#include <iostream>

#include <string>

#include <map>

#include <algorithm> // for std::find and std::remove

#include <stdlib.h> // for system("CLS")

#include <vector>

using namespace std;

// declare book vectors globally

vector<string> bookTitles;

vector<string> bookAuthors;

vector<int> bookIds;

vector<string> requestedBooks;

vector<string> feedbacks; // vector to store feedbacks

vector<string> holdbooks; // vector to store hold books

int bookCount = 0;

int person = 0;

int tcount = 0;

int hcount = 0;

map<int, float> fines;

// Function to add a book to the library

void addBook()

{

string title, author;

int id;

// Get the book details from the user

cout << "Enter book title: ";

cin >> title;

cout << "Enter book author: ";

cin >> author;

cout << "Enter book ID: ";

cin >> id;

// Add the book to the vectors

bookTitles.push\_back(title);

bookAuthors.push\_back(author);

bookIds.push\_back(id);

bookCount++;

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

}

// function to add books (MAIN)

void add\_books() {

int choice;

do {

// Display the main menu

cout << "\nMain Menu" << endl;

cout << "1. Add a book" << endl;

cout << "2. Exit" << endl;

cout << "Enter your choice (1-2): ";

cin >> choice;

switch (choice) {

case 1:

addBook();

break;

case 2:

cout << "Exiting..." << endl;

break;

default:

cout << "Invalid choice" << endl;

break;

}

} while (choice != 2);{

system("CLS"); // clears the output screen

}

return;

}

void show\_books() {

// Display all books in the database

cout << "\nList of Books:\n";

cout << "----------------\n";

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

cout << "Title: " << bookTitles[i] << "\t" << "Author: " << bookAuthors[i] << "\t" << "ID: " << bookIds[i] << endl;

cout << "----------------\n";

}

}

// Function to add a new member to the library

void addMember(vector<string>& names, vector<string>& ids, vector<string>& emails) {

string name, id, email;

cout << "Enter member name: ";

cin >> name;

cout << "Enter member ID: ";

cin >> id;

cout << "Enter member email: ";

cin >> email;

names.push\_back(name);

ids.push\_back(id);

emails.push\_back(email);

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

person ++;

}

// Function to remove a member from the library by ID

void removeMember(vector<string>& names, vector<string>& ids, vector<string>& emails) {

string id;

cout << "Enter member ID to remove: ";

cin >> id;

for (int i = 0; i < ids.size(); i++) {

if (ids[i] == id) {

names.erase(names.begin() + i);

ids.erase(ids.begin() + i);

emails.erase(emails.begin() + i);

cout << "Member removed successfully!" << endl;

return;

}

}

cout << "Member not found! Try again." << endl;

}

// Function to display all members in the library

void viewMembers(vector<string>& names, vector<string>& ids) {

cout << "Library Members:" << endl;

for (int i = 0; i < ids.size(); i++) {

cout << ids[i] << " - " << names[i] << endl;

}

cout << endl;

}

// Main function to manage library members

void manage\_members() {

vector<string> names;

vector<string> ids;

vector<string> emails;

int choice = 0;

while (choice != 4) {

cout << endl;

cout << "Library Management System - Admin Menu" << endl;

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

cout << "2. Remove Member" << endl;

cout << "3. View Members" << endl;

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

cout << "Enter choice: ";

cin >> choice;

cout << "\n";

switch (choice) {

case 1:

system("CLS"); // clears the output screen

addMember(names, ids, emails);

break;

case 2:

system("CLS"); // clears the output screen

removeMember(names, ids, emails);

break;

case 3:

system("CLS"); // clears the output screen

viewMembers(names, ids);

break;

case 4:

system("CLS"); // clears the output screen

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

system("CLS"); // clears the output screen

break;

default:

cout << "Invalid choice! Try again." << endl;

break;

}

}

return;

}

// Function to submit fines for a member

void submitFines() {

int memberId;

float amount;

cout << "Enter member ID: ";

cin >> memberId;

cout << "Enter fine amount: ";

cin >> amount;

fines[memberId] += amount;

cout << "Fine of " << amount << " added for member with ID: " << memberId << endl;

}

// Function to display fines in the library

void displayFines() {

cout << "Fines: " << endl;

for (map<int, float>::iterator it = fines.begin(); it != fines.end(); ++it) {

cout << "Member ID: " << it->first << " - Fine Amount: " << it->second << endl;

}

cout << endl;

}

// function to generate reports

void generate\_reports() {

int a;

int memberId, amount;

string feedback;

vector<string> names;

vector<string> ids;

do {

cout << "Libraray Report" << endl;

cout << "1. Requested Book Check\n";

cout << "2. Member Check\n";

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

cout << "4. Feedbacks \n";

cout << "5. Hold Book\n";

cout << "6. Fine Collections\n";

cout << "7. Exist\n";

cout<< "Enter Your Choice: ";

cin >> a;

cout << "\n";

system("CLS"); // clears the output screen

switch (a) {

case 1:

cout << "Total Number of Requested Books: "<< tcount << endl;

if (tcount > 0) {

cout << "Requested Books:" << endl;

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

cout << requestedBooks[i] << endl;

}

}

cout << endl;

break;

case 2:

cout << "Total No of Members Added: "<< person << endl;

break;

case 3:

cout << "Total No of books added: " << bookCount << endl;

cout << "\nList of Books:\n";

cout << "----------------\n";

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

cout << "Title: " << bookTitles[i] << "\t" << "Author: " << bookAuthors[i] << "\t" << "ID: " << bookIds[i] << endl;

cout << "----------------\n";

}

cout << endl;

break;

case 4:

cout << "Feedbacks "<< endl;

cout << endl << "\n---------------------------\n";

if (feedbacks.empty()) {

cout << "No feedbacks to display." << endl;

} else {

for (int i = 0; i < feedbacks.size(); i++) {

cout << feedbacks[i] << endl;

}

}

break;

case 5:

if(holdbooks.size() == 0) { // if there are no hold books

cout << "You do not currently have any hold books." << endl;

} else {

cout << "Hold Books: " << endl;

for(int i = 0; i < holdbooks.size(); i++) {

cout << holdbooks[i] << endl;

}

}

case 6:

displayFines();

break;

case 7:

cout << "Exiting...";

cout << endl;

default:

cout << "Invalid Input";

cout << endl;

}

}

while (a != 7); {

system("CLS"); // clears the output screen

}

return;

}

// Primary Function to delete a book from the library

void deleteBook()

{

int id;

cout << "Enter the ID of the book to delete: ";

cin >> id;

// Find the index of the book with the given ID

vector<int>::iterator it = find(bookIds.begin(), bookIds.end(), id);

if (it != bookIds.end()) {

// Remove the book from the vectors

int index = distance(bookIds.begin(), it);

bookTitles.erase(bookTitles.begin() + index);

bookAuthors.erase(bookAuthors.begin() + index);

bookIds.erase(it);

bookCount--;

cout << "Book deleted successfully" << endl;

}

else {

cout << "Book with ID " << id << " not found" << endl;

}

}

// main function to delete books

void delete\_books() {

int choice;

do {

// Display the main menu

cout << "\nMain Menu" << endl;

cout << "1. Delete a book" << endl;

cout << "2. Exit" << endl;

cout << "Enter your choice (1-3): ";

cin >> choice;

switch (choice) {

case 1:

deleteBook();

break;

case 2:

cout << "Exiting..." << endl;

break;

default:

cout << "Invalid choice" << endl;

break;

}

} while (choice != 2); {

system("CLS"); // clears the output screen

}

return;

}

// function for book search

void book\_search() {

string title;

cout << "Enter the Name of the book: ";

cin >> title;

// Find the name of the book with the given title

vector<string>::iterator it = find(bookTitles.begin(), bookTitles.end(), title);

if (it != bookTitles.end()) {

cout << "\n -----------------------------------\n";

cout << "This Book is Avaiable" << endl;

}

else {

cout << "\n -----------------------------------\n";

cout << "Book: " << title << " not found" << endl;

}

}

// function for requesting books

void request\_books() {

string bookTitle;

cout << "Enter the Title of the Requested Book: ";

cin >> bookTitle;

requestedBooks.push\_back(bookTitle);

tcount++;

cout << "\nResponse Submitted Successfully!\n";

}

// function for book recommendations

void book\_recommendations() {

// Display recommended books in the database

cout << "\nRecommended For You\n";

cout << "----------------\n";

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

cout << "Title: " << bookTitles[i] << "\t" << "Author: " << bookAuthors[i] << "\t" << "ID: " << bookIds[i] << endl;

cout << "----------------\n";

}

}

// function for Feedback

void feedback() {

string feedback\_text;

cout << "Enter your feedback: ";

cin.ignore();

getline(cin, feedback\_text);

feedbacks.push\_back(feedback\_text);

cout << "\nThank you for your feedback!" << endl;

}

// function for holding and reserving books

void hold\_books() {

string title;

int a;

cout << "enter the name of the book: ";

cin >> title;

// Find the name of the book with the given title

vector<string>::iterator it = find(bookTitles.begin(), bookTitles.end(), title);

if (it != bookTitles.end()) {

cout << "\n -----------------------------------\n";

cout << "This Book is Avaiable" << endl;

cout << "Press 1 to reserve this book: ";

cin >> a;

if (a==1) {

hcount++;

holdbooks.push\_back(title); // add the book to holdbooks vector

cout << "This book is reserved in your cart" << endl;

}

else {

cout << "invalid input!" << endl;

}

}

else {

cout << "\n -----------------------------------\n";

cout << "book: " << title << " not found" << endl;

}

return;

}

// Function to manage user profiles

void manage\_profile(map<string, string>& userCredentials, const string& username)

{

string newPassword;

cout << "\nManage Profile\n";

// Check if the user exists

if (userCredentials.find(username) == userCredentials.end())

{

cout << "User not found!" << endl;

return;

}

// Get the new password from the user

cout << "Enter new password: ";

cin >> newPassword;

// Update the password in the userCredentials map

userCredentials[username] = newPassword;

cout << "Password updated successfully!" << endl;

}

// Function to display the main menu

void displayMainMenu()

{

cout << "\nMain Menu" << endl;

cout << "1. Login" << endl;

cout << "2. Signup" << endl;

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

cout << "Enter your choice (1-3): ";

}

// Function to handle the admin login

void adminLogin(map<string, string>& adminCredentials)

{

string username, password;

cout << "\nAdmin Login" << endl;

// Get the username and password from the user

cout << "Enter username: ";

cin >> username;

cout << "Enter password: ";

cin >> password;

// Check if the entered username and password are valid

if (adminCredentials.count(username) > 0 && adminCredentials[username] == password)

{

system("CLS"); // clears the output screen

cout << "Login successful" << endl;

int choice;

// loop until admin logs out

// loop until admin logs out

while (true){

cout << "\nAdmin Menu:\n";

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

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

cout << "3. Manage Members\n";

cout << "4. Create Report\n";

cout << "5. Available Books \n";

cout << "6. Logout\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice)

{

case 1:

system("CLS"); // clears the output screen

add\_books();

break;

case 2:

system("CLS"); // clears the output screen

delete\_books();

break;

case 3:

system("CLS"); // clears the output screen

manage\_members();

break;

case 4:

system("CLS"); // clears the output screen

generate\_reports();

break;

case 5:

system("CLS"); // clears the output screen

show\_books();

break;

case 6:

system("CLS"); // clears the output screen

cout << "Logging out...\n";

return;

default:

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

break;

}

}

}

else

{

cout << "Invalid username or password" << endl;

}

}

// Function to handle the admin signup

void adminSignup(map<string, string>& adminCredentials)

{

string username, password;

cout << "\nAdmin Signup" << endl;

// Get the username and password from the user

cout << "Enter username: ";

cin >> username;

cout << "Enter password: ";

cin >> password;

// Add the new admin credentials to the map

adminCredentials[username] = password;

cout << "Signup successful" << endl;

}

// function to login or sign up for admin

void admin\_login() {

map<string, string> adminCredentials;

// Add some sample admin credentials to the map

adminCredentials["admin1"] = "password1";

adminCredentials["admin2"] = "password2";

int choice;

do

{

displayMainMenu();

cin >> choice;

switch (choice)

{

case 1:

adminLogin(adminCredentials);

break;

case 2:

adminSignup(adminCredentials);

break;

case 3:

cout << "Exiting..." << endl;

break;

default:

cout << "Invalid choice" << endl;

break;

}

} while (choice != 3); {

system("CLS"); // clears the output screen

}

return;

}

// Function to display the main menu

void displayMainMenu2()

{

cout << "\nMain Menu" << endl;

cout << "1. Login" << endl;

cout << "2. Signup" << endl;

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

cout << "Enter your choice (1-3): ";

}

// Function to handle the user login

void userLogin (map<string, string>& userCredentials) {

string username, password;

cout << "\nUser Login" << endl;

// Get the username and password from the user

cout << "Enter username: ";

cin >> username;

cout << "Enter password: ";

cin >> password;

// Check if the entered username and password are valid

if (userCredentials.count(username) > 0 && userCredentials[username] == password) {

system("CLS");

// clears the output screen

cout << "Login successful" << endl;

int choice;

// Loop until user logs out

while (true) {

cout << "\nUser Menu:\n";

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

cout << "2. Request for Books\n";

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

cout << "4. Give Feedbacks\n";

cout << "5. Holds and Reserves Books\n";

cout << "6. Submit Fines\n";

cout << "7. Manage Profile\n";

cout << "8. Logout\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

system ("CLS"); // clears the output screen

book\_search();

break;

case 2:

system ("CLS"); // clears the output screen

request\_books();

break;

case 3:

system ("CLS"); // clears the output screen

book\_recommendations();

break;

case 4:

system ("CLS"); // clears the output screen

feedback();

break;

case 5:

system ("CLS"); // clears the output screen

hold\_books();

break;

case 6:

system ("CLS"); // clears the output screen

submitFines();

break;

case 7:

system ("CLS"); // clears the output screen

manage\_profile (userCredentials, username);

break;

case 8:

system ("CLS"); // clears the output screen

cout << "Logging out...\n";

return;

default:

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

break;

}

}

}

else{

cout << "Invalid username or password" << endl;

}

}

// Function to handle the user signup

void userSignup(map<string, string>& userCredentials)

{

string username, password;

cout << "\nUser Signup" << endl;

// Get the username and password from the user

cout << "Enter username: ";

cin >> username;

cout << "Enter password: ";

cin >> password;

// Add the new admin credentials to the map

userCredentials[username] = password;

cout << "Signup successful" << endl;

}

// function to login or sign up for user

void user\_login() {

map<string, string> userCredentials;

// Add some sample admin credentials to the map

userCredentials["user1"] = "password1";

userCredentials["user2"] = "password2";

int choice;

do

{

displayMainMenu2();

cin >> choice;

switch (choice)

{

case 1:

userLogin(userCredentials);

break;

case 2:

userSignup(userCredentials);

break;

case 3:

cout << "Exiting..." << endl;

system("CLS"); // clears the output screen

break;

default:

cout << "Invalid choice" << endl;

break;

}

} while (choice != 3); {

system("CLS"); // clears the output screen

}

return;

}

// main function

int main()

{

int choice;

do {

cout << "Welcome to Library Management System" << endl;

cout << "1. Admin Login/Sign Up" << endl;

cout << "2. User Login/Sign Up" << endl;

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

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

admin\_login();

break;

case 2:

user\_login();

break;

case 3:

cout << "Exiting" << endl;

break;

default:

cout << "Invalid choice. Try again" << endl;

break;

}

} while (choice != 3);

return 0;

}