DoctorClinicApp (Home.cs) :

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using ClinicBLLibrary;

using ClinicModelLibrary;

namespace DoctorClinicApp

{

internal class Home

{

IClinicService clinicService;

public Home()

{

clinicService = new ClinicService();

}

void DisplayAdminMenu()

{

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

Console.WriteLine("2. Modify Doctor's Phone No.");

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

Console.WriteLine("4. Show all doctors");

Console.WriteLine("5. Delete Doctor Profile");

Console.WriteLine("0. Exit");

Console.WriteLine("Enter Your Choice:");

}

void StartAdminActivities()

{

int choice; do

{

DisplayAdminMenu();

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

switch (choice)

{

case 0:

Console.WriteLine("Get Well Soon");

break;

case 1:

AddDoctor();

break;

case 2:

DoctorPhoneUpdate();

break;

case 3:

DoctorExperienceUpdate();

break;

case 4:

ShowAllDoctors();

break;

case 5:

RemoveDoctor();

break;

default:

Console.WriteLine("Not Valid. Try again");

break;

}

} while (choice != 0);

}

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("Please enter doctor name");

doctor.Name = Console.ReadLine(); thispoint:

Console.WriteLine("Please enter doctor's phone number ");

doctor.DoctorNumber = Convert.ToInt64(Console.ReadLine());

string numberString = doctor.DoctorNumber.ToString();

int numberOfDigits = numberString.Length;

if (numberOfDigits != 10)

{

Console.WriteLine("Please enter a valid number");

goto thispoint;

}

Console.WriteLine("Please enter doctor speciality");

doctor.DoctorSpeciality = Console.ReadLine();

Console.WriteLine("Please enter doctor experience in years");

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

return doctor;

}

int GetDoctorIdFromUser()

{

int id;

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

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

return id;

}

private void DoctorPhoneUpdate()

{

var id = GetDoctorIdFromUser();

Console.WriteLine("Please enter the new Phone Number");

long doctorNumber = Convert.ToInt64(Console.ReadLine());

Doctor doctor = new Doctor();

doctor.DoctorNumber = doctorNumber;

doctor.Id = id;

try

{

var result = clinicService.UpdateContactNumber(id, doctorNumber);

if (result != null)

Console.WriteLine("Updated");

}

catch (NoSuchDoctorException e)

{

Console.WriteLine(e.Message);

}

}

private void DoctorExperienceUpdate()

{

var id = GetDoctorIdFromUser();

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

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

Doctor doctor = new Doctor();

doctor.DoctorExperience = doctorExperience;

doctor.Id = id; try

{

var result = clinicService.UpdateExperience(id,doctorExperience);

if (result != null)

Console.WriteLine("Updated");

}

catch (NoSuchDoctorException e)

{

Console.WriteLine(e.Message);

}

}

private void ShowAllDoctors()

{

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

try

{

var doctors = clinicService.GetDoctor();

foreach (var item in doctors)

{

Console.WriteLine(item);

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

}

}

catch (NoSuchDoctorException e) {

}

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

}

private void RemoveDoctor()

{

try

{

int id = GetDoctorIdFromUser();

if (clinicService.Delete(id) != null) Console.WriteLine("Doctor Remove");

}

catch (NoSuchDoctorException e)

{

Console.WriteLine(e.Message);

}

}

static void Main(string[] args)

{

Console.WriteLine("Welcome To The Amma Clinic");

Home home = new Home();

home.StartAdminActivities();

}

}

}

ClinicBLLibrary (ClinicServices.cs):

using ClinicDALLibrary;

using ClinicModelLibrary;

using System.Numerics;

namespace ClinicBLLibrary

{

public class ClinicService : IClinicService

{

IRepository repository;

public ClinicService()

{

repository = new ProjectRepository();

}

public Doctor AddDoctor(Doctor doctor)

{

var result = repository.Add(doctor);

if (result != null) return result;

throw new NotAddedException();

}

public Doctor Delete(int id)

{

var doctor = GetDoctor(id);

if (doctor != null)

{

repository.Delete(id);

return doctor;

}

throw new NoSuchDoctorException();

}

public Doctor GetDoctor(int id)

{

var result = repository.GetById(id);

return result ?? throw new NoSuchDoctorException();

}

public List<Doctor> GetDoctor()

{

var doctors = repository.GetAll();

if (doctors.Count != 0)

return doctors;

throw new NoDoctorsAvailableException();

}

public Doctor UpdateContactNumber(int id, long phone number)

{

var doctor = GetDoctor(id); if (doctor != null)

{

string numberString = doctor.DoctorNumber.ToString();

int numberOfDigits = numberString.Length;

if (numberOfDigits == 10)

{

doctor.DoctorNumber = phonenumber;

var result = repository.Update(doctor);

return doctor;

}

else

{

Console.WriteLine("Invalid choice");

}

}

throw new NoSuchDoctorException();

}

public Doctor UpdateExperience(int id, int doctorexperience)

{

var doctor = GetDoctor(id);

if (doctor != null)

{

if (doctor.DoctorExperience > 0)

{

doctor.DoctorExperience = doctorexperience;

var result = repository.Update(doctor);

return doctor;

}

else

{

Console.WriteLine("Invalid choice");

}

}

throw new NoSuchDoctorException();

}

}

}

ClinicBLLibrary (IClinicServices.cs) :

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using ClinicModelLibrary;

using ClinicDALLibrary;

using System.Numerics;

namespace ClinicBLLibrary

{

public interface IClinicService

{

public Doctor AddDoctor(Doctor doctor); public Doctor Delete(int id); public Doctor GetDoctor(int id); public List<Doctor> GetDoctor();

public Doctor UpdateContactNumber(int id, long phone number); public Doctor UpdateExperience(int id, int doctor experience);

}

}

ClinicBLLibrary (NoDoctorAvailableException) :

using System.Runtime.Serialization;

namespace ClinicBLLibrary

{

[Serializable]

public class NoDoctorsAvailableException : Exception

{

string message;

public NoDoctorsAvailableException()

{

message = "No doctors are available currently";

}

public override string Message => message;

}

}

ClinicBLLibrary(NoSuchDoctorException) :

using System.Runtime.Serialization;

namespace ClinicBLLibrary

{

[Serializable]

public class NoSuchDoctorException : Exception

{

string message;

public NoSuchDoctorException()

{

message = "The doctor with the entered id is not present";

}

public override string Message => message;

}

}

ClinicBLLibrary(NotAddedException) :

using System.Runtime.Serialization;

namespace ClinicBLLibrary

{

[Serializable]

public class NotAddedException : Exception

{

string message; public NotAddedException()

{

message = "Doctor was not addedd.";

}

public override string Message => message;

}

}

ClinicDALLibrary (ProjectRepository.cs) :

using ClinicModelLibrary;

namespace ClinicDALLibrary

{

public class ProjectRepository : IRepository

{

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

/// <summary>

/// Adds the given doctor to the dictionary

/// </summary>

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

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

///

public Doctor Add(Doctor doctor)

{

int id = GetNextID();

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 GetNextID()

{

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 doctor to be removed</param>

/// <returns>The removed doctor</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;

}

public Doctor GetById(int id)

{

if (doctors.ContainsKey(id)) return doctors[id]; return null;

}

public Doctor Update(Doctor doctor)

{

doctors[doctor.Id] = doctor; return doctors[doctor.Id];

}

}

}

ClinicDALLibrary - (Irepository.cs) :

using System;

using System.Collections.Generic;

using System.Linq;

using System.Numerics;

using System.Text;

using System.Threading.Tasks;

using ClinicModelLibrary;

namespace ClinicDALLibrary

{

public interface IRepository

{

public Doctor Add(Doctor doctor); public Doctor Update(Doctor doctor); public Doctor Delete(int id); public Doctor GetById(int id); public List<Doctor> GetAll();

}

}

ClinicModelLibrary - (Doctor.cs) :

using System.Xml.Linq;

namespace ClinicModelLibrary

{

public class Doctor

{

public int Id { get; set; }

public string Name { get; set; } = String.Empty; public long DoctorNumber { get; set; }

public string DoctorSpeciality { get; set; } = String.Empty; public int DoctorExperience { get; set; }

public Doctor()

{

DoctorNumber = 0;

DoctorExperience = 0;

}

/// <summary>

/// Construct essential object properties

/// </summary>

/// <param name="id">ID of the doctor</param>

/// <param name="name">name of the doctor</param>

/// <param name="doctorNumber">Doctor's phone number</param>

/// <param name="doctorSpeciality">Doctor's speciality</param>

/// <param name="doctorExperience">Doctor's experience in years</param>

public Doctor(int id, string name, long doctorNumber, string doctorSpeciality, int doctorExperience)

{

Id = id;

Name = name;

DoctorNumber = doctorNumber;

DoctorSpeciality = doctorSpeciality;

DoctorExperience = doctorExperience;

}

public override string ToString()

{

return $"Doctor Id : {Id}\nDoctor Name : {Name}\nDoctor's Phone Number :{DoctorNumber}\nDoctor's Speciality : {DoctorSpeciality}" +

$"\nDoctor's Experience : {DoctorExperience}";

}

}

}













