#include <iostream>

#include <string>

#include <vector>

using namespace std;

// Define structures for hospital details, doctor details, and a binary search tree node

struct Hospital {

string name;

string city;

string state;

string specialty;

};

struct Doctor {

string name;

string field;

string timings;

};

struct BSTNode {

Doctor doctor;

BSTNode\* left;

BSTNode\* right;

};

// Hashing function to map hospital names to their details

const int HASH\_TABLE\_SIZE = 100;

vector<Hospital> hashTable[HASH\_TABLE\_SIZE];

// Function to add a hospital to the hash table

void addHospital(const Hospital& hospital) {

int index = hash<string>{}(hospital.name) % HASH\_TABLE\_SIZE;

hashTable[index].push\_back(hospital);

}

// Function to search for a hospital by name

Hospital\* findHospital(const string& name) {

int index = hash<string>{}(name) % HASH\_TABLE\_SIZE;

for (Hospital& hospital : hashTable[index]) {

if (hospital.name == name) {

return &hospital;

}

}

return nullptr;

}

// Function to create a new binary search tree node

BSTNode\* createBSTNode(const Doctor& doctor) {

BSTNode\* newNode = new BSTNode;

newNode->doctor = doctor;

newNode->left = newNode->right = nullptr;

return newNode;

}

// Function to insert a doctor into the binary search tree

BSTNode\* insertDoctor(BSTNode\* root, const Doctor& doctor) {

if (!root) {

return createBSTNode(doctor);

}

if (doctor.name < root->doctor.name) {

root->left = insertDoctor(root->left, doctor);

} else if (doctor.name > root->doctor.name) {

root->right = insertDoctor(root->right, doctor);

}

return root;

}

// Function to search for a doctor by name in the binary search tree

BSTNode\* findDoctor(BSTNode\* root, const string& name) {

if (!root || root->doctor.name == name) {

return root;

}

if (name < root->doctor.name) {

return findDoctor(root->left, name);

} else {

return findDoctor(root->right, name);

}

}

// Function to search for doctors by field in the binary search tree and display their details

void searchAndDisplayDoctors(BSTNode\* root, const string& field) {

if (root) {

searchAndDisplayDoctors(root->left, field);

if (root->doctor.field == field) {

cout << "Doctor Name: " << root->doctor.name << endl;

cout << "Field: " << root->doctor.field << endl;

cout << "Timings: " << root->doctor.timings << endl;

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

}

searchAndDisplayDoctors(root->right, field);

}

}

// Display a terminal-like menu

void displayTerminalMenu() {

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

cout << "Loc-A-Doc Terminal" << endl;

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

cout << "Available Options:" << endl;

cout << "1. Register" << endl;

cout << "2. Login" << endl;

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

cout << "Enter your choice: ";

}

// Admin and User registration functions

string adminName;

string adminID;

string userPhoneNumber;

bool registerAdmin() {

cout << "Admin Registration" << endl;

cout << "Enter your name: ";

cin >> adminName;

cout << "Enter your admin ID: ";

cin >> adminID;

cout << "Admin registration successful." << endl;

return true;

}

bool registerUser() {

cout << "User Registration" << endl;

cout << "Enter your phone number: ";

cin >> userPhoneNumber;

cout << "User registration successful." << endl;

return true;

}

int main() {

bool isAdmin = false;

bool isUser = false;

BSTNode\* doctorBST = nullptr; // Binary search tree to store doctors

// Declare the variables outside the switch statement

bool found = false;

string searchCity;

string doctorField; // Added declaration

while (true) {

int choice;

displayTerminalMenu();

cin >> choice;

switch (choice) {

case 1:

// Implement registration

if (!isAdmin && !isUser) {

int registrationChoice;

cout << "User Registration" << endl;

cout << "1. Register as Admin" << endl;

cout << "2. Register as User" << endl;

cout << "Enter your choice: ";

cin >> registrationChoice;

if (registrationChoice == 1) {

isAdmin = registerAdmin();

} else if (registrationChoice == 2) {

isUser = registerUser();

} else {

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

}

} else {

cout << "You are already registered." << endl;

}

break;

case 2:

// Implement login

if (!isAdmin && !isUser) {

cout << "You need to register before you can login." << endl;

} else {

int loginChoice;

cout << "Login" << endl;

if (isAdmin) {

cout << "Admin Login" << endl;

// Check admin credentials

string enteredAdminName, enteredAdminID;

cout << "Enter admin name: ";

cin >> enteredAdminName;

cout << "Enter admin ID: ";

cin >> enteredAdminID;

if (enteredAdminName == adminName && enteredAdminID == adminID) {

cout << "Admin login successful." << endl;

while (isAdmin) {

int adminOptions;

cout << "Admin Options:" << endl;

cout << "1. Register a Doctor" << endl;

cout << "2. Register a Hospital" << endl;

cout << "3. Logout" << endl;

cout << "Enter your choice: ";

cin >> adminOptions;

switch (adminOptions) {

case 1:

if (isAdmin) {

// Doctor registration

cout << "Doctor Registration" << endl;

Doctor newDoctor;

cout << "Enter doctor's name: ";

cin.ignore();

getline(cin, newDoctor.name);

cout << "Enter doctor's field: ";

getline(cin, newDoctor.field);

cout << "Enter doctor's timings: ";

getline(cin, newDoctor.timings);

// Add the doctor to your data structure (binary search tree)

doctorBST = insertDoctor(doctorBST, newDoctor);

cout << "Doctor registration successful." << endl;

} else {

cout << "You need to log in as an admin to register a doctor." << endl;

}

break;

case 2:

// Implement hospital registration

if (isAdmin) {

cout << "Hospital Registration" << endl;

Hospital newHospital;

// Prompt for hospital details

cout << "Enter hospital name: ";

cin.ignore();

getline(cin, newHospital.name);

cout << "Enter city: ";

getline(cin, newHospital.city);

cout << "Enter state: ";

getline(cin, newHospital.state);

cout << "Enter timings: ";

getline(cin, newHospital.specialty);

// Add the hospital to your data structure (e.g., hashTable)

addHospital(newHospital);

cout << "Hospital registration successful." << endl;

} else {

cout << "You need to log in as an admin to register a hospital." << endl;

}

break;

case 3:

// Logout admin

isAdmin = false;

cout << "Admin logout successful." << endl;

break;

default:

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

break;

}

}

} else {

cout << "Admin login failed. Invalid credentials." << endl;

}

} else if (isUser) {

cout << "User Login" << endl;

// Check user credentials

string enteredUserPhoneNumber;

cout << "Enter your phone number: ";

cin >> enteredUserPhoneNumber;

if (enteredUserPhoneNumber == userPhoneNumber) {

cout << "User login successful." << endl;

while (isUser) {

int userOptions;

cout << "User Options:" << endl;

cout << "1. Search for a Hospital" << endl;

cout << "2. Search for a Doctor" << endl;

cout << "3. Logout" << endl;

cout << "Enter your choice: ";

cin >> userOptions;

switch (userOptions) {

case 1:

// Implement search for a hospital by city

cout << "Enter the city to search for hospitals: ";

cin.ignore();

getline(cin, searchCity);

found = false;

for (int i = 0; i < HASH\_TABLE\_SIZE; i++) {

for (const Hospital& hospital : hashTable[i]) {

if (hospital.city == searchCity) {

cout << "Hospital Name: " << hospital.name << endl;

cout << "City: " << hospital.city << endl;

cout << "State: " << hospital.state << endl;

cout << "Specialty: " << hospital.specialty << endl;

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

found = true;

}

}

}

if (!found) {

cout << "No hospitals found in the city: " << searchCity << endl;

}

break;

case 2:

// Implement search for a doctor by field

cout << "Enter the field to search for doctors: ";

cin.ignore();

getline(cin, doctorField);

found = false;

cout << "Doctors in the field of " << doctorField << ":" << endl;

// Search for doctors by field and display their details

searchAndDisplayDoctors(doctorBST, doctorField);

if (!found) {

cout << "No doctors found in the field: " << doctorField << endl;

}

break;

case 3:

// Logout user

isUser = false;

cout << "User logout successful." << endl;

break;

default:

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

break;

}

}

} else {

cout << "User login failed. Invalid phone number." << endl;

}

}

}

break;

case 3:

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

cout<<"Thank You !!!"<<endl;

return 0;

default:

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

}

}

return 0;

}