram that illustrates the structure of a system by showing its classes, attributes, operations, and relationships between objects. It provides a blueprint for designing and understanding object-oriented software systems. It will have the access specifier's name within itself.

Private (-): Denotes that the member is accessible only within its own class.

Public (+): Denotes that the member is accessible from outside the class.

The below figure (Fig.2) represents the class diagram of the Complex Library Management system

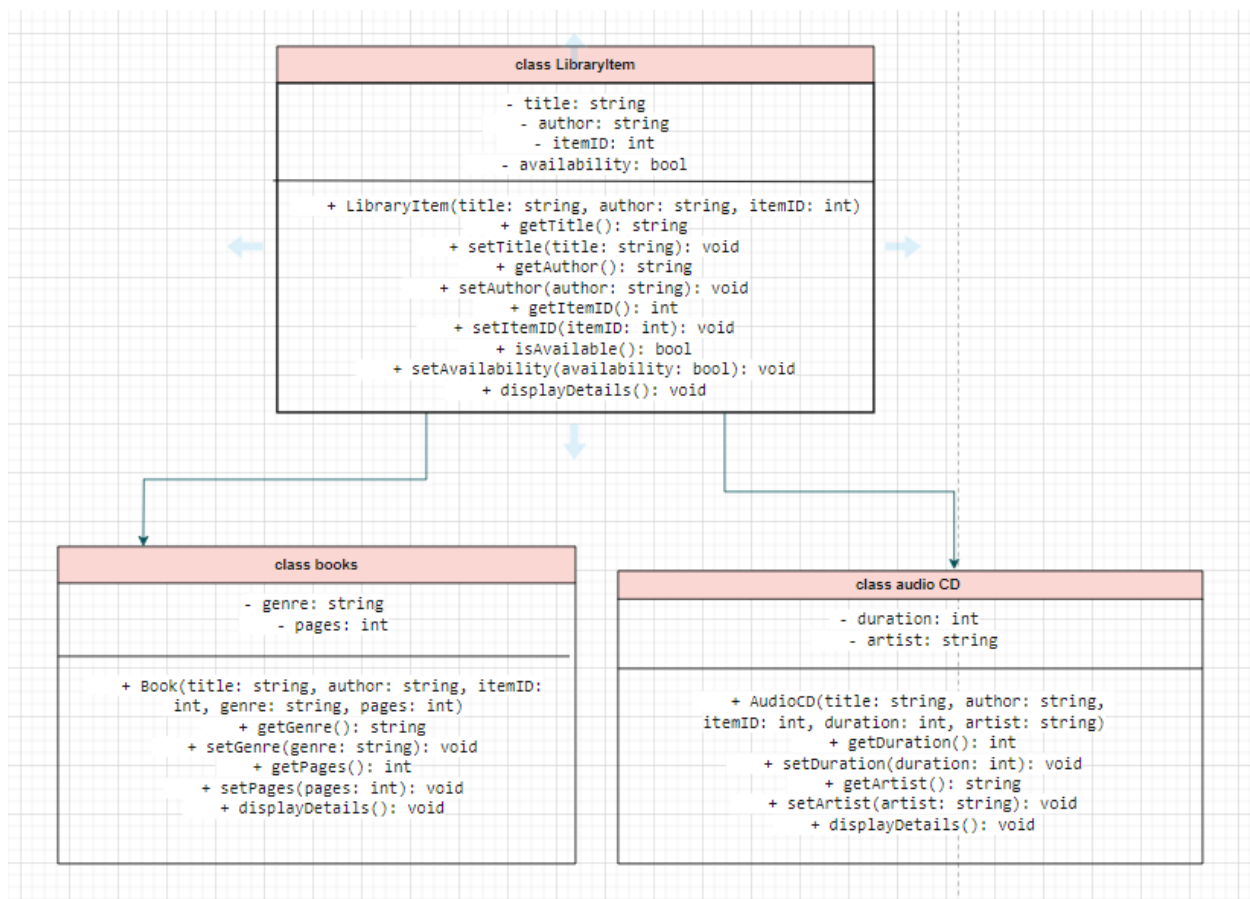


Fig . 2 - Class diagram - Complex Library Management system

OBJECT ORIENTED CONCEPTS:

- Inheritance
- Polymorphism
- Encapsulation (It also provides Abstraction)

INHERITANCE:

Inheritance allows a class(LibraryItem) to inherit attributes and methods from another class(Book and AudioCD), promoting code reusability and hierarchy among classes.

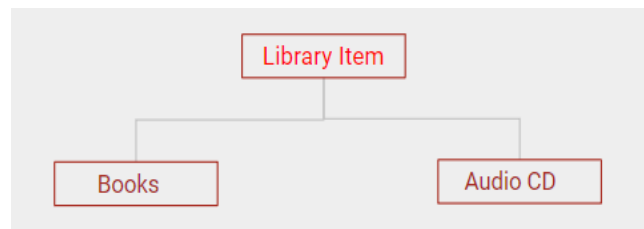


Fig. 3 - Hierarchical Inheritance

Inheritance Type : **Hierarchical inheritance**

Base class: 'LibraryItem'

Derived classes: Book and AudioCD

REASON FOR USING IT IN OUR CODE :

Code Reuse:

By using inheritance, common attributes and functionalities can be defined in a base class (e.g., LibraryItem), and then derived classes (e.g., Book and AudioCD under LibraryItem) can inherit these common characteristics. This promotes code reuse and reduces redundancy.

Easy Updates:

If there's a change in how a libraryItem(base class), and all items that inherit from it (derived class : Book and AudioCD) automatically reflect that change.

Different Types, Same Actions:

Inheritance allows using the same actions (like borrowing and returning) on different library items even though they may have different details (like a book and an audio CD).

POLYMORPHISM:

Type of polymorphism used here: Runtime polymorphism.

- “Run time polymorphism” is implemented through virtual function and override.
- Polymorphism allows objects of different classes to be treated as objects of a common superclass, enabling methods to perform differently based on the actual object type. It supports flexibility and reusability .

REASON FOR USING IT IN OUR CODE

Code Reusability:

By defining common behavior in the base class (LibraryItem), you avoid duplicating code across different derived classes (Book, AudioCD). **Virtual functions** allow you to reuse the same function names and logic across related classes. - **Run time polymorphism**

Flexibility:

We can treat different types of library items (Book, AudioCD, and potentially others like DVD in the future) uniformly through a common interface (LibraryItem). This simplifies code maintenance and supports adding new item types without modifying existing code.

ENCAPSULATION: (hiding unwanted and showing essentials) :

Encapsulation is achieved through the use of classes, private and protected member variables, and public member functions.

Classes:LibraryItem, Book, AudioCD

Private Members: `genre`, `pages`, `duration`, `artist` in `Book` and `AudioCD` classes.

Protected Members: `title`, `author`, `itemID`, `availability` in the `LibraryItem` class.

Public Member Functions: These functions provide an interface for accessing and modifying the private/protected data members (e.g., `getTitle()`, `getAuthor()`, `setAvailability()`, `displayDetails()`).

ABSTRACTION:

Encapsulation provides a level of abstraction by exposing only the necessary details through the public interface.

Users of the `LibraryItem` class do **not need to know how the data is stored** or managed internally; they just need to know how to interact with it via the provided member functions.

FILE CONCEPTS :

INPUT FILES:

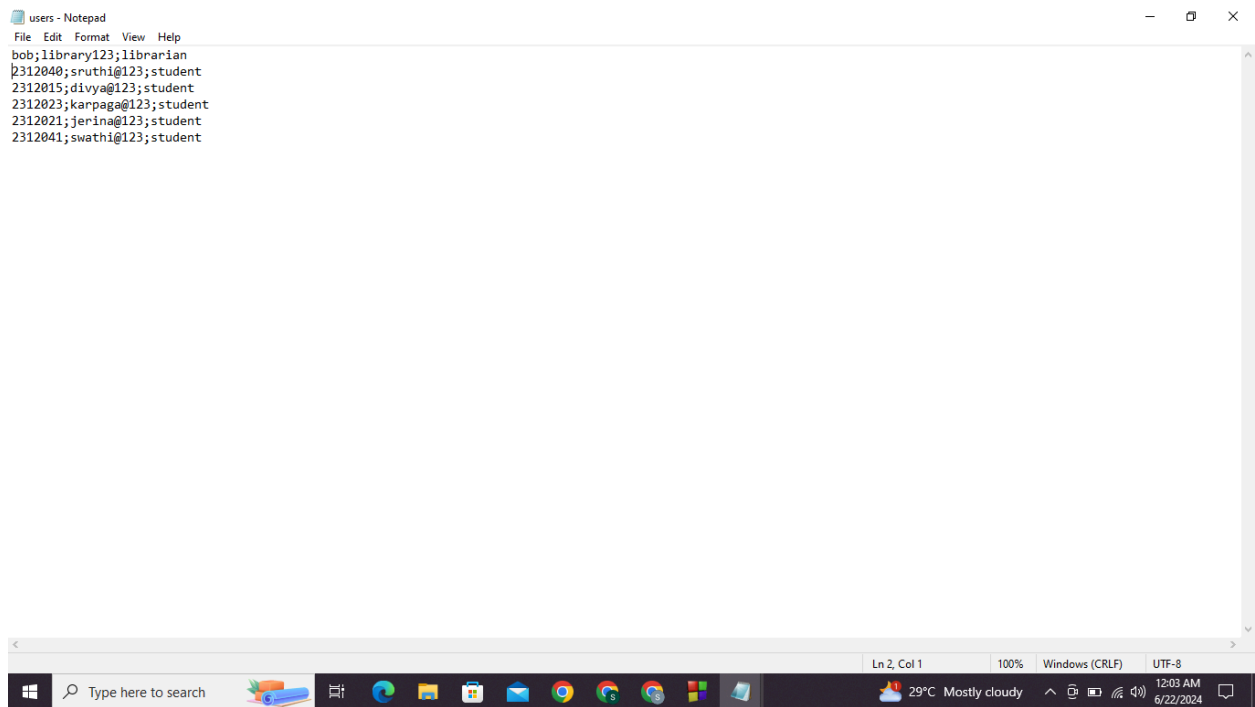
Files that we used here are ,

users.txt

books.txt

audioCDs.txt

CONTENTS OF users.txt :

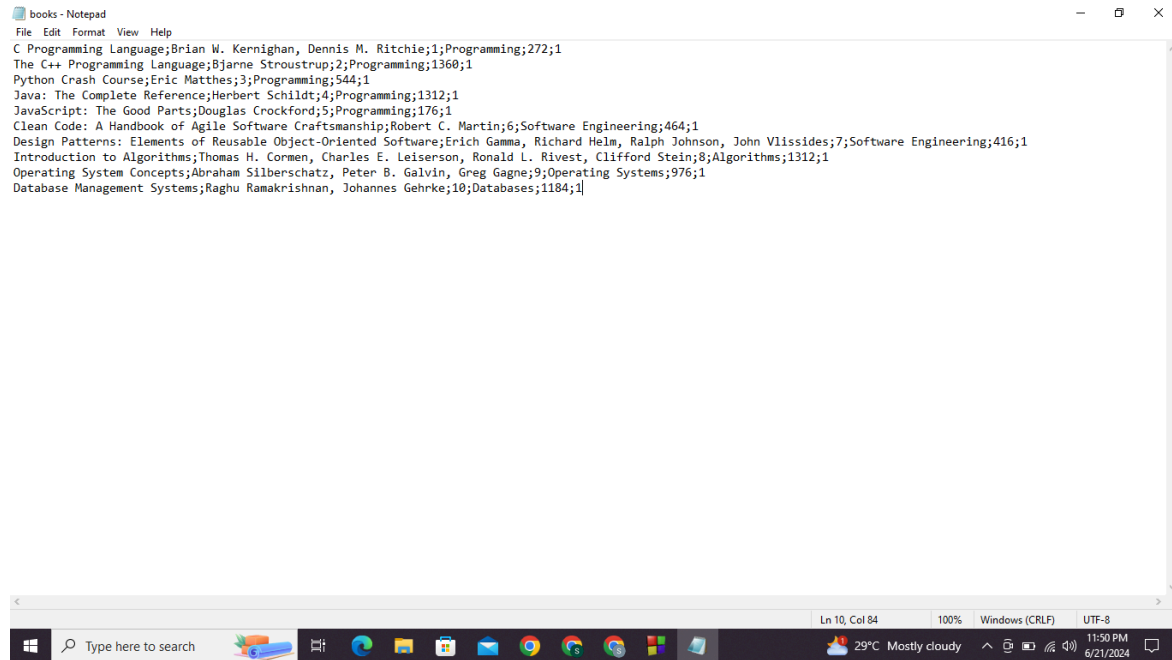


```
users - Notepad
File Edit Format View Help
bob;library123;librarian
2312040;sruthi@123;student
2312015;divya@123;student
2312023;karpaga@123;student
2312021;jerina@123;student
2312041;swathi@123;student
```

contents of “ users.txt ” in text format:

```
bob;library123;librarian
2312040;sruthi@123;student
2312015;divya@123;student
2312023;karpaga@123;student
2312021;jerina@123;student
2312041;swathi@123;student
```

CONTENTS OF books.txt :

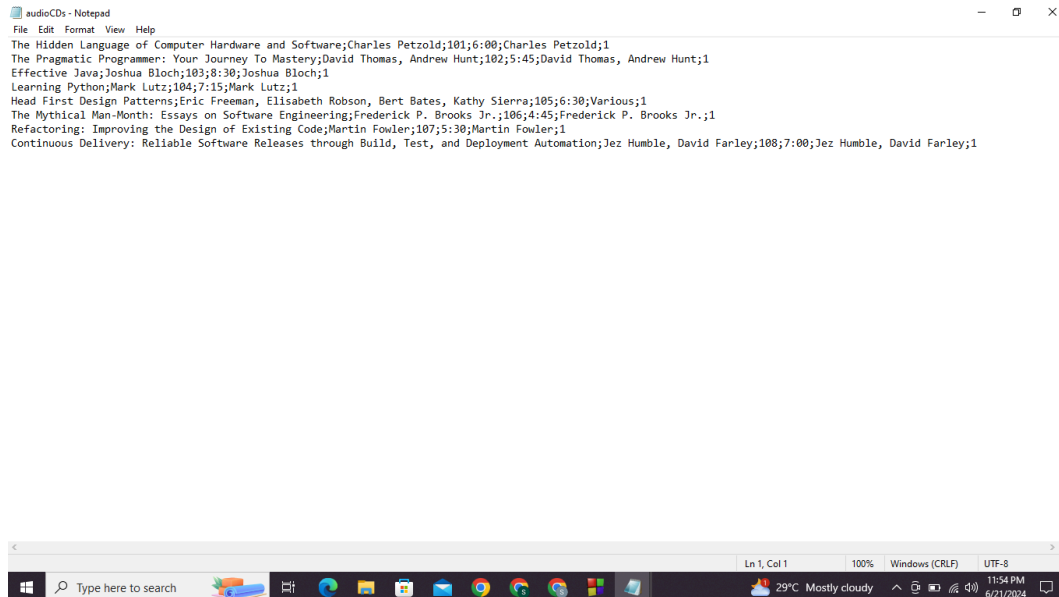


```
books - Notepad
File Edit Format View Help
C Programming Language;Brian W. Kernighan, Dennis M. Ritchie;1;Programming;272;1
The C++ Programming Language;Bjarne Stroustrup;2;Programming;1360;1
Python Crash Course;Eric Matthes;3;Programming;544;1
Java: The Complete Reference;Herbert Schildt;4;Programming;1312;1
JavaScript: The Good Parts;Douglas Crockford;5;Programming;176;1
Clean Code: A Handbook of Agile Software Craftsmanship;Robert C. Martin;6;Software Engineering;464;1
Design Patterns: Elements of Reusable Object-Oriented Software;Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides;7;Software Engineering;416;1
Introduction to Algorithms;Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein;8;Algorithms;1312;1
Operating System Concepts;Abraham Silberschatz, Peter B. Galvin, Greg Gagne;9;Operating Systems;976;1
Database Management Systems;Raghu Ramakrishnan, Johannes Gehrke;10;Databases;1184;1
```

contents of “ books.txt ” in text format:

C Programming Language;Brian W. Kernighan, Dennis M. Ritchie;1;Programming;272;1
The C++ Programming Language;Bjarne Stroustrup;2;Programming;1360;1
Python Crash Course;Eric Matthes;3;Programming;544;1
Java: The Complete Reference;Herbert Schildt;4;Programming;1312;1
JavaScript: The Good Parts;Douglas Crockford;5;Programming;176;1
Clean Code: A Handbook of Agile Software Craftsmanship;Robert C. Martin;6;Software Engineering;464;1
Design Patterns: Elements of Reusable Object-Oriented Software;Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides;7;Software Engineering;416;1
Introduction to Algorithms;Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein;8;Algorithms;1312;1
Operating System Concepts;Abraham Silberschatz, Peter B. Galvin, Greg Gagne;9;Operating Systems;976;1
Database Management Systems;Raghu Ramakrishnan, Johannes Gehrke;10;Databases;1184;1

CONTENTS OF AudioCDs.txt :



```
audioCDs - Notepad
File Edit Format View Help
The Hidden Language of Computer Hardware and Software;Charles Petzold;101;6:00;Charles Petzold;1
The Pragmatic Programmer: Your Journey To Mastery;David Thomas, Andrew Hunt;102;5:45;David Thomas, Andrew Hunt;1
Effective Java;Joshua Bloch;103;8:30;Joshua Bloch;1
Learning Python;Mark Lutz;104;7:15;Mark Lutz;1
Head First Design Patterns;Eric Freeman, Elisabeth Robson, Bert Bates, Kathy Sierra;105;6:30;Various;1
The Mythical Man-Month: Essays on Software Engineering;Frederick P. Brooks Jr.;106;4:45;Frederick P. Brooks Jr.;1
Refactoring: Improving the Design of Existing Code;Martin Fowler;107;5:30;Martin Fowler;1
Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation;Jez Humble, David Farley;108;7:00;Jez Humble, David Farley;1
```

contents of “ AudioCDs.txt ” in text format:

The Hidden Language of Computer Hardware and Software;Charles Petzold;101;6:00;Charles Petzold;1
The Pragmatic Programmer: Your Journey To Mastery;David Thomas, Andrew Hunt;102;5:45;David Thomas, Andrew Hunt;1
Effective Java;Joshua Bloch;103;8:30;Joshua Bloch;1
Learning Python;Mark Lutz;104;7:15;Mark Lutz;1
Head First Design Patterns;Eric Freeman, Elisabeth Robson, Bert Bates, Kathy Sierra;105;6:30;Various;1
The Mythical Man-Month: Essays on Software Engineering;Frederick P. Brooks Jr.;106;4:45;Frederick P. Brooks Jr.;1
Refactoring: Improving the Design of Existing Code;Martin Fowler;107;5:30;Martin Fowler;1
Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation;Jez Humble, David Farley;108;7:00;Jez Humble, David Farley;1

EXCEPTION :

- Exceptions for **FILE OPERATIONS:**

When reading from files” (books.txt, audioCDs.txt, users.txt)”, the code uses standard checks such as checking if the file is open “if(file.is_open()“, if the file does not exist , then it will display an error message.

- Exceptions for **USER INPUT VALIDATION:**

When the user enters invalid inputs like entering the wrong password or username, it will throw an error message.

- Exceptions for **ITEM NOT FOUND:**

When attempting to remove or borrow an item from the library, the system checks if the item with the specified itemID exists in the library's collection. If not found, it displays a message indicating that the item was not found.

TEMPLATES:

Templates would be applied to create a generic container or data structure that can store objects of different types (Book and AudioCD in this case).

The template class would allow flexibility in storing and manipulating different types of library items without duplicating code.

Templates comes under generic functions , but we used other generic to add and remove books and audioCDs to the library like

for adding an item ,

```
void addItem(LibraryItem* item)
```

for removing an item ,

```
void removeItem(int itemID)
```

IMPLEMENTATION:

Code:

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <ctime>
#include <string>
#include <algorithm>
using namespace std;

// Base class: LibraryItem

class LibraryItem {
protected:
    string title;
    string author;
    int itemID;
    bool availability;
public:
    LibraryItem(const string& _title, const string& _author, int _itemID, bool _availability = true)
    : title(_title), author(_author), itemID(_itemID), availability(_availability) {}
    virtual ~LibraryItem() {}
    string getTitle() const { return title; }
    string getAuthor() const { return author; }
    int getItemID() const { return itemID; }
    bool isAvailable() const { return availability; }
    void setTitle(const string& _title) { title = _title; }
    void setAuthor(const string& _author) { author = _author; }
    void setItemID(int _itemID) { itemID = _itemID; }
    void setAvailability(bool _availability) { availability = _availability; }
    virtual void displayDetails() const {
        cout << "Title: " << title << endl;
        cout << "Author: " << author << endl;
        cout << "Item ID: " << itemID << endl;
        cout << "Availability: " << (availability ? "Available" : "Not Available") << endl;
    }
    virtual void saveToFile(ofstream& file) const = 0;
    // Pure virtual function(polymorphism) to save to file
};
```

```

class Book : public LibraryItem {                                // Derived class: Book

private:
    string genre;                                                //genre , pages are private members
    int pages;
public:
    Book(const string& _title, const string& _author, int _itemID, const string& _genre, int _pages,
    bool _availability = true)
    : LibraryItem(_title, _author, _itemID, _availability), genre(_genre), pages(_pages) {}

                                // constructor to initialise libraryItem's genre and pages

    void displayDetails() const override                        //override - calls base class
    {
        LibraryItem::displayDetails();
        cout << "Genre: " << genre << endl;
        cout << "Pages: " << pages << endl;
    }
    void saveToFile(ofstream& file) const override {
        file << title << ";" << author << ";" << itemID << ";" << genre << ";" << pages << ";" <<
        availability << endl;
    }
};

                                // Derived class: AudioCD

class AudioCD : public LibraryItem {
private:
    string duration;                                            //duration and artist are private members

    string artist;
public:
    AudioCD(const string& _title, const string& _author, int _itemID, const string& _duration,
    const string& _artist, bool
    _availability = true)
    : LibraryItem(_title, _author, _itemID, _availability), duration(_duration), artist(_artist) {}
    void displayDetails() const override {
        LibraryItem::displayDetails();
        cout << "Duration: " << duration << endl;
        cout << "Artist: " << artist << endl;
    }
    void saveToFile(ofstream& file) const override                //overriding to savetofile

```

```

{
    file << title << ";" << author << ";" << itemID << ";" << duration << ";" << artist << ";" <<
availability << endl;
}
};
// Base class: User
class User {
protected:
    string id;
    string password;
    string role;
    vector<pair<int, time_t>> borrowedItems; // Pair of itemID and expiry date
    double fine;
public:
    User(const string& _id, const string& _password, const string& _role)
    : id(_id), password(_password), role(_role), fine(0.0) {}
                                // constructor to initialise role, and password

    virtual ~User() {}
    string getId() const { return id; }
    string getPassword() const { return password; }
    string getRole() const { return role; }
    void addBorrowedItem(int itemID, time_t expiryDate) {
        borrowedItems.push_back({itemID, expiryDate});
    }
    bool isItemBorrowed(int itemID) const {
        return find_if(borrowedItems.begin(), borrowedItems.end(), [itemID](const pair<int, time_t>&
p) { return p.first ==
itemID; }) != borrowedItems.end();
    }
    time_t getExpiryDate(int itemID) const {
        for (const auto& item : borrowedItems) {
            if (item.first == itemID) {
                return item.second;
            }
        }
        return 0;
    }
    void returnItem(int itemID) {
        borrowedItems.erase(remove_if(borrowedItems.begin(), borrowedItems.end(),
[itemID](const pair<int, time_t>& p) { return p.first == itemID; }), borrowedItems.end());
    }

```

```

}
void addFine(double amount) {
    fine += amount;
}
void displayDetails() const {
    cout << "ID: " << id << endl;
    cout << "Role: " << role << endl;
    cout << "Fine: $" << fine << endl;
    cout << "Borrowed Items:" << endl;
    for (const auto& item : borrowedItems) {
        cout << "Item ID: " << item.first << ", Expiry Date: " << ctime(&item.second);
    }
}
};
// Class: Library
class Library {
private:
    vector<LibraryItem*> items;
public:
    Library() {
        loadItems();
    }
    ~Library() {
        saveItems();
        for (auto& item : items) {
            delete item;
        }
    }
    void loadItems() {
        ifstream bookFile("books.txt");
        ifstream audioCDFile("audioCDs.txt");
        string line;
        while (getline(bookFile, line)) {
            istringstream iss(line);
            string title, author, genre, availabilityStr;
            int itemID, pages;
            bool availability;
            char delim;
            if (getline(iss, title, ';') && getline(iss, author, ';') && (iss >> itemID >> delim) && getline(iss,
                genre, ';') && (iss >>

```



```

pages >> delim) && (iss >> availability)) {
    items.push_back(new Book(title, author, itemID, genre, pages, availability));
}
}
while (getline(audioCDFile, line)) {
    istringstream iss(line);
    string title, author, duration, artist, availabilityStr;
    int itemID;
    bool availability;
    char delim;
    if (getline(iss, title, ';') && getline(iss, author, ';') && (iss >> itemID >> delim) && getline(iss,
duration, ';') &&
getline(iss, artist, ';') && (iss >> availability)) {
        items.push_back(new AudioCD(title, author, itemID, duration, artist, availability));
    }
}
bookFile.close();
audioCDFile.close();
}
void saveItems() {
    ofstream bookFile("books.txt");
    ofstream audioCDFile("audioCDs.txt");
    for (const auto& item : items) {
        if (dynamic_cast<Book*>(item)) {
            item->saveToFile(bookFile);
        } else if (dynamic_cast<AudioCD*>(item)) {
            item->saveToFile(audioCDFile);
        }
    }
}
bookFile.close();
audioCDFile.close();
}
void addItem(LibraryItem* item) {
    items.push_back(item);
    saveItems(); // Save to file immediately after adding
}
void removeItem(int itemID) {
    for (auto it = items.begin(); it != items.end(); ++it) {
        if ((*it)->getItemID() == itemID) {
            delete *it;

```

```

items.erase(it);
cout << "Item with ID " << itemID << " removed from the library." << endl;
saveItems(); // Save to file immediately after removing
return;
}
}
cout << "Item with ID " << itemID << " not found in the library." << endl;
}
LibraryItem* getItemById(int itemID) {
for (auto& item : items) {
if (item->getItemID() == itemID) {
return item;
}
}
return nullptr;
}
void searchItemByTitle(const string& title) const {
bool found = false;
for (const auto& item : items) {
if (item->getTitle() == title) {
item->displayDetails();
found = true;
cout << "-----" << endl;
}
}
if (!found) {
cout << "No items found with the title \"" << title << "\"" << endl;
}
}
void displayAllItems() const {
if (items.empty()) {
cout << "No items in the library." << endl;
} else {
for (const auto& item : items) {
item->displayDetails();
cout << "-----" << endl;
}
}
}
};

```

```

// Function to get current date plus given days
time_t getExpiryDate(int days) {
    time_t now = time(nullptr);
    return now + days * 24 * 60 * 60;
}

// Function to calculate fine if any
double calculateFine(time_t expiryDate) {
    time_t now = time(nullptr);
    double finePerDay = 1.0; // Example fine per day
    if (now > expiryDate) {
        double daysLate = difftime(now, expiryDate) / (24 * 60 * 60);
        return daysLate * finePerDay;
    }
    return 0.0;
}

// Function to load users from file
void loadUsers(vector<User*>& users) {
    ifstream userFile("users.txt");
    string line;
    while (getline(userFile, line)) {
        istringstream iss(line);
        string id, password, role;
        if (getline(iss, id, ';') && getline(iss, password, ';') && getline(iss, role)) {
            users.push_back(new User(id, password, role));
        }
    }
    userFile.close();
}

// Function to save users to file
void saveUsers(const vector<User*>& users) {
    ofstream userFile("users.txt");
    for (const auto& user : users) {
        userFile << user->getId() << ";" << user->getPassword() << ";" << user->getRole() << endl;
    }
    userFile.close();
}

// Function to register a new user
void registerUser(vector<User*>& users) {
    string id, password, role;
    cout << "Enter new user ID: ";

```

```

cin >> id;
cout << "Enter new user password: ";
cin >> password;
cout << "Enter role (student/librarian): ";
cin >> role;

// Check if user already exists
for (const auto& user : users) {
    if (user->getId() == id) {
        cout << "User ID already exists. Registration failed." << endl;
        return;
    }
}
users.push_back(new User(id, password, role));
saveUsers(users);
cout << "User registered successfully!" << endl;
}

// Function to login
User* login(const vector<User*>& users) {
    string id, password;
    cout << "Enter user ID: ";
    cin >> id;
    cout << "Enter password: ";
    cin >> password;
    for (const auto& user : users) {
        if (user->getId() == id && user->getPassword() == password) {
            cout << "Login successful!" << endl;
            return user;
        }
    }
    cout << "Login failed. Incorrect ID or password." << endl;
    return nullptr;
}

// Main menu function
void mainMenu(User* loggedInUser, Library& library) {
    int choice;
    do {
        cout << "\n--- Main Menu ---\n";
        cout << "1. View all items\n";
        cout << "2. Search item by title\n";
        cout << "3. Borrow item\n";
        //To be displayed
    } while (choice != 0);
}

```

```

cout << "4. Return item\n";
cout << "5. Add new item (Librarians only)\n";
cout << "6. Remove item (Librarians only)\n";
cout << "7. View account details\n";
cout << "8. Logout\n";
cout << "Enter your choice: ";
cin >> choice;
switch (choice) {
case 1:
library.displayAllItems();
break;
case 2: {
string title;
cout << "Enter item title: ";
cin.ignore();
getline(cin, title);
library.searchItemByTitle(title);
break;
}
case 3: {
int itemID;
cout << "Enter item ID to borrow: ";
cin >> itemID;
LibraryItem* item = library.getItemById(itemID);
if (item && item->isAvailable()) {
item->setAvailability(false);
time_t expiryDate = getExpiryDate(14); // Borrow period of 14 days
loggedInUser->addBorrowedItem(itemID, expiryDate);
cout << "Item borrowed successfully! Return by: " << ctime(&expiryDate);
} else {
cout << "Item is not available or does not exist." << endl;
}
break;
}
case 4: {
int itemID;
cout << "Enter item ID to return: ";
cin >> itemID;
if (loggedInUser->isItemBorrowed(itemID)) {
time_t expiryDate = loggedInUser->getExpiryDate(itemID);

```

```

double fine = calculateFine(expiryDate);
if (fine > 0) {
    cout << "Item returned late. Fine: $" << fine << endl;
    loggedInUser->addFine(fine);
} else {
    cout << "Item returned on time. No fine." << endl;
}
library.getItemById(itemID)->setAvailability(true);
loggedInUser->returnItem(itemID);
} else {
    cout << "Item not borrowed by you or does not exist." << endl;
}
break;
}
case 5:
    if (loggedInUser->getRole() == "librarian") {
        int itemType;
        cout << "Add new item:\n1. Book\n2. Audio CD\nEnter your choice: ";
        cin >> itemType;
        if (itemType == 1) {
            string title, author, genre;
            int itemID, pages;
            cout << "Enter title: ";
            cin.ignore();
            getline(cin, title);
            cout << "Enter author: ";
            getline(cin, author);
            cout << "Enter item ID: ";
            cin >> itemID;
            cout << "Enter genre: ";
            cin.ignore();
            getline(cin, genre);
            cout << "Enter pages: ";
            cin >> pages;
            library.addItem(new Book(title, author, itemID, genre, pages));
            cout << "Book added successfully!" << endl;
        } else if (itemType == 2) {
            string title, author, duration, artist;
            int itemID;
            cout << "Enter title: ";

```

```

cin.ignore();
getline(cin, title);
cout << "Enter author: ";
getline(cin, author);
cout << "Enter item ID: ";
cin >> itemID;
cout << "Enter duration: ";
cin.ignore();
getline(cin, duration);
cout << "Enter artist: ";
getline(cin, artist);
library.addItem(new AudioCD(title, author, itemID, duration, artist));
cout << "Audio CD added successfully!" << endl;
} else {
cout << "Invalid choice." << endl;
}
} else {
cout << "Permission denied. Only librarians can add items." << endl;
}
break;
case 6:
if (loggedInUser->getRole() == "librarian") {
int itemID;
cout << "Enter item ID to remove: ";
cin >> itemID;
library.removeItem(itemID);
} else {
cout << "Permission denied. Only librarians can remove items." << endl;
}
break;
case 7:
loggedInUser->displayDetails();
break;
case 8:
cout << "Logging out...\n";
break;
default:
cout << "Invalid choice. Please try again.\n";
}
} while (choice != 8);

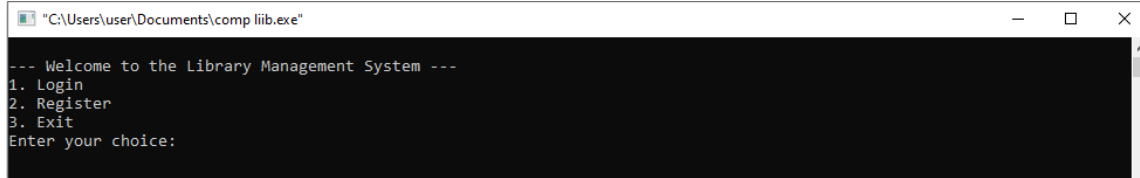
```

```

}
int main() {
    vector<User*> users;
    Library library;
    loadUsers(users);
    int choice;
    do {
        cout << "\n--- Welcome to the Library Management System ---\n";
        cout << "1. Login\n";
        cout << "2. Register\n";
        cout << "3. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice) {
            case 1: {
                User* loggedInUser = login(users);
                if (loggedInUser) {
                    mainMenu(loggedInUser, library);
                }
                break;
            }
            case 2:
                registerUser(users);
                break;
            case 3:
                cout << "Exiting the system. Goodbye!\n";
                break;
            default:
                cout << "Invalid choice. Please try again.\n";
        }
    } while (choice != 3);
    saveUsers(users);
    for (auto user : users) {
        delete user;
    }
    return 0;
}

```


OUTPUT SCREENSHOTS:



```
"C:\Users\user\Documents\comp liib.exe"

--- Welcome to the Library Management System ---
1. Login
2. Register
3. Exit
Enter your choice:
```

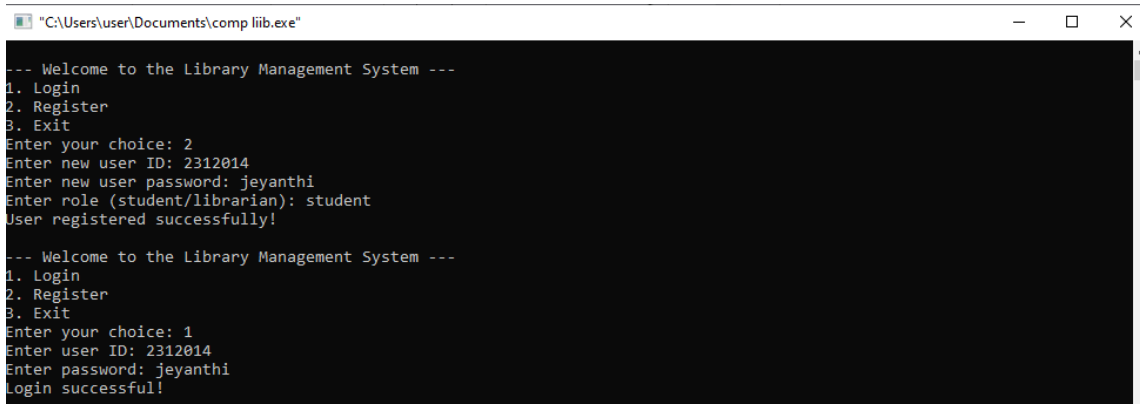
NEW USER REGISTERING:



```
"C:\Users\user\Documents\comp liib.exe"

--- Welcome to the Library Management System ---
1. Login
2. Register
3. Exit
Enter your choice: 2
Enter new user ID: 2312014
Enter new user password: jeyanthi
Enter role (student/librarian): student
User registered successfully!
```

LOGIN AFTER REGISTERING :

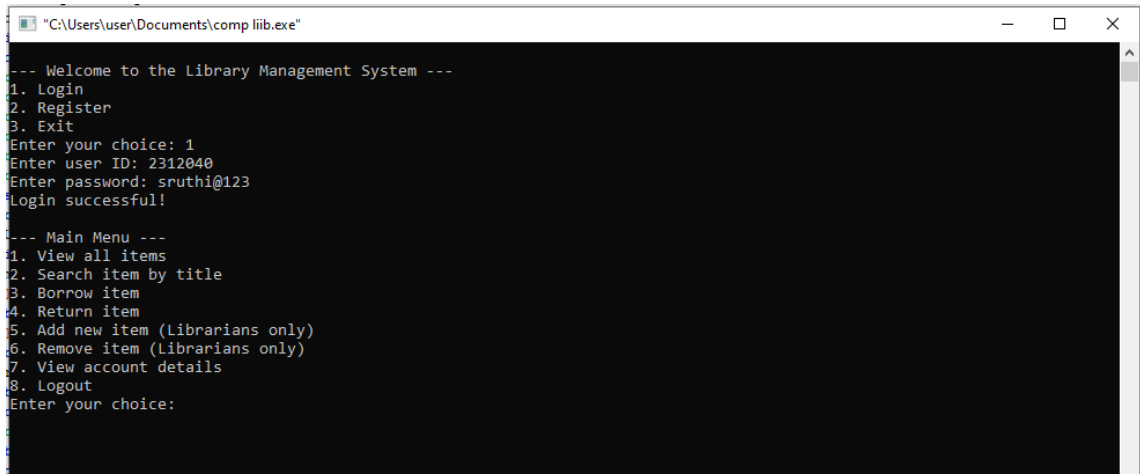


```
"C:\Users\user\Documents\comp liib.exe"

--- Welcome to the Library Management System ---
1. Login
2. Register
3. Exit
Enter your choice: 2
Enter new user ID: 2312014
Enter new user password: jeyanthi
Enter role (student/librarian): student
User registered successfully!

--- Welcome to the Library Management System ---
1. Login
2. Register
3. Exit
Enter your choice: 1
Enter user ID: 2312014
Enter password: jeyanthi
Login successful!
```

LOGGING IN AS AN EXISTING USER:



```
"C:\Users\user\Documents\comp liib.exe"

--- Welcome to the Library Management System ---
1. Login
2. Register
3. Exit
Enter your choice: 1
Enter user ID: 2312040
Enter password: sruthi@123
Login successful!

--- Main Menu ---
1. View all items
2. Search item by title
3. Borrow item
4. Return item
5. Add new item (Librarians only)
6. Remove item (Librarians only)
7. View account details
8. Logout
Enter your choice:
```

ENTERING CHOICE 1 FOR VIEWING ALL ITEMS :

```
"C:\Users\user\Documents\comp liib.exe"

--- Welcome to the Library Management System ---
1. Login
2. Register
3. Exit
Enter your choice: 1
Enter user ID: 2312040
Enter password: sruthi@123
Login successful!

--- Main Menu ---
1. View all items
2. Search item by title
3. Borrow item
4. Return item
5. Add new item (Librarians only)
6. Remove item (Librarians only)
7. View account details
8. Logout
Enter your choice: 1
Title: C Programming Language
Author: Brian W. Kernighan, Dennis M. Ritchie
Item ID: 1
Availability: Available
Genre: Programming
Pages: 272
-----
Title: The C++ Programming Language
Author: Bjarne Stroustrup
Item ID: 2
```

```
"C:\Users\user\Documents\comp liib.exe"

Author: Brian W. Kernighan, Dennis M. Ritchie
Item ID: 1
Availability: Available
Genre: Programming
Pages: 272
-----
Title: The C++ Programming Language
Author: Bjarne Stroustrup
Item ID: 2
Availability: Available
Genre: Programming
Pages: 1360
-----
Title: Python Crash Course
Author: Eric Matthes
Item ID: 3
Availability: Available
Genre: Programming
Pages: 544
-----
Title: Java: The Complete Reference
Author: Herbert Schildt
Item ID: 4
Availability: Available
Genre: Programming
Pages: 1312
-----
Title: JavaScript: The Good Parts
Author: Douglas Crockford
Item ID: 5
```

ENTERING CHOICE 2 TO SEARCH BY TITLE :

```
"C:\Users\user\Documents\comp liib.exe"

--- Main Menu ---
1. View all items
2. Search item by title
3. Borrow item
4. Return item
5. Add new item (Librarians only)
6. Remove item (Librarians only)
7. View account details
8. Logout
Enter your choice: 2
Enter item title: Python Crash Course
Title: Python Crash Course
Author: Eric Matthes
Item ID: 3
Availability: Available
Genre: Programming
Pages: 544
-----
```

ENTERING CHOICE 3 TO BORROW AN ITEM:

```
"C:\Users\user\Documents\comp liib.exe"

Author: Eric Matthes
Item ID: 3
Availability: Available
Genre: Programming
Pages: 544
-----

--- Main Menu ---
1. View all items
2. Search item by title
3. Borrow item
4. Return item
5. Add new item (Librarians only)
6. Remove item (Librarians only)
7. View account details
8. Logout
Enter your choice: 3
Enter item ID to borrow: 2
Item borrowed successfully! Return by: Sat Jul 06 00:47:27 2024
```

ENTERING CHOICE 4 TO RETURN AN ITEM :

```
"C:\Users\user\Documents\comp liib.exe"

6. Remove item (Librarians only)
7. View account details
8. Logout
Enter your choice: 4
Enter item ID to return: 3
Item not borrowed by you or does not exist.

--- Main Menu ---
1. View all items
2. Search item by title
3. Borrow item
4. Return item
5. Add new item (Librarians only)
6. Remove item (Librarians only)
7. View account details
8. Logout
Enter your choice: 4
Enter item ID to return: 2
Item returned on time. No fine.
```

ENTERING CHOICE 5 TO ADD A BOOK AS A LIBRARIAN:

```
"C:\Users\user\Documents\comp liib.exe"

--- Welcome to the Library Management System ---
1. Login
2. Register
3. Exit
Enter your choice: 1
Enter user ID: bob
Enter password: library123
Login successful!

--- Main Menu ---
1. View all items
2. Search item by title
3. Borrow item
4. Return item
5. Add new item (Librarians only)
6. Remove item (Librarians only)
7. View account details
8. Logout
Enter your choice: 5
Add new item:
1. Book
2. Audio CD
Enter your choice: 1
Enter title: OOP with c++
Enter author: E Balaguruswamy
Enter item ID: 231
Enter genre: programming
Enter pages: 350
Book added successfully!
```

books.txt after adding an item :

```
books - Notepad
File Edit Format View Help
C Programming Language;Brian W. Kernighan, Dennis M. Ritchie;1;Programming;272;1
The C++ Programming Language;Bjarne Stroustrup;2;Programming;1360;1
Python Crash Course;Eric Matthes;3;Programming;544;1
Java: The Complete Reference;Herbert Schildt;4;Programming;1312;1
JavaScript: The Good Parts;Douglas Crockford;5;Programming;176;1
Clean Code: A Handbook of Agile Software Craftsmanship;Robert C. Martin;6;Software Engineering;464;1
Design Patterns: Elements of Reusable Object-Oriented Software;Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides;7;Software Engineering;416;1
Introduction to Algorithms;Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein;8;Algorithms;1312;1
Operating System Concepts;Abraham Silberschatz, Peter B. Galvin, Greg Gagne;9;Operating Systems;976;1
Database Management Systems;Raghu Ramakrishnan, Johannes Gehrke;10;Databases;1184;1
OOP with c++;E Balaguruswamy;231;programming;350;1
```

ENTERING CHOICE 6 TO REMOVE AN ITEM :

```
"C:\Users\user\Documents\comp liib.exe"
2. Register
3. Exit
Enter your choice: 1
Enter user ID: bob
Enter password: library123
Login successful!

--- Main Menu ---
1. View all items
2. Search item by title
3. Borrow item
4. Return item
5. Add new item (Librarians only)
6. Remove item (Librarians only)
7. View account details
8. Logout
Enter your choice: 6
Enter item ID to remove: 231
Item with ID 231 removed from the library.
```

content of the file after removing an item:

```
books - Notepad
File Edit Format View Help
C Programming Language;Brian W. Kernighan, Dennis M. Ritchie;1;Programming;272;1
The C++ Programming Language;Bjarne Stroustrup;2;Programming;1360;1
Python Crash Course;Eric Matthes;3;Programming;544;1
Java: The Complete Reference;Herbert Schildt;4;Programming;1312;1
JavaScript: The Good Parts;Douglas Crockford;5;Programming;176;1
Clean Code: A Handbook of Agile Software Craftsmanship;Robert C. Martin;6;Software Engineering;464;1
Design Patterns: Elements of Reusable Object-Oriented Software;Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides;7;Software Engineering;416;1
Introduction to Algorithms;Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein;8;Algorithms;1312;1
Operating System Concepts;Abraham Silberschatz, Peter B. Galvin, Greg Gagne;9;Operating Systems;976;1
Database Management Systems;Raghu Ramakrishnan, Johannes Gehrke;10;Databases;1184;1
```

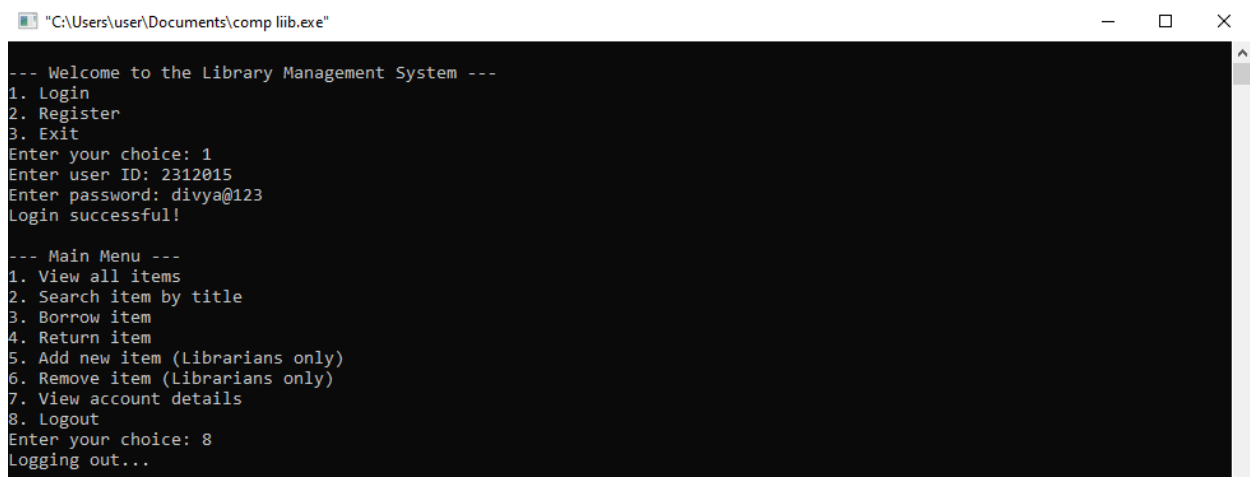
ENTERING CHOICE 7 TO VIEW ACCOUNT DETAILS:



```
"C:\Users\user\Documents\comp liib.exe"
Enter your choice: 1
Enter user ID: 2312041
Enter password: swathi@123
Login successful!

--- Main Menu ---
1. View all items
2. Search item by title
3. Borrow item
4. Return item
5. Add new item (Librarians only)
6. Remove item (Librarians only)
7. View account details
8. Logout
Enter your choice: 7
ID: 2312041
Role: student
Fine: $0
```

ENTERING CHOICE 8 TO LOGOUT:

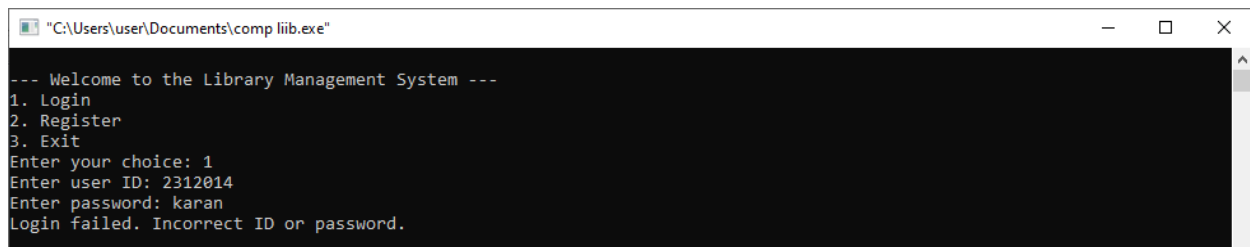


```
"C:\Users\user\Documents\comp liib.exe"
--- Welcome to the Library Management System ---
1. Login
2. Register
3. Exit
Enter your choice: 1
Enter user ID: 2312015
Enter password: divya@123
Login successful!

--- Main Menu ---
1. View all items
2. Search item by title
3. Borrow item
4. Return item
5. Add new item (Librarians only)
6. Remove item (Librarians only)
7. View account details
8. Logout
Enter your choice: 8
Logging out...
```

OUTPUT FOR EXCEPTIONAL CASE:

If we enter invalid password or ID:



```
"C:\Users\user\Documents\comp liib.exe"
--- Welcome to the Library Management System ---
1. Login
2. Register
3. Exit
Enter your choice: 1
Enter user ID: 2312014
Enter password: karan
Login failed. Incorrect ID or password.
```

CONCLUSION:

This comprehensive library management system for a large university, implemented in C++, effectively demonstrates various object-oriented programming principles and constructs. The system includes a base class 'LibraryItem' and derived classes 'Book' and 'AudioCD', which encapsulate the attributes and behaviors of different types of library items. The implementation of templates allows the generic handling of library items, and dynamic binding with virtual functions provides polymorphic behavior, ensuring that specific details of different item types are displayed correctly.

The system includes robust functionality for adding, removing, and searching for items, as well as borrowing and returning items, with appropriate exception handling to manage runtime errors. The inclusion of file handling ensures that library item data and user information are persistently stored and retrieved. User management features, including registration, login, and role-based permissions (e.g., librarians adding or removing items), are well-defined and implemented.

As this library management system is well-structured, maintainable, and capable of handling real-world scenarios. It incorporates essential C++ features and adheres to object-oriented programming principles, making it a reliable and efficient tool for managing a university library's resources.

REFERENCES:

https://www.w3schools.com/cpp/cpp_oop.asp

<https://www.javatpoint.com/cpp-oops-concepts>

<https://www.youtube.com/watch?v=MzPjHEq-MYg>

https://books.google.com/books/about/Object_Oriented_Programming_With_C++.html?id=TN9wQjjDwp0C

<http://libraryopac.bits-hyderabad.ac.in/cgi-bin/koha/opac-detail.pl?biblionumber=1192>

<https://www.youtube.com/watch?v=vXpkpKQUfkW>

<https://www.geeksforgeeks.org/object-oriented-programming-in-cpp/>

https://www.cet.edu.in/noticefiles/285_OOPS%20lecture%20notes%20Complete.pdf

<https://www.geeksforgeeks.org/file-handling-c-classes/>

<https://www.geeksforgeeks.org/file-handling-c-classes/>

<https://www.mygreatlearning.com/blog/file-handling-in-cpp/>