Project Report: Hospital Management System

Authors: Baris AK & Kerem Emre ESLEMEZ

Student IDs: 5003240010 | 5003240017

Course: Object-Oriented Programming Term Project

Institution: Istanbul University

1. Project Overview

This project implements a **Hospital Management System** in C++ designed to manage patients, doctors, and their interactions through appointments. The system is built using modular programming, separating declarations into header files (`.h`) and logic into implementation files (`.cpp`).

2. Technical Architecture & OOP Implementation

2.1 Inheritance and Polymorphism

The system uses a base class `Person` to handle common identity data, which is then extended by `Patient` and `Doctor`.

- **Code Example (Inheritance):**

```
class Patient : public Person {
private:
    int age;
    string illness;
```

- **Code Example (Polymorphism):** We use `virtual` functions to ensure the correct `printInfo()` is called at runtime.

```
    virtual void printInfo();
```

```
void Patient::printInfo() {
    cout << "Patient - ID: " << getId() << ", Name: " << getName()
         << ", Age: " << age << ", Illness: " << illness << endl;
}
```

2.2 Encapsulation

Data integrity is maintained by keeping attributes `private` and providing `public` getter and setter methods.

- **Code Example:**

```
void Person::setName(string name) {
    this->name = name;
}
```

2.3 Composition and Associations

The `Appointment` class demonstrates **composition** by grouping objects of other classes together.

- **Code Example:**

```
class Appointment {
private:
    Patient patient;
    Doctor doctor;
    string date;
```

3. Core Functionalities

3.1 Data Management (The Hospital Class)

The `Hospital` class acts as the central engine, using `std::vector` to store records dynamically.

- **Adding Records:**

```
void Hospital::addPatient(Patient p) {
    patients.push_back(p);
    cout << "Patient added." << endl;
}
```

3.2 Search and Validation Logic

Before an appointment is created, the system searches for the existence of the Patient and Doctor by their ID.

- **Code Example (Search):**

```
Patient* Hospital::findPatient(int id) {
    for (int i = 0; i < patients.size(); i++) {
        if (patients[i].getId() == id) {
            return &patients[i];
        }
    }
    return NULL;
}
```

3.3 User Interface (Main Menu)

The `main.cpp` provides an interactive loop that processes user input and executes the appropriate hospital logic.

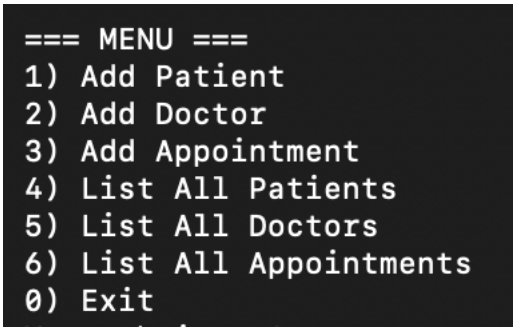
4. Class Summary Table

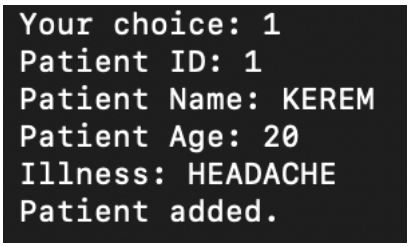
Class	Responsibility	Key Features Used
Person	Base class for all individuals	Virtual Destructor, Protected/Private members
Patient	Stores medical data	Inheritance, Overriding
Doctor	Stores specialty and assigned patients	Inheritance, Vector of Patients
Appointment	Links Patient and Doctor	Object Interaction
Hospital	System controller	Data search, Vector management

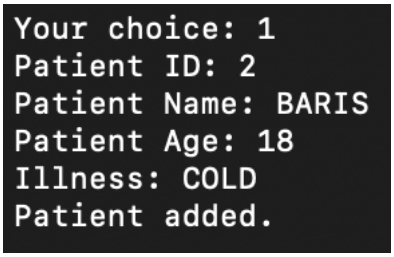
5. Conclusion

The "Hospital Management System" effectively utilizes C++ OOP principles to create a scalable and organized application. By separating the logic into specific classes and using inheritance, the code remains clean and easy to maintain.

6. Outputs







```
Your choice: 2
Doctor ID: 11
Doctor Name: SMITH
Specialty: CARDIOLOGY
Doctor added.
```

```
Your choice: 2
Doctor ID: 2
Doctor Name: BROWN
Specialty: ORTHOPEDIC
Doctor added.
```

```
Your choice: 3
Patient ID: 1
Doctor ID: 11
Appointment Date (DD/MM/YYYY): 31/12/2025
Appointment added.
```

```
Your choice: 3
Patient ID: 2
Doctor ID: 2
Appointment Date (DD/MM/YYYY): 01/01/2026
Appointment added.
```

```
Your choice: 4

=== All Patients ===
Patient - ID: 1, Name: KEREM, Age: 20, Illness: HEADACHE
Patient - ID: 2, Name: BARIS, Age: 18, Illness: COLD
```

```
Your choice: 5

=== All Doctors ===
Doctor - ID: 11, Name: SMITH, Specialty: CARDIOLOGY, Patient Count: 0
Doctor - ID: 2, Name: BROWN, Specialty: ORTHOPEDIC, Patient Count: 0
```

```
Your choice: 6

=== All Appointments ===
Appointment - Patient: KEREM, Doctor: SMITH, Date: 31/12/2025
Appointment - Patient: BARIS, Doctor: BROWN, Date: 01/01/2026
```