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

```

```csharp
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.");
        }
    }

}

```

```csharp
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);
    }
}

```

```csharp
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");
            }
        }
    }
}

```

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


```

```csharp
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;
        }

    }
}


```

```csharp
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.");
        }
    }
}

```

```csharp
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.");
        }
    }
}

```

```csharp
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() { }
    }
}

```

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

```

```csharp
using HospitalManagementSystem;

class Program
{
    static void Main(string[] args)
    {
        Manage hospitalManager = new Manage();
        MedicalStaffFactory staffFactory = new MedicalStaffFactory();
        Receptionist receptionist = new Receptionist("Alice", "1234567890", new DateTime(1990, 5, 12));
        SurgicalNurse surgicalNurse = 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 newPatient = 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");
                    newPatient.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 treatmentPlanChoice;
                    while (!int.TryParse(Console.ReadLine(), out treatmentPlanChoice) || (treatmentPlanChoice != 1 && treatmentPlanChoice != 2))
                    {
                        Console.WriteLine("Invalid input. Please choose either 1 (Mild) or 2 (Aggressive): ");
                    }

                    TreatmentPlan treatmentPlan;
                    MedicalStaff assignedStaff;

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

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

```

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

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

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




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

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

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

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

                    surgicalNurse.UpdatePatientRecord(hospitalManager, nameToEdit, newContact, newSymptoms);
                    break;

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

                case "5":
                    return;

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

```