Updated Design Overview for Hospital Management System

Name: Nguyen Duc Thang
Student ID: 104776473

Custom Program methods and design features

Overview of the program

The program implemented will provide treatments and stores their informations such as the patient's name, date of birth, phone number, symptoms, type of treatment, medical staff assigned. The code will be using the Patient class to retrieve the user's information, then you can assign them to a specific observer (medical staff) by using the AttachObserver() method. After all the necessary information related to the patient are done the Manage class will give them a treatment choice depends on their condition using the TreatmentPlan() class, then the Manage class will add the patient's record using the AddPatientRecord() method, the manage class can also delete the patient record using the RemovePatientRecord(). Now the Medical staff can give treatment to the patient using the Diagnose() and ProvideTreatment() method. After the patient's data are stored in the txt file the Nurse() class can edit the patient record by using the

Design features

the custom program use a total of 5 design patterns in order to make the solution more practical while keeping the code's stability and scalability

- Singleton Pattern
- Factory Pattern
- Observer Pattern
- Strategy Pattern
- Decorator Pattern

Classes used in the custom program

Responsibility	Type Details	Notes
_medical_staff:	MedicalStaff	Include information about doctors and nurse
_patient:	Patient	Include patient information and symtoms
_receptionist:	Receptionist	Include information and roles of a receptionist
_patient_record:	PatientRecord	Include: date and description given to the patient
_issue_date:	DateTime	Contains the day, month and year
_is_valid:	Bool	Check if the record is from a medical staff or not
_name:	String	Name of patient or medical staff
_contact:	String	Phone number of the person
_dob:	DateTime	Date of birth of the person
_symptoms:	string[]	Problem the patient are having
_working_hours:	string	Time that a medical staff will be on
_experience:	string	Number of years in the job
Hospital()	void	Contains patient and medical staff data
Manage()	void	Control how many medical staff are available and enable patient record and find it
PatientRecord()	void	Contains feedback given from the medical staff
Receptionist()	void	Can add and remove a patient record
Person()	void	Abstract class represent a human information
Patient()	void	Include patient's personal information and symtoms
Nurse()	void	A medical staff that will update the patient record
SurgicalNurse()	void	A type of nurse will update feedback from a surgeon
PediaricNurse()	void	A type of nurse will update feedback from a pediatric
EmergencyNurse()	void	A type of nurse will update feedback from others doctor type
MedicalStaff()	void	Abstract class inherit from person class
Surgeon()	void	Will give feedback to the patient
Pediatric()	void	Will give feedback to the patient
Cardiologist()	void	Will give feedback to the patient
Psychologist()	void	Will give feedback to the patient
MedicalStaffFactory	void	Create a new Medical Staff
TreatmentPlan	void	Give the solution based on patient's condition

MildTreatment	void	Give a mild treatment to the medical staff
AggressiveTreatment	void	Give a aggressive treatment to the medical staff
MedicalStaffFactory	void	Create a medical staff
IObserver()	void	Update the data based on the input
Doctor()	void	Set treatment for patient
MedicalStaffDecorator()	void	Allow flexible addition to the responsibility to medical staff
SpecialDecorator()	void	Use for rare patient condition that need special consultation
EmergencyDecorator()	void	Use for patient with emergency issue

Class Diagram

Sequence Diagram

Doctor
TreatmentPlan
MildTreatment
AggressiveTreatment
SetTreatment(MildTreatment)
ExecuteTreatment(Patient)
SetTreatment(AggressiveTreatment)
ExecuteTreatment(Patient)

```
namespace HospitalManagementSystem
{
    public class AggressiveTreatment : TreatmentPlan
    {
        public string ExecuteTreatment(Patient patient)
        {
            return "Aggressive";
        }
    }
}

```



```
namespace HospitalManagementSystem
{
    public class Cardiologist : Doctor
    {
        public Cardiologist(string name, string contact, DateTime dob,
          string workingHours, string experience):base(name, contact, dob,
          workingHours, experience)
        {
        }


        public override void Diagnose()
        {
            Console.WriteLine($"{Name} (Cardiologist) is diagnosing a heart
               condition.");
        }

        public override void ProvideTreatment()
        {
            Console.WriteLine($"{Name} (Cardiologist) is providing heart
              treatment.");
        }
    }
}

```

```
namespace HospitalManagementSystem
{
    public class Doctor : MedicalStaff, IObserver
    {
        public Doctor(string name, string contact, DateTime dob, string workingHours, string experience):base(name, contact, dob, workingHours, experience)
        {
        }

        public void Update(Patient patient)
        {
            Console.WriteLine($"Doctor {Name} has been notified of the patient's update: {patient.Name}");

            Console.WriteLine($"Updated Symptoms: {string.Join(", ", patient.Symptoms)}");
        }

        public override void Diagnose()
        {
            Console.WriteLine($"Doctor {Name} is diagnosing the patient.");
        }

        public override void ProvideTreatment()
        {
            Console.WriteLine($"Doctor {Name} is providing treatment.");
        }
    }

}

```

```
using HospitalManagementSystem;

public class EmergencyDecorator : MedicalStaff
{
    private MedicalStaff _medical_staff;

    public EmergencyDecorator(MedicalStaff medical_staff) : base (medical_staff.Name, medical_staff.Contact, medical_staff.DOB, medical_staff.WorkingHours, medical_staff.Experience)
    {
        _medical_staff = medical_staff;
    }

    public override void Diagnose()
    {
        _medical_staff.Diagnose();
        Console.WriteLine("Emergency handling diagnosis.");
    }

    public override void ProvideTreatment()
    {
        _medical_staff.ProvideTreatment();
        Console.WriteLine("Emergency treatment provided.");
    }
}

```

```
namespace HospitalManagementSystem
{
    public class EmergencyNurse : Nurse
    {
        public EmergencyNurse(string name, string contact, DateTime dob, string working_hours, string experience):base(name, contact, dob, working_hours, experience)
        {
        }

        public override void UpdatePatientRecord(Manage hospital_manager, string patient_name, string new_contact, string[] new_symptoms)
        {
            PatientRecord record_edit = hospital_manager.FindPatientRecordByName(patient_name);

            if (record_edit != null)
            {
                record_edit.Contact = new_contact;
                record_edit.Symptoms = new_symptoms;
                Console.WriteLine($"Patient record for {patient_name} has been updated.");
                hospital_manager.SaveRecordsToFile();
            }
            else
            {
                Console.WriteLine($"No patient record found for {patient_name}.");
            }
        }

        public override void Diagnose() { }

        public override void ProvideTreatment() { }
    }
}

```

```
using System;
using System.Collections.Generic;

namespace HospitalManagementSystem
{
    public class Hospital
    {
        private List<Patient> _patients = new List<Patient>();
        private List<MedicalStaff> _medical_staff = new List<MedicalStaff>();
        private List<Receptionist> _receptionists = new List<Receptionist>();

        public void AddPatient(Patient patient)
        {
            _patients.Add(patient);
            Console.WriteLine("Patient added to the hospital.");
        }

        public void RemovePatient(Patient patient)
        {
            _patients.Remove(patient);
            Console.WriteLine("Patient removed from the hospital.");
        }

        public void AddMedicalStaff(MedicalStaff staff)
        {
            _medical_staff.Add(staff);
            Console.WriteLine("Medical staff added.");
        }

        public void AddReceptionist(Receptionist receptionist)
        {
            _receptionists.Add(receptionist);
            Console.WriteLine("Receptionist added.");
        }
    }
}

```

```
namespace HospitalManagementSystem
{
    public interface IObserver
    {
        void Update(Patient patient);
    }
}

```

```
namespace HospitalManagementSystem
{
    public class Manage
    {
        private List<PatientRecord> _patient_records = new List<PatientRecord>();
        private string _file_path = "E:/COS20007/CustomProgram/HospitalRecords.txt";


        public Manage()
        {
            LoadRecordsFromFile();
        }

        public void AddPatientRecord(Patient patient, string treatment_plan, string assigned_staff)
        {
            PatientRecord new_record = new PatientRecord(patient.Name, patient.DOB, patient.Contact, patient.Symptoms, treatment_plan, assigned_staff);

            _patient_records.Add(new_record);
            Console.WriteLine($"Patient record for {patient.Name} has been added.");

            SaveRecordsToFile();
        }

        public void RemovePatientRecord(Patient patient)
        {
            PatientRecord record_remove = _patient_records.Find(r => r.Name == patient.Name);

            if (record_remove != null)
            {
                _patient_records.Remove(record_remove);
                Console.WriteLine($"Patient record for {patient.Name} has been removed.");
                SaveRecordsToFile();
            }
            else
            {
                Console.WriteLine($"No patient record found for {patient.Name}.");
            }
        }

        public void DisplayAllRecords()
```

```
        {
            if (_patient_records.Count == 0)
            {
                Console.WriteLine("No patient records available.");
                return;
            }

            Console.WriteLine("Displaying all patient records:");
            foreach (var record in _patient_records)
            {
                record.DisplayRecord();
                Console.WriteLine();
            }
        }


        public PatientRecord FindPatientRecordByName(string patient_name)
        {
            return _patient_records.Find(r => r.Name == patient_name);
        }

        public void SaveRecordsToFile() // Make this public for external saving
        {
            using (StreamWriter writer = new StreamWriter(_file_path))
            {
                foreach (var record in _patient_records)
                {
                    writer.WriteLine(record.Name);
                    writer.WriteLine(record.DateOfBirth.ToString("yyyy-MM-dd"));
                    writer.WriteLine(record.Contact);
                    writer.WriteLine(string.Join(",", record.Symptoms));
                    writer.WriteLine(record.TreatmentPlan);
                    writer.WriteLine(record.AssignedStaff);
                }
            }
            Console.WriteLine("Records saved to file.");
        }

        private void LoadRecordsFromFile()
        {
            if (File.Exists(_file_path))
            {
                using (StreamReader reader = new StreamReader(_file_path))
                {
                    while (!reader.EndOfStream)
                    {
                        string name = reader.ReadLine();
```

```
                        DateTime dob = DateTime.Parse(reader.ReadLine());
                        string contact = reader.ReadLine();
                        string[] symptoms = reader.ReadLine().Split(',');
                        string treatment_plan = reader.ReadLine();
                        string assigned_staff = reader.ReadLine();

                        PatientRecord record = new PatientRecord(name, dob, contact, symptoms, treatment_plan, assigned_staff);
                        _patient_records.Add(record);
                    }
                }
                Console.WriteLine("Records loaded from file.");
            }
            else
            {
                Console.WriteLine("No records file found, starting fresh.");
            }
        }
    }
}
```



```
namespace HospitalManagementSystem
{
    public abstract class MedicalStaff : Person
    {
        protected MedicalStaff(string name, string contact, DateTime dob, string working_hours, string experience)
            : base(name, contact, dob)
        {
            WorkingHours = working_hours;
            Experience = experience;
        }

        public string WorkingHours
        {
            get;
            set;
        }

        public string Experience
        {
            get;
            set;
        }

        public abstract void Diagnose();

        public abstract void ProvideTreatment();
    }
}
```

```
using HospitalManagementSystem;

public class MedicalStaffDecorator : MedicalStaff
{
    private MedicalStaff _medical_staff;
    private string _expertise;

    public MedicalStaffDecorator(MedicalStaff medical_staff, string expertise) : base(medical_staff.Name, medical_staff.Contact, medical_staff.DOB, medical_staff.WorkingHours, medical_staff.Experience)
    {
        _medical_staff = medical_staff;
        _expertise = expertise;
    }

    public override void Diagnose()
    {
        _medical_staff.Diagnose();
        Console.WriteLine($"Specialist in {_expertise} is diagnosing the patient.");
    }

    public override void ProvideTreatment()
    {
        _medical_staff.ProvideTreatment();
        Console.WriteLine($"Specialist in {_expertise} is providing treatment.");
    }
}

```

```
namespace HospitalManagementSystem
{
    public class MedicalStaffFactory
    {
        public MedicalStaff CreateStaff(string type, string name, string contact, DateTime dob, string working_hours, string experience)
        {
            switch (type)
            {
                case "SurgicalNurse":
                    return new SurgicalNurse(name, contact, dob, working_hours, experience);
                case "Surgeon":
                    return new Surgeon(name, contact, dob, working_hours, experience);
                case "Pediatrician":
                    return new Pediatric(name, contact, dob, working_hours, experience);
                case "EmergencyNurse":
                    return new EmergencyNurse(name, contact, dob, working_hours, experience);
                case "PediatricNurse":
                    return new PediatricNurse(name, contact, dob, working_hours, experience);
                case "Psychologist":
                    return new Psychologist(name, contact, dob, working_hours, experience);
                default:
                    throw new ArgumentException("Invalid staff type");
            }
        }
    }
}

```

```
namespace HospitalManagementSystem
{
    public class MildTreatment : TreatmentPlan
    {
        public string ExecuteTreatment(Patient patient)
        {
            return "Mild";
        }
    }
}
```



```
namespace HospitalManagementSystem
{
    public abstract class Nurse : MedicalStaff
    {
        protected Nurse(string name, string contact, DateTime dob, string working_hours, string experience):base(name, contact, dob, working_hours, experience)
        {
        }

        public abstract void UpdatePatientRecord(Manage hospital_manager, string patientName, string newContact, string[] newSymptoms);
    }
}
```

```
using System;
using System.Collections.Generic;

namespace HospitalManagementSystem
{
    public class Patient : Person
    {
        private List<IObserver> _observers = new List<IObserver>();

        public Patient(string name, DateTime dob, string contact, string[] symptoms):base(name, contact, dob)
        {
            Symptoms = symptoms;
        }

        public string[] Symptoms
        {
            get;
            set;
        }

        public void AttachObserver(IObserver observer)
        {
            if (!_observers.Contains(observer))
            {
                _observers.Add(observer);
            }
        }

        public void DetachObserver(IObserver observer)
        {
            if (_observers.Contains(observer))
            {
                _observers.Remove(observer);
            }
        }

        public void Notify()
        {
            foreach (var observer in _observers)
            {
                observer.Update(this); // Passing the patient data to the observers
            }
        }
    }
}

```

```
namespace HospitalManagementSystem
{
    public class PatientRecord
    {
        public PatientRecord(string name, DateTime dob, string contact,
          string[] symptoms, string treatment_plan, string assigned_staff)
        {
            Name = name;
            DateOfBirth = dob;
            Contact = contact;
            Symptoms = symptoms;
            TreatmentPlan = treatment_plan;
            AssignedStaff = assigned_staff;
        }

        public void DisplayRecord()
        {
            Console.WriteLine($"Name: {Name}");
            Console.WriteLine($"Date of Birth: {DateOfBirth:yyyy-MM-dd}");
            Console.WriteLine($"Contact: {Contact}");
            Console.WriteLine($"Symptoms: {string.Join(", ", Symptoms)}");
            Console.WriteLine($"Treatment Plan: {TreatmentPlan}");
            Console.WriteLine($"Assigned Medical Staff: {AssignedStaff}");
        }

        public string Name
        {
            get;
            set;
        }
        public DateTime DateOfBirth
        {
            get;
            set;
        }
        public string Contact
        {
            get;
            set;
        }
        public string[] Symptoms
        {
            get;
            set;
        }
        public string TreatmentPlan
        {
            get;
            set;
```

```
        }
        public string AssignedStaff
        {
            get;
            set;
        }
    }
}

```



```
namespace HospitalManagementSystem
{
    public class Pediatric : Doctor
    {
        public Pediatric(string name, string contact, DateTime dob, string workingHours, string experience):base(name, contact, dob, workingHours, experience)
        {
        }


        public override void Diagnose()
        {
            Console.WriteLine($"{Name} (Pediatrician) is diagnosing a child patient.");
        }

        public override void ProvideTreatment()
        {
            Console.WriteLine($"{Name} (Pediatrician) is providing treatment.");
        }
    }
}

```

```
namespace HospitalManagementSystem
{
    public class PediatricNurse : Nurse
    {
        public PediatricNurse(string name, string contact, DateTime dob, string working_hours, string experience):base(name, contact, dob, working_hours, experience)
        {
        }

        public override void UpdatePatientRecord(Manage hospital_manager, string patient_name, string new_contact, string[] new_symptoms)
        {
            PatientRecord recordToEdit = hospital_manager.FindPatientRecordByName(patient_name);

            if (recordToEdit != null)
            {
                recordToEdit.Contact = new_contact;
                recordToEdit.Symptoms = new_symptoms;
                Console.WriteLine($"Patient record for {patient_name} has been updated.");
                hospital_manager.SaveRecordsToFile();  // Save updated records to file
            }
            else
            {
                Console.WriteLine($"No patient record found for {patient_name}.");
            }
        }

        public override void Diagnose() { }

        public override void ProvideTreatment() { }
    }
}

```

```
namespace HospitalManagementSystem
{
    public abstract class Person
    {
        protected Person(string name, string contact, DateTime dob)
        {
            Name = name;
            Contact = contact;
            DOB = dob;
        }

        public string Name
        {
            get;
            set;
        }

        public string Contact
        {
            get;
            set;
        }

        public DateTime DOB
        {
            get;
            set;
        }

    }
}


```

```
namespace HospitalManagementSystem
{
    public class Psychologist : Doctor
    {
        public Psychologist(string name, string contact, DateTime dob, string working_hours, string experience):base(name, contact, dob, working_hours, experience)
        {
        }


        public override void Diagnose()
        {
            Console.WriteLine($"{Name} (Psychologist) is diagnosing a mental health condition.");
        }

        public override void ProvideTreatment()
        {
            Console.WriteLine($"{Name} (Psychologist) is providing mental health treatment.");
        }
    }
}

```



```
namespace HospitalManagementSystem
{
    public class Receptionist : Person
    {
        public Receptionist(string name, string contact, DateTime dob):base (name, contact, dob)
        {
        }

        public void AddPatientRecord()
        {
            Console.WriteLine($"{Name} is adding a patient record to the system.");
        }
    }
}

```

```
using System;

namespace HospitalManagementSystem
{
    public class Surgeon : MedicalStaff
    {

        public Surgeon(string name, string contact, DateTime dob, string workingHours, string experience):base(name, contact, dob, workingHours, experience)
        {
        }

        public override void Diagnose()
        {
            Console.WriteLine($"{Name} is diagnosing the patient.");
        }

        public override void ProvideTreatment()
        {
            Console.WriteLine($"{Name} is performing surgery.");
        }
    }
}

```

```
namespace HospitalManagementSystem
{
    public class SurgicalNurse : Nurse
    {
        public SurgicalNurse(string name, string contact, DateTime dob, string working_hours, string experience) : base(name, contact, dob, working_hours, experience)
        {
        }

        public override void UpdatePatientRecord(Manage hospital_manager, string patient_name, string new_contact, string[] new_symptoms)
        {
            PatientRecord recordToEdit = hospital_manager.FindPatientRecordByName(patient_name);

            if (recordToEdit != null)
            {
                recordToEdit.Contact = new_contact;
                recordToEdit.Symptoms = new_symptoms;
                Console.WriteLine($"Patient record for {patient_name} has been updated.");
                hospital_manager.SaveRecordsToFile();  // Save updated records to file
            }
            else
            {
                Console.WriteLine($"No patient record found for {patient_name}.");
            }
        }

        public override void Diagnose() { }
        public override void ProvideTreatment() { }
    }
}

```

```
namespace HospitalManagementSystem
{
    public interface TreatmentPlan
    {
        string ExecuteTreatment(Patient patient);
    }
}

```

```
using HospitalManagementSystem;

class Program
{
    static void Main(string[] args)
    {
        Manage hospital_manager = new Manage();
        MedicalStaffFactory staff_factory = new MedicalStaffFactory();
        Receptionist receptionist = new Receptionist("Alice", "1234567890", new DateTime(1990, 5, 12));
        SurgicalNurse surgical_nurse = new SurgicalNurse("Nurse Mary", "4567891230", new DateTime(1982, 4, 15), "9 AM - 5 PM", "10 years");

        while (true)
        {
            Console.WriteLine("\nWelcome to the Hospital Management System");
            Console.WriteLine("1 - Add Patient Record");
            Console.WriteLine("2 - Remove Patient Record");
            Console.WriteLine("3 - Edit Patient Record");
            Console.WriteLine("4 - Display All Patient Records");
            Console.WriteLine("5 - Exit");

            string option = Console.ReadLine();

            switch (option)
            {
                case "1":
                    Console.Write("Name: ");
                    string name = Console.ReadLine();

                    Console.Write("Date of Birth (yyyy-mm-dd): ");
                    DateTime dob = DateTime.Parse(Console.ReadLine());

                    Console.Write("Contact Number: ");
                    string contact = Console.ReadLine();

                    Console.WriteLine("Please enter your symptoms (comma-separated): ");
                    string[] symptoms = Console.ReadLine().Split(',');

                    Patient patient1 = new Patient(name, dob, contact, symptoms);


                    // Notify observers (Doctor) about the new patient record
                    Doctor doctor = new Doctor("Dr. Smith", "9876543210",
```

```
                    new DateTime(1970, 6, 15), "9 AM - 5 PM", "15 years");
                    patient1.AttachObserver(doctor);  // Attach observer to patient

                    Console.WriteLine("Your information has been added to the hospital records.");

                    Console.WriteLine("Please select a treatment plan: 1 - Mild, 2 - Aggressive");
                    int treatment_plan_choice;
                    while (!int.TryParse(Console.ReadLine(), out treatment_plan_choice) || (treatment_plan_choice != 1 && treatment_plan_choice != 2))
                    {
                        Console.WriteLine("Invalid input. Please choose either 1 (Mild) or 2 (Aggressive): ");
                    }

                    TreatmentPlan treatment_plan;
                    MedicalStaff assigned_staff;

                    // Strategy and Factory Patterns: Based on the treatment plan, assign the appropriate medical staff
                    if (treatment_plan_choice == 1)
                    {
                        treatment_plan = new MildTreatment();
                        assigned_staff = staff_factory.CreateStaff("PediatricNurse", "Nurse Nina", "7890123456", new DateTime(1985, 3, 21), "9 AM - 5 PM", "10 years");
                        hospital_manager.AddPatientRecord(patient1, treatment_plan.ExecuteTreatment(patient1), assigned_staff.Name);
                    }
                    else
                    {
                        treatment_plan = new AggressiveTreatment();
                        assigned_staff = staff_factory.CreateStaff("Surgeon", "Dr. John", "9876543210", new DateTime(1975, 3, 10), "24/7", "15 years");
                        hospital_manager.AddPatientRecord(patient1, treatment_plan.ExecuteTreatment(patient1), assigned_staff.Name);
                    }

                    // Decorator Pattern: Add special abilities or expertise to the assigned medical staff
                    assigned_staff = new MedicalStaffDecorator(assigned_staff, "Cardiology Expertise");

```

```
                    // Display the assigned staff and begin treatment
                    Console.WriteLine($"You have been assigned to: {assigned_staff.GetType().Name} {assigned_staff.Name}");
                    Console.WriteLine("Your treatment plan is: " + treatment_plan.ExecuteTreatment(patient1));

                    // The assigned staff provides diagnosis and treatment
                    assigned_staff.Diagnose();
                    assigned_staff.ProvideTreatment();

                    // Update the patient record and notify doctor (Observer Pattern)
                    patient1.Notify();




                    break;
                case "2":
                    Console.Write("Enter the patient's name to remove: ");
                    string name_to_remove = Console.ReadLine();
                    hospital_manager.RemovePatientRecord(new Patient(name_to_remove, DateTime.MinValue, "", null));
                    break;

                case "3":
                    Console.Write("Enter the patient's name to edit: ");
                    string name_to_edit = Console.ReadLine();

                    Console.Write("New Contact Number: ");
                    string new_contact = Console.ReadLine();

                    Console.WriteLine("New Symptoms (comma-separated): ");
                    string[] new_symptoms = Console.ReadLine().Split(',');


                    surgical_nurse.UpdatePatientRecord(hospital_manager, name_to_edit, new_contact, new_symptoms);
                    break;

                case "4":
                    hospital_manager.DisplayAllRecords();
                    break;

                case "5":
                    return;

                default:
                    Console.WriteLine("Invalid option. Try again.");
```

```
                    break;
                }
            }
        }
    }
```

```
E:\COS20007\CustomProgram

Records loaded from file.

Welcome to the Hospital Management System
1 - Add Patient Record
2 - Remove Patient Record
3 - Edit Patient Record
4 - Display All Patient Records
5 - Exit
1
Name: thang
Date of Birth (yyyy-mm-dd): 2005-11-30
Contact Number: 0000000000
Please enter your symptoms (comma-separated):
sore throat, headache
Your information has been added to the hospital records.
Please select a treatment plan: 1 - Mild, 2 - Aggressive
1
Patient record for thang has been added.
Records saved to file.
You have been assigned to: MedicalStaffDecorator Nurse Nina
Your treatment plan is: Mild
Specialist in Cardiology Expertise is diagnosing the patient.
Specialist in Cardiology Expertise is providing treatment.
Doctor Dr. Smith has been notified of the patient's update: thang
Updated Symptoms: sore throat,  headache

Welcome to the Hospital Management System
1 - Add Patient Record
2 - Remove Patient Record
3 - Edit Patient Record
4 - Display All Patient Records
5 - Exit
2
Enter the patient's name to remove: thang
Patient record for thang has been removed.
Records saved to file.
```

```
E:\COS20007\CustomProgram

Records loaded from file.

Welcome to the Hospital Management System
1 - Add Patient Record
2 - Remove Patient Record
3 - Edit Patient Record
4 - Display All Patient Records
5 - Exit
1
Name: thang
Date of Birth (yyyy-mm-dd): 2005-11-30
Contact Number: 0000000000
Please enter your symptoms (comma-separated):
sore throat, headache
Your information has been added to the hospital records.
Please select a treatment plan: 1 - Mild, 2 - Aggressive
2
Patient record for thang has been added.
Records saved to file.
You have been assigned to: MedicalStaffDecorator Dr. John
Your treatment plan is: Aggressive
Dr. John is diagnosing the patient.
Specialist in Cardiology Expertise is diagnosing the patient.
Dr. John is performing surgery.
Specialist in Cardiology Expertise is providing treatment.
Doctor Dr. Smith has been notified of the patient's update: thang
Updated Symptoms: sore throat,  headache

Welcome to the Hospital Management System
1 - Add Patient Record
2 - Remove Patient Record
3 - Edit Patient Record
4 - Display All Patient Records
5 - Exit
3
Enter the patient's name to edit: thang
New Contact Number: 0123456789
New Symptoms (comma-separated):
leg pain
Patient record for thang has been updated.
Records saved to file.
```

Program.cs | Psychologist.cs | Receptionist.cs | **HospitalRecords.txt**

```
thang
2005-11-30
0123456789
leg pain
Aggressive
Dr. John
```