**Code**

#include <bits/stdc++.h>

using namespace std;

class User

{

protected:

string user\_id, name, email, password, role;

public:

User(string id, string n, string e, string p, string r)

: user\_id(id), name(n), email(e), password(p), role(r) {}

virtual void registerUser() = 0;

virtual bool login(string e, string p) = 0;

virtual void updateProfile(string newName, string newEmail) = 0;

string getRole() { return role; }

string getEmail() { return email; }

};

class Patient : public User

{

private:

vector<string> medical\_history;

vector<string> appointments;

unordered\_map<string, string> insurance\_details;

public:

Patient(string id, string n, string e, string p, unordered\_map<string, string> insurance)

: User(id, n, e, p, "Patient"), insurance\_details(insurance) {}

void registerUser() override

{

cout << "Patient " << name << " registered." << endl;

}

bool login(string e, string p) override

{

return (email == e && password == p);

}

void updateProfile(string newName, string newEmail) override

{

name = newName;

email = newEmail;

cout << "Profile updated for " << name << endl;

}

void requestAppointment(string doctor\_id, string date\_time)

{

appointments.push\_back(doctor\_id + " on " + date\_time);

cout << "Appointment requested with Doctor ID " << doctor\_id << " at " << date\_time << endl;

}

void addMedicalRecord(string record)

{

medical\_history.push\_back(record);

cout << "Added Medical Record: " << record << endl;

}

};

class Doctor : public User

{

private:

string specialization;

vector<string> appointments;

vector<string> patients;

public:

Doctor(string id, string n, string e, string p, string spec)

: User(id, n, e, p, "Doctor"), specialization(spec) {}

void registerUser() override

{

cout << "Doctor " << name << " registered." << endl;

}

bool login(string e, string p) override

{

return (email == e && password == p);

}

void updateProfile(string newName, string newEmail) override

{

name = newName;

email = newEmail;

cout << "Profile updated for Dr. " << name << endl;

}

void addMedicalRecord(string patient\_id, string record)

{

patients.push\_back(patient\_id + ": " + record);

cout << "Medical record added for Patient ID: " << patient\_id << endl;

}

};

class Administrator : public User

{

public:

Administrator(string id, string n, string e, string p)

: User(id, n, e, p, "Administrator") {}

void addUser(shared\_ptr<User> user)

{

user->registerUser();

}

void removeUser(string user\_id)

{

cout << "User ID " << user\_id << " removed from the system." << endl;

}

void generateReports()

{

cout << "Generating reports..." << endl;

}

void registerUser() override

{

cout << "Administrator " << name << " registered." << endl;

}

bool login(string e, string p) override

{

return (email == e && password == p);

}

void updateProfile(string newName, string newEmail) override

{

name = newName;

email = newEmail;

cout << "Profile updated for " << name << endl;

}

};

int main()

{

shared\_ptr<Administrator> admin = make\_shared<Administrator>("ADMIN001", "Admin", "admin@hospital.com", "adminpass");

vector<shared\_ptr<User>> users;

users.push\_back(admin);

int choice;

while (true)

{

cout << "\nWelcome to Healthcare Management System!\n";

cout << "1. Login as Administrator\n2. Login as Doctor\n3. Login as Patient\n4. Exit\n";

cout << "Choose an option: ";

cin >> choice;

if (choice == 1)

{

// Administrator login

string email, password;

cout << "Enter Admin Email: ";

cin >> email;

cout << "Enter Password: ";

cin >> password;

if (admin->login(email, password))

{

cout << "Admin logged in successfully!\n";

int admin\_choice;

while (true)

{

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

cout << "1. Add User\n2. Remove User\n3. Generate Reports\n4. Logout\n";

cout << "Choose an option: ";

cin >> admin\_choice;

if (admin\_choice == 1)

{

string role, id, name, email, password, specialization, insurance\_provider, policy\_number;

cout << "Enter 1 for Doctor 2 for Patient): ";

cin >> role;

if (role == "1")

{

cout << "Enter Doctor ID: ";

cin >> id;

cout << "Enter Doctor Name: ";

cin.ignore();

getline(cin, name);

cout << "Enter Doctor Email: ";

cin >> email;

cout << "Enter Doctor Password: ";

cin >> password;

cout << "Enter Specialization: ";

cin >> specialization;

shared\_ptr<Doctor> new\_doctor = make\_shared<Doctor>(id, name, email, password, specialization);

admin->addUser(new\_doctor);

users.push\_back(new\_doctor);

}

else if (role == "2")

{

cout << "Enter Patient ID: ";

cin >> id;

cout << "Enter Patient Name: ";

cin.ignore();

getline(cin, name);

cout << "Enter Patient Email: ";

cin >> email;

cout << "Enter Patient Password: ";

cin >> password;

cout << "Enter Insurance Provider: ";

cin.ignore();

getline(cin, insurance\_provider);

cout << "Enter Policy Number: ";

cin >> policy\_number;

unordered\_map<string, string> insurance = {{"provider", insurance\_provider}, {"policy\_number", policy\_number}};

shared\_ptr<Patient> new\_patient = make\_shared<Patient>(id, name, email, password, insurance);

admin->addUser(new\_patient);

users.push\_back(new\_patient);

}

}

else if (admin\_choice == 2)

{

string user\_id;

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

cin >> user\_id;

admin->removeUser(user\_id);

}

else if (admin\_choice == 3)

{

admin->generateReports();

}

else if (admin\_choice == 4)

{

break;

}

}

}

else

{

cout << "Invalid login credentials for Admin!\n";

}

}

else if (choice == 2 || choice == 3)

{

string email, password;

cout << "Enter Email: ";

cin >> email;

cout << "Enter Password: ";

cin >> password;

bool loggedIn = false;

shared\_ptr<User> loggedInUser = nullptr;

for (auto &user : users)

{

if (user->getEmail() == email && user->login(email, password))

{

loggedIn = true;

loggedInUser = user;

break;

}

}

if (loggedIn && loggedInUser->getRole() == "Doctor")

{

cout << "Doctor logged in successfully!\n";

int doc\_choice;

while (true)

{

cout << "\nDoctor Actions:\n";

cout << "1. Add Medical Record\n2. Logout\n";

cout << "Choose an option: ";

cin >> doc\_choice;

if (doc\_choice == 1)

{

string patient\_id, record;

cout << "Enter Patient ID: ";

cin >> patient\_id;

cout << "Enter Medical Record: ";

cin.ignore();

getline(cin, record);

dynamic\_pointer\_cast<Doctor>(loggedInUser)->addMedicalRecord(patient\_id, record);

}

else if (doc\_choice == 2)

{

break;

}

}

}

else if (loggedIn && loggedInUser->getRole() == "Patient")

{

cout << "Patient logged in successfully!\n";

int pat\_choice;

while (true)

{

cout << "\nPatient Actions:\n";

cout << "1. Request Appointment\n2. Add Medical Record\n3. Logout\n";

cout << "Choose an option: ";

cin >> pat\_choice;

if (pat\_choice == 1)

{

string doctor\_id, appointment\_time;

cout << "Enter Doctor ID: ";

cin >> doctor\_id;

cout << "Enter Appointment Date & Time (format: YYYY-MM-DD HH:MM): ";

cin.ignore();

getline(cin, appointment\_time);

dynamic\_pointer\_cast<Patient>(loggedInUser)->requestAppointment(doctor\_id, appointment\_time);

}

else if (pat\_choice == 2)

{

string record;

cout << "Enter Medical Record: ";

cin.ignore();

getline(cin, record);

dynamic\_pointer\_cast<Patient>(loggedInUser)->addMedicalRecord(record);

}

else if (pat\_choice == 3)

{

break;

}

}

}

else

{

cout << "Invalid login credentials!\n";

}

}

else if (choice == 4)

{

cout << "Exiting system...\n";

break;

}

}

return 0;

}

**OUTPUT**







