**Program.cs**

using ClinicModelLibrary;

using ClinicBLLibrary;

namespace ClinicApp

{

internal class Program

{

IClinicService clinicService;

public Program()

{

clinicService = new ClinicService();

}

void displayAdminMenu()

{

Console.WriteLine(" 1.Add Doctor");

Console.WriteLine(" 2.Modify Doctor Phone");

Console.WriteLine(" 3.Modify Doctor Experience");

Console.WriteLine(" 4.Delete Doctor");

Console.WriteLine(" 5.Print All Doctors");

Console.WriteLine(" 0. Exit ");

}

void StartAdminActivities()

{

int choice;

do

{

displayAdminMenu();

choice = Convert.ToInt32(Console.ReadLine());

switch (choice)

{

case 0:

Console.WriteLine("Bye bye");

break;

case 1:

AddDoctor();

break;

case 2:

UpdatePhone();

break;

case 3:

UpdateExperience();

break;

case 4:

DeleteDoctor();

break;

case 5:

PrintAllDoctors();

break;

default:

Console.WriteLine("Invalid choice. Try again");

break;

}

} while (choice != 0);

}

private void PrintAllDoctors()

{

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

try

{

var doctors = clinicService.GetDoctors();

foreach (var item in doctors)

{

Console.WriteLine(item);

Console.WriteLine("-------------------------------");

}

}

catch (NoDoctorsException e)

{

Console.WriteLine(e.Message);

} Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

void AddDoctor()

{

try

{

Doctor doctor = TakeDoctorDetails();

var result = clinicService.AddDoctor(doctor);

if (result != null)

{

Console.WriteLine("Doctor added");

}

}

catch (FormatException e)

{

Console.WriteLine(e.Message);

}

catch (NotAddedException e)

{

Console.WriteLine(e.Message);

}

}

Doctor TakeDoctorDetails()

{

Doctor doctor = new Doctor();

Console.WriteLine("Enter Doctor Name");

doctor.Name = Console.ReadLine();

Console.WriteLine("Enter Doctor Qualification");

doctor.Qualification = Console.ReadLine();

Console.WriteLine("Enter Doctor Specialization");

doctor.Specialization = Console.ReadLine();

Console.WriteLine("Enter Doctor Experience");

doctor.Experience = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Enter Doctor Fee");

doctor.Fee = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Enter Doctor Mobile Number");

doctor.Phone = Convert.ToDouble(Console.ReadLine());

return doctor;

}

int GetProductIdFromUser()

{

int id;

Console.WriteLine("Please enter the product id");

id = Convert.ToInt32(Console.ReadLine());

return id;

}

private void DeleteDoctor()

{

try

{

int id = GetProductIdFromUser();

if (clinicService.Delete(id) != null)

Console.WriteLine("Doctor deleted");

}

catch (NoSuchDoctorException e)

{

Console.WriteLine(e.Message);

}

}

private void UpdateExperience()

{

var id = GetProductIdFromUser();

Console.WriteLine("Please enter the new experience");

int experience = Convert.ToInt32(Console.ReadLine());

Doctor doctor = new Doctor();

doctor.Experience = experience;

doctor.Id = id;

try

{

var result = clinicService.UpdateDoctorExperience(id, experience);

if (result != null)

Console.WriteLine("Update success");

}

catch (NoSuchDoctorException e)

{

Console.WriteLine(e.Message);

}

}

private void UpdatePhone()

{

var id = GetProductIdFromUser();

Console.WriteLine("Please enter the new phone number");

double phone = Convert.ToDouble(Console.ReadLine());

Doctor doctor = new Doctor();

doctor.Phone = phone;

doctor.Id = id;

try

{

var result = clinicService.UpdateDoctorPhone(id, phone);

if (result != null)

Console.WriteLine("Update success");

}

catch (NoSuchDoctorException e)

{

Console.WriteLine(e.Message);

}

}

static int Main(string[] args)

{

Program program = new Program();

program.StartAdminActivities();

Console.WriteLine("Welcome to Clinic!!");

return 0;

}

}

}

ClinicModelLibrary

using ShoppingDALLibrary;

using ShoppingModelLibrary;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ShoppingBLLibrary

{

public class CustomerService : ICustomerService

{

IRepository<string, Customer> repository;

public CustomerService()

{

repository = new CustomerRepository();

}

public bool Login(string username, string password)

{

var myCustomer = repository.GetById(username);

if (myCustomer != null)

{

if (myCustomer.ComparePassword(password))

return true;

}

return false;

}

public Customer Register(Customer customer)

{

var result = repository.Add(customer);

if (result != null)

{

return result;

}

throw new UnableToRegisterCustomerException();

}

}

}

**ClinicDALLibrary**

IClinicRepository.cs

using ClinicModelLibrary;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClinicDALLibrary

{

public interface IClinicRepository

{

public Doctor Add(Doctor doctor);

public Doctor Update(Doctor doctor);

public Doctor Delete(int id);

public Doctor GetById(int id);

public List<Doctor> GetAll();

}

}

**ClinicRepository.cs**

using ClinicModelLibrary;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClinicDALLibrary

{

public class ClinicRepository : IClinicRepository

{

Dictionary<int, Doctor> doctors = new Dictionary<int, Doctor>();

/// <summary>

///

/// </summary>

/// <param name="doctor">Doctor object that has to be added</param>

/// <returns>The doctor object that has been added</returns>

/// <exception cref="NotImplementedException"></exception>

public Doctor Add(Doctor doctor)

{

int id = GetTheNextId();

try

{

doctor.Id = id;

doctors.Add(doctor.Id, doctor);

return doctor;

}

catch (ArgumentException e)

{

Console.WriteLine("The doctor Id already exists");

Console.WriteLine(e.Message);

}

return null;

}

private int GetTheNextId()

{

if (doctors.Count == 0)

return 1;

int id = doctors.Keys.Max();

return ++id;

}

/// <summary>

/// Deletes the doctor from the dictionary using the id as key

/// </summary>

/// <param name="id">The Id of the docotr to be deleted</param>

/// <returns>The deleted product</returns>

public Doctor Delete(int id)

{

var doctor = doctors[id];

doctors.Remove(id);

return doctor;

}

public List<Doctor> GetAll()

{

var doctorList = doctors.Values.ToList();

return doctorList;

}

/// <summary>

///

/// </summary>

/// <param name="id">The id of the Doctors are getting</param>

/// <returns> returns the doctors using id</returns>

public Doctor GetById(int id)

{

return doctors[id];

}

// <summary>

///

/// </summary>

/// <param name="product"> Doctors updated using id from the dictionary</param>

/// <returns>Returns the updated Doctor</returns>

public Doctor Update(Doctor doctor)

{

doctors[doctor.Id] = doctor;

return doctors[doctor.Id];

}

}

}

**ClinicBLLibrary**

**ICLinicService.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using ClinicModelLibrary;

using ClinicDALLibrary;

namespace ClinicBLLibrary

{

public interface IClinicService

{

public Doctor AddDoctor(Doctor doctor);

public Doctor UpdateDoctorPhone(int id, double phone);

public Doctor GetDoctor(int id);

public List<Doctor> GetDoctors();

public Doctor UpdateDoctorExperience(int id, int experience);

public Doctor Delete(int id);

}

}

**ClinicService.cs**

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using ClinicDALLibrary;

using ClinicModelLibrary;

namespace ClinicBLLibrary

{

public class ClinicService : IClinicService

{

IClinicRepository repository;

public ClinicService()

{

repository = new ClinicRepository();

}

// <summary>

/// Adds the doctor to the collection using the repository

/// </summary>

/// <param name="doctor">The doctor to be added</param>

/// <returns></returns>

/// <exception cref="NotAddedException">Doctor Id duplicated</exception>

public Doctor AddDoctor(Doctor doctor)

{

var result = repository.Add(doctor);

if (result != null)

return result;

throw new NotAddedException();

}

public Doctor Delete(int id)

{

var product = GetDoctor(id);

if (product != null)

{

repository.Delete(id);

return product;

}

throw new NoSuchDoctorException();

}

/// <summary>

/// Returns the doctor for the given Id

/// </summary>

/// <param name="id">Id of the doctor to be returned</param>

/// <returns></returns>

/// <exception cref="NoSuchDoctorException">No product with the given Id</exception>

public Doctor GetDoctor(int id)

{

var result = repository.GetById(id);

return result == null ? throw new NoSuchDoctorException() : result;

}

public List<Doctor> GetDoctors()

{

var doctors = repository.GetAll();

if (doctors.Count != 0)

return doctors;

throw new NoDoctorsAvailableException();

}

public Doctor UpdateDoctorExperience(int id, int experience)

{

var doctor = GetDoctor(id);

if (doctor != null)

{

doctor.Experience = experience;

var result = repository.Update(doctor);

return result;

}

throw new NoSuchDoctorException();

}

public Doctor UpdateDoctorPhone(int id, double phone)

{

var doctor = GetDoctor(id);

if (doctor != null)

{

doctor.Phone = phone;

var result = repository.Update(doctor);

return result;

}

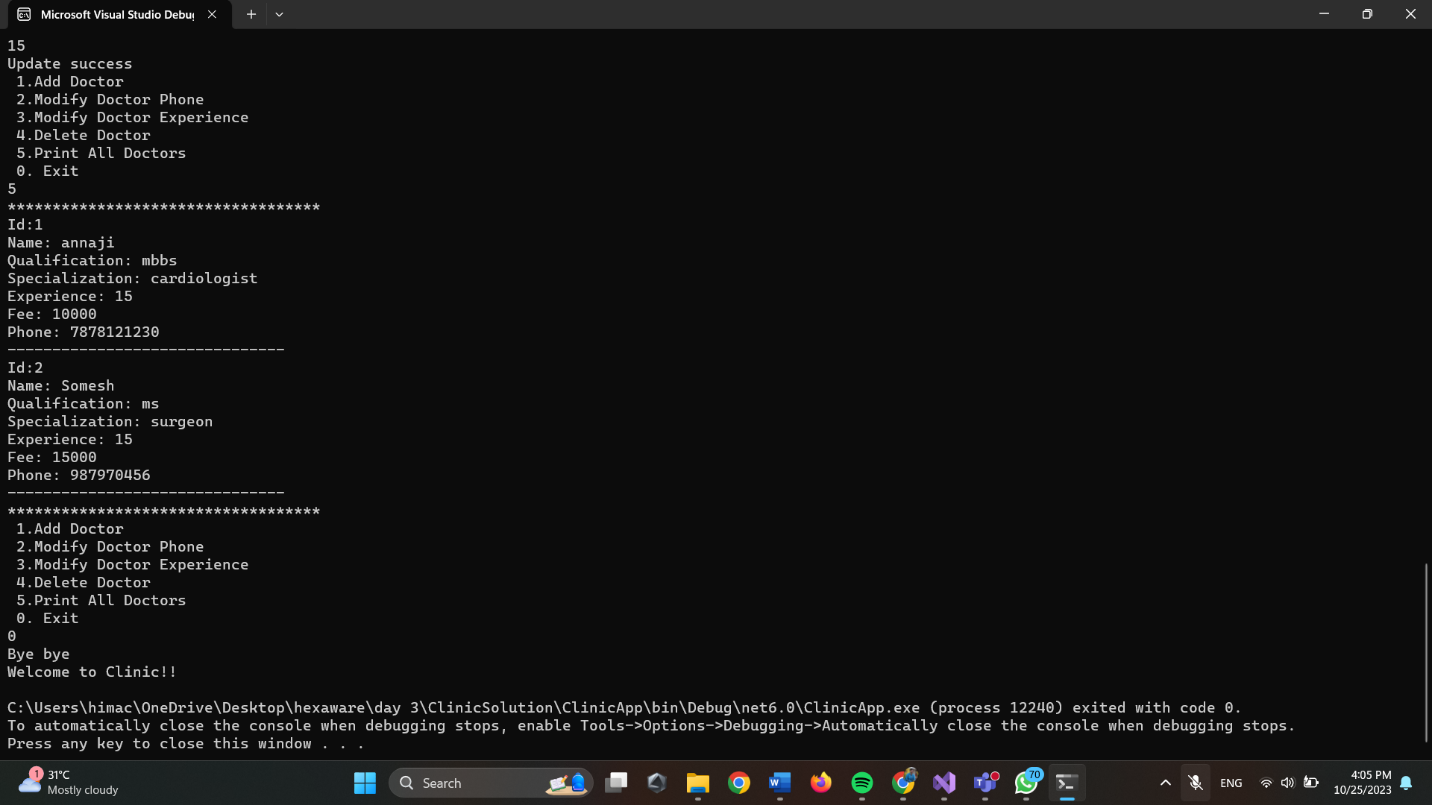
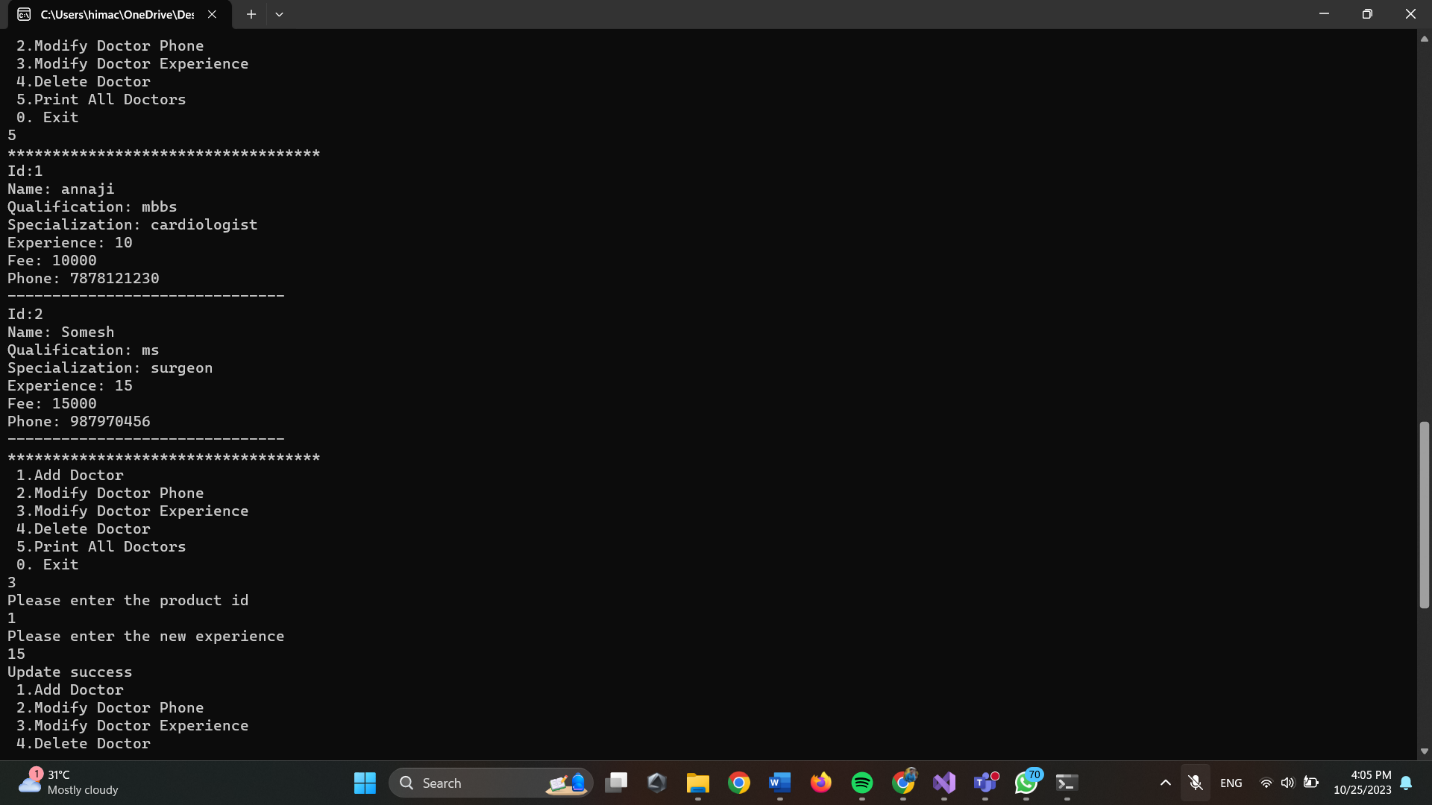
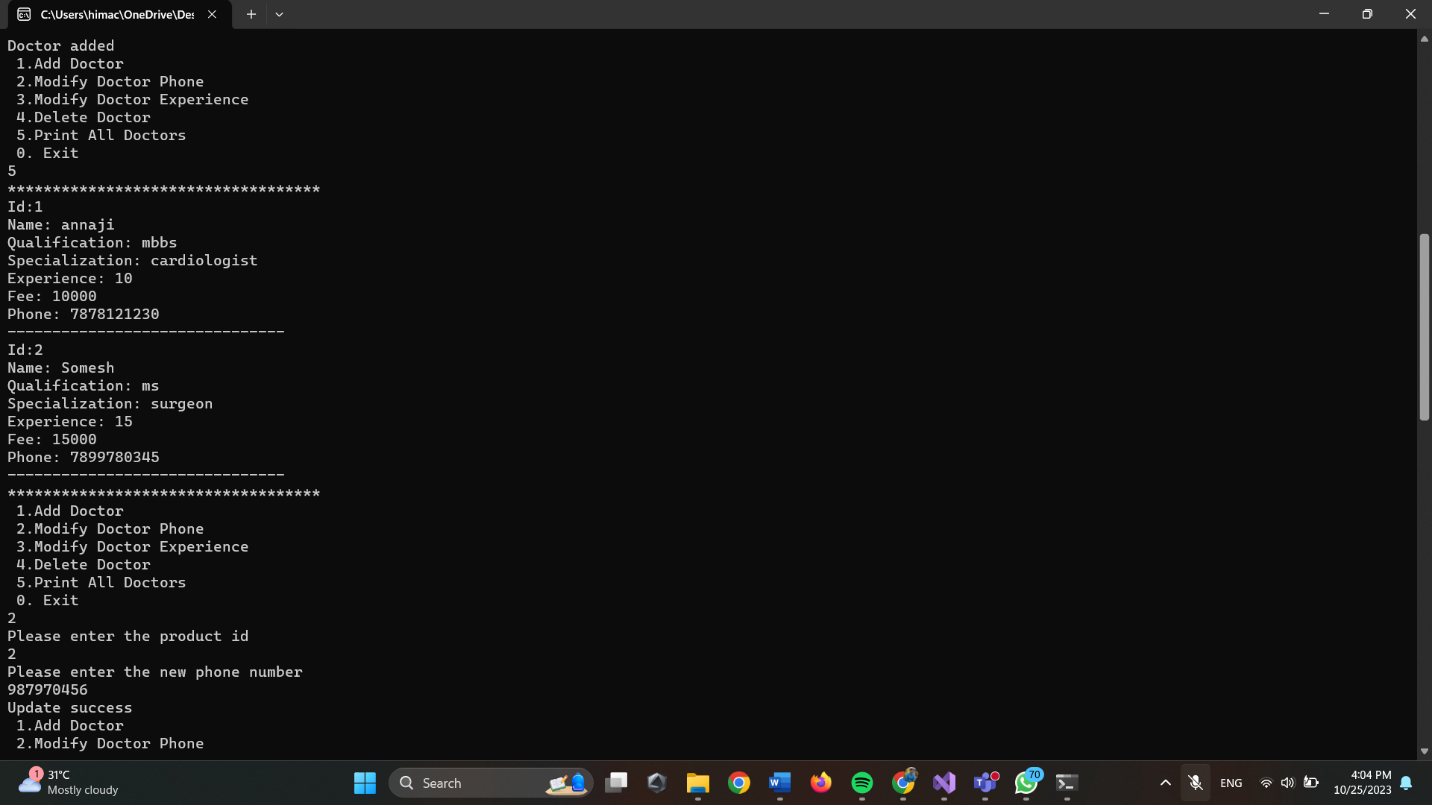
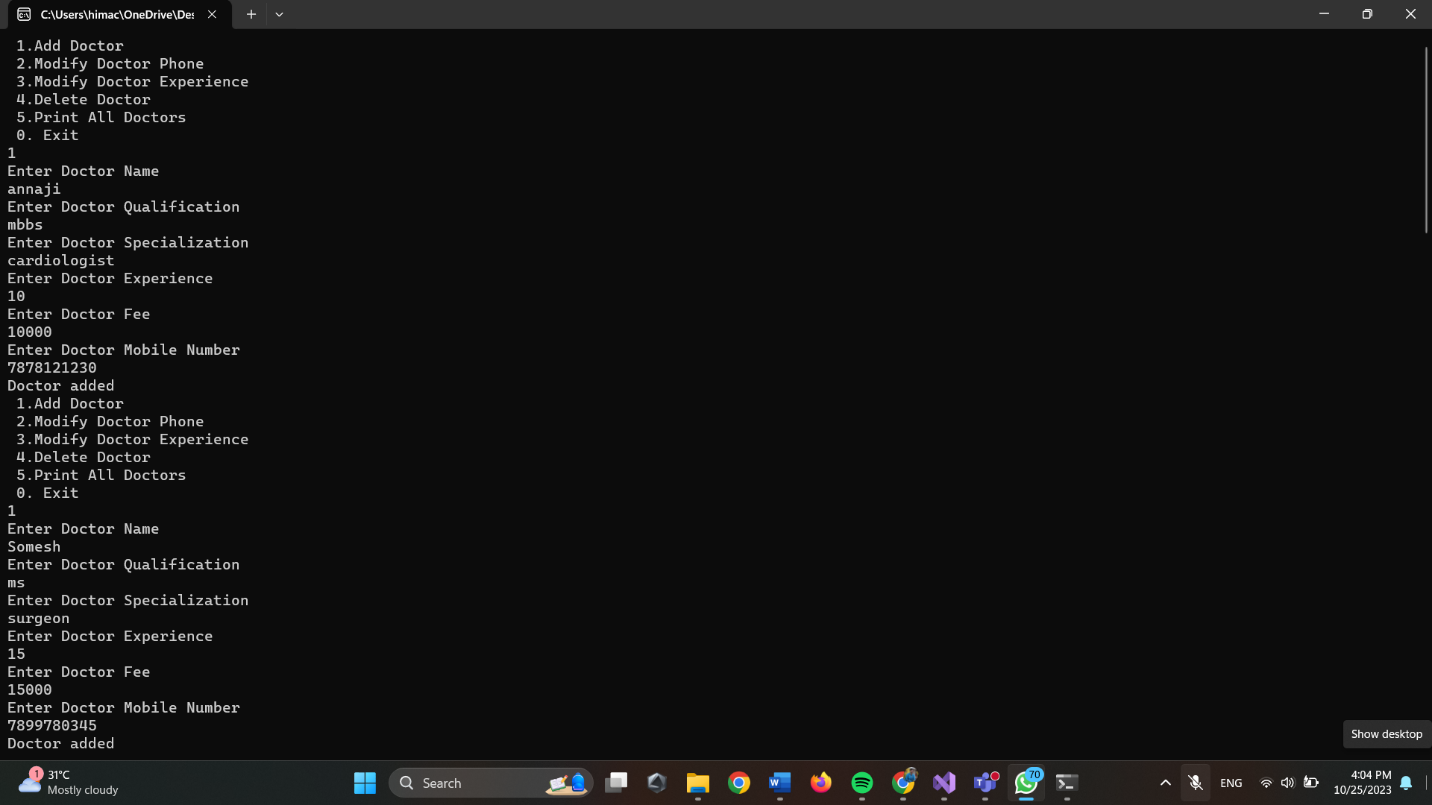
throw new NoSuchDoctorException();

}

}

}

**Output**

****A screenshot of a computer

Description automatically generated