MVC

Doctor - id,name,exp(int),Speciality,Phone,email,Pic

Patient - id,name,age,phone,email

Appointment -appoitmentNumber, doctorid,patientid,date+time

9-11

Create the appilication

create the models

Ensure relational integrity

apply migration

Seed/Initialize data - > Context

11-12

Repository - Doctor patient and appointments

Interface -> Generic

12-2:30

Service

DoctorService -> adding doctor, Updating Doctor Exp, Updating doctor Speciality, Updating Doctor phone

PatientServic-> AddPatient, UpdatePatientPhone

AppointmentService->BookAppointment, CancelAppointment,CheckAvailbility

================================

Model

using System.ComponentModel.DataAnnotations;

namespace DoctorApp.Models

{

public class Doctor

{

[Key]

public int DoctorId { get; set; }

public string DoctorName { get; set; }

public int YearOfExp { get; set; }

public string Speciality { get; set; }

public string Doctorphone { get; set; }

public string DoctorEmail { get; set; }

public string? pic { get; set; }

}

}

-=-------------------------------------------------------------------

using System.ComponentModel.DataAnnotations;

namespace DoctorApp.Models

{

public class Patient

{

[Key]

public int PatientId { get; set; }

public string PatientName { get; set; }

public int Age { get; set; }

public string PatientPhone { get; set; }

public string PatientEmail { get; set; }

}

}

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace DoctorApp.Models

{

public class Appointment

{

[Key]

public int AppointmentNumber { get; set; }

public DateTime? DateTime { get; set; }

public int DoctorId { get; set; }

[ForeignKey("DoctorId")]

public Doctor Doctor { get; set; }

public int PatientId { get; set; }

[ForeignKey("PatientId")]

public Patient Patient { get; set; }

public ICollection<Doctor> Doctors { get; set; }

public ICollection<Patient> Patients { get; set; }

}

}

Interfaces

namespace DoctorApp.Interfaces

{

public interface IRepository<K,T>where T : class

{

public T Add(T entity);

public T Delete(K id);

public T GetById(K id);

public T Update(T entity);

public ICollection<T> GetAll();

}

}

using DoctorApp.Models;

namespace DoctorApp.Interfaces

{

public interface IManagePatientService

{

Patient AddPatient(Patient patient);

}

}

using DoctorApp.Models;

namespace DoctorApp.Interfaces

{

public interface IManageDoctorService

{

Doctor AddDoctor(Doctor doctor);

public ICollection<Doctor> GetDoctorExp(int min, int max);

public Doctor UpdateDoctorExp(int id, Doctor doctor);

public Doctor UpdateDoctorPhone(int id, Doctor doctor);

public Doctor UpdateDoctorSpeciality(int id, Doctor doctor);

}

}

using DoctorApp.Models;

namespace DoctorApp.Interfaces

{

public interface IManageAppointmentService

{

Appointment BookAppointment(Appointment appointment);

public Appointment cancelAppointment(Appointment appointment);

bool CheckAvailability(Appointment appointment);

}

}

Repositories

using DoctorApp.Contexts;

using DoctorApp.Interfaces;

using DoctorApp.Models;

using Microsoft.EntityFrameworkCore;

namespace DoctorApp.Repositories

{

public class DoctorsRepository : IRepository<int, Doctor>

{

private readonly DoctorAppContexts \_context;

static List<Doctor> doctors = new List<Doctor>();

public DoctorsRepository(DoctorAppContexts contexts)

{

\_context = contexts;

}

public Doctor Add(Doctor entity)

{

if (entity == null)

throw new ArgumentNullException("No Doctor Available");

entity.DoctorId = GenerateIndex();

\_context.doctors.Add(entity);

\_context.SaveChanges();

return entity;

}

private int GenerateIndex()

{

int DoctorId = doctors.Count;

return ++DoctorId;

}

public Doctor Delete(int id)

{

var doctor = GetById(id);

if (doctor != null)

{

\_context.doctors.Remove(doctor);

\_context.SaveChanges();

return doctor;

}

return doctor;

}

public ICollection<Doctor> GetAll()

{

return \_context.doctors.ToList();

}

public Doctor Update(Doctor entity)

{

\_context.Entry<Doctor>(entity).State = EntityState.Modified;

\_context.SaveChanges();

return entity;

}

public Doctor GetById(int id)

{

Doctor doct = doctors.FirstOrDefault(p => p.DoctorId == id);

if (doct == null)

throw new InvalidOperationException("No doctor with id " + id);

return doct;

}

}

}

using DoctorApp.Contexts;

using DoctorApp.Interfaces;

using DoctorApp.Models;

using Microsoft.EntityFrameworkCore;

namespace DoctorApp.Repositories

{

public class PatientRepository : IRepository<int, Patient>

{

private readonly DoctorAppContexts \_context;

static List<Patient> patients = new List<Patient>();

public PatientRepository(DoctorAppContexts contexts)

{

\_context = contexts;

}

public Patient Add(Patient entity)

{

\_context.patients.Add(entity);

\_context.SaveChanges();

return entity;

}

public Patient Delete(int id)

{

var patient = Get(id);

if (patient != null)

{

\_context.patients.Remove(patient);

\_context.SaveChanges();

return patient;

}

return patient;

}

public Patient Get(int id)

{

var employee = \_context.patients.FirstOrDefault(e => e.PatientId == id);

return employee;

}

public ICollection<Patient> GetAll()

{

return \_context.patients.ToList();

}

public Patient Update(Patient entity)

{

\_context.Entry<Patient>(entity).State = EntityState.Modified;

\_context.SaveChanges();

return entity;

}

public Patient GetById(int id)

{

Patient doct = patients.FirstOrDefault(predicate: p => p.PatientId == id);

if (doct == null)

throw new InvalidOperationException("No Patient with id " + id);

return doct;

}

public Patient UpdatePatient(Patient patient )

{

Patient pat=GetById(patient.PatientId);

pat.PatientName = patient.PatientName;

pat.PatientPhone = patient.PatientPhone;

pat.PatientEmail = patient.PatientEmail;

return pat;

}

}

}

using DoctorApp.Interfaces;

using DoctorApp.Models;

using DoctorApp.Contexts;

using DoctorApp.Interfaces;

using DoctorApp.Models;

namespace DoctorApplication.Repositories

{

public class AppointmentRepository : IRepository<int, Appointment>

{

private readonly DoctorAppContexts \_context;

static List<Appointment> appointments = new List<Appointment>();

public AppointmentRepository(DoctorAppContexts contexts)

{

\_context = contexts;

}

public Appointment Add(Appointment entity)

{

if (entity == null)

throw new ArgumentNullException("No Doctor Available");

entity.AppointmentNumber = GenerateIndex();

\_context.appointments.Add(entity);

\_context.SaveChanges();

return entity;

}

private int GenerateIndex()

{

int AppointmentNumber = appointments.Count;

return ++AppointmentNumber;

}

public Appointment Delete(int id)

{

var apmt = GetById(id);

\_context.appointments.Remove(apmt);

\_context.SaveChanges();

return apmt;

}

public Appointment GetById(int id)

{

var apmt = \_context.appointments.FirstOrDefault(d => d.AppointmentNumber == id);

return apmt;

}

public ICollection<Appointment> GetAll()

{

return \_context.appointments.ToList();

}

public Appointment Update(Appointment entity)

{

\_context.Update(entity);

\_context.SaveChanges();

return entity;

}

}

}

Services

using DoctorApp.Exceptions;

using DoctorApp.Exceptions.PatientException;

using DoctorApp.Interfaces;

using DoctorApp.Models;

using DoctorApp.Repositories;

using Microsoft.EntityFrameworkCore;

namespace DoctorApp.Services

{

public class AppointmentService : IManageAppointmentService

{

private readonly IRepository<int, Appointment> \_repository;

private readonly IRepository<int, Doctor> \_DRepository;

private readonly IRepository<int, Patient> \_PRepository;

//static List<Doctor> doctors = new List<Doctor>();

public AppointmentService(IRepository<int, Appointment> repository , IRepository<int , Doctor>repository1 , IRepository<int , Patient>repository2)

{

\_repository = repository;

\_DRepository=repository1;

\_PRepository=repository2;

}

public Appointment BookAppointment(Appointment appointment)

{

bool book = CheckAvailability(appointment);

if (book== true)

{

\_repository.Add(appointment);

return appointment;

}

else

{

throw new DoctorNotAvailableException();

}

}

public Appointment cancelAppointment(Appointment appointment)

{

var dAppointment = \_repository.GetById(appointment.AppointmentNumber);

if(dAppointment == null)

{

throw new InvalidPatientException();

}

dAppointment = \_repository.Delete(dAppointment.AppointmentNumber);

return dAppointment;

}

public bool CheckAvailability(Appointment appointment)

{

if(appointment != null)

{

Doctor details = \_DRepository.GetById(appointment.DoctorId);

if(details != null)

{

var appointments = \_repository.GetAll();

var existingAppointment = appointments.Where(D => D.DoctorId == appointment.DoctorId

&& D.DateTime == appointment.DateTime).ToList();

if(existingAppointment == null)

{

return true;

}

else

{

return false;

}

}

else

{

throw new InvalidDoctorExceptions();

}

}

throw new InvalidDoctorExceptions();

}

}

}

using DoctorApp.Exceptions;

using DoctorApp.Interfaces;

using DoctorApp.Models;

using DoctorApp.Repositories;

namespace DoctorApp.Services

{

public class DoctorService : IManageDoctorService

{

private readonly IRepository<int, Doctor> \_repository;

private readonly DoctorsRepository \_dRepository;

private object p;

//static List<Doctor> doctors = new List<Doctor>();

public DoctorService(IRepository<int, Doctor> repository, DoctorsRepository repository1)

{

\_repository = repository;

\_dRepository= repository1;

}

#region AddDoctor

/// <summary>

/// this method to add Doctor

/// </summary>

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

/// <returns></returns>

public Doctor AddDoctor(Doctor doctor)

{

var result = \_repository.Add(doctor);

return result;

}

#endregion

#region GetDoctor Experience

public ICollection<Doctor> GetDoctorExp(int min, int max)

{

var doct = \_repository.GetAll();

if (min >= 00 || max > 0)

{

var expRange = doct.Where(p=>p.YearOfExp>= min && p.YearOfExp <= max).ToList();

if (expRange.Count == 0)

throw new InvalidDoctorExceptions();

return expRange;

}

throw new InvalidRangeOfYearOfExpException();

}

#endregion

#region Update Doctor Speciality

public Doctor UpdateDoctorSpeciality(int id, Doctor doctor)

{

var doct = \_dRepository.GetById(id);

if (doct == null)

throw new InvalidDoctorExceptions();

doct.Speciality = doctor.Speciality;

doct = \_repository.Update(doct);

return doct;

}

#endregion

#region Update Doctor Experience

public Doctor UpdateDoctorExp(int id ,Doctor doctor)

{

var doct = \_dRepository.GetById(id);

if (doct == null)

throw new InvalidDoctorExceptions();

doct.YearOfExp = doctor.YearOfExp;

doct = \_repository.Update(doct);

return doct;

}

#endregion

#region Update Doctor Phone

public Doctor UpdateDoctorPhone(int id, Doctor doctor)

{

var doct = \_dRepository.GetById(id);

if (doct == null)

throw new InvalidDoctorExceptions();

doct.Doctorphone = doctor.Doctorphone;

doct = \_repository.Update(doct);

return doct;

}

#endregion

#region Update Doctor

public Doctor UpdateDoctor(Doctor doctor)

{

Doctor doct = \_dRepository.GetById(doctor.DoctorId);

doct.DoctorName = doctor.DoctorName;

doct.YearOfExp = doctor.YearOfExp;

doct.Doctorphone = doctor.Doctorphone;

doct.DoctorEmail = doctor.DoctorEmail;

doct.Speciality = doctor.Speciality;

doct.pic = doctor.pic;

return doct;

}

#endregion

}

}

using DoctorApp.Interfaces;

using DoctorApp.Models;

using DoctorApp.Repositories;

using Microsoft.EntityFrameworkCore;

using System.Numerics;

namespace DoctorApp.Services

{

public class PatientService : IManagePatientService

{

private readonly IRepository<int, Patient> \_repository;

private object p;

private readonly PatientRepository \_pRepository;

public PatientService(IRepository<int, Patient> repository , PatientRepository repository1)

{

\_repository = repository;

\_pRepository = repository1;

}

public Patient AddPatient(Patient patient)

{

var result = \_repository.Add(patient);

return result;

}

#region Update Doctor

public Patient UpdatePatient(Patient patient)

{

Patient pat = \_pRepository.GetById(patient.PatientId);

pat.PatientName = patient.PatientName;

pat.PatientPhone = patient.PatientPhone;

pat.PatientEmail = patient.PatientEmail;

return pat;

}

#endregion

}

}

Contexts

using DoctorApp.Models;

using Microsoft.EntityFrameworkCore;

namespace DoctorApp.Contexts

{

public class DoctorAppContexts : DbContext

{

internal object appointment;

public DoctorAppContexts(DbContextOptions options) : base(options) // ctor chaining

{

}

#region CollectionToTable

public DbSet<Doctor> doctors { get; set; }// doctor

public DbSet<Patient> patients { get; set; } //patient

public DbSet<Appointment> appointments { get; set; } // appointments

#endregion

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Appointment>().HasKey(es => new { es.DoctorId, es.PatientId });

// Initializing data

modelBuilder.Entity<Doctor>().HasData(

new Doctor { DoctorId = 1, DoctorName = "Deva1", YearOfExp = 4, Speciality = "Surgen", Doctorphone = "1236541256", DoctorEmail = "abc@gmail.com", pic = "/Images/pic1.jpeg" }, // path is necessary

new Doctor { DoctorId = 2, DoctorName = "Deva2", YearOfExp = 4, Speciality = "General", Doctorphone = "8856904770", DoctorEmail = "xyz@gmail.com", pic = "/Images/pic2.jpeg" } // path is necessary

);

modelBuilder.Entity<Patient>().HasData(

new Patient { PatientId=101,PatientName="Ramu1",Age=23,PatientEmail="ramu1@gmail.com",PatientPhone="8525852252"},

new Patient { PatientId = 102, PatientName = "Ramu2", Age = 50, PatientEmail = "ramu2@gmail.com", PatientPhone = "00012102151" }

);

modelBuilder.Entity<Appointment>().HasData(

new Appointment { AppointmentNumber = 1, DoctorId = 1, PatientId = 101,DateTime=null},

new Appointment { AppointmentNumber = 2, DoctorId = 2, PatientId = 102, DateTime = null }

) ;

}

}

}

Exceptions

namespace DoctorApp.Exceptions

{

public class InvalidRangeOfYearOfExpException : Exception

{

string message;

public InvalidRangeOfYearOfExpException()

{

message = "Invalid value range Id";

}

public override string Message => message;

}

}

namespace DoctorApp.Exceptions

{

public class InvalidDoctorExceptions : Exception

{

string message;

public InvalidDoctorExceptions()

{

message = "Invalid Doctor Id";

}

public override string Message => message;

}

}

namespace DoctorApp.Exceptions

{

public class DoctorNotAvailableException : Exception

{

string message;

public DoctorNotAvailableException()

{

message = "Doctors Not Available";

}

public override string Message => message;

}

}

namespace DoctorApp.Exceptions.PatientException

{

public class InvalidPatientException : Exception

{

string message;

public InvalidPatientException()

{

message = "Invalid Patient";

}

public override string Message => message;

}

}

Controller

using DoctorApp.Interfaces;

using DoctorApp.Models;

using DoctorApp.Repositories;

using DoctorApp.Services;

using DoctorApplication.Repositories;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using System.Diagnostics;

namespace DoctorApp.Controllers

{

public class AppointmentController : Controller

{

private readonly IRepository<int, Appointment> \_repository;

private readonly AppointmentService \_appointmentService;

static List<Patient> patient = new List<Patient>();

public AppointmentController(IRepository<int, Appointment> repository, AppointmentRepository repository1, AppointmentService appointmentService)

{

\_repository = repository;

\_appointmentService = appointmentService;

}

#region Index = /Appointment/Index

public IActionResult Index()

{

return View(\_repository.GetAll());

}

#endregion

#region book = /Appointment/BookAppointment

[HttpGet]

public IActionResult BookAppointment()

{

//ViewBag.ampylist = GetAppointment();

return View();

}

[HttpPost]

public IActionResult BookAppointment(Appointment appointment)

{

//ViewBag.ampylist = GetAppointment();

\_repository.Add(appointment);

return RedirectToAction("Index");

}

//private List<SelectListItem> GetAppointment()

//{

// List<SelectListItem> ampylist = new List<SelectListItem>();

// var ampt = \_ARepo.GetAll();

// foreach (var d in ampt)

// {

// ampylist.Add(

// new SelectListItem

// { Value = d.AppointmentNumber.ToString() }

// );

// }

// return ampylist;

//}

#endregion

#region CheckAvailability = /Appointment/CheckAvailability

[HttpGet]

public IActionResult CheckAvailability(Appointment appointment)

{

//ViewBag.ampylist = GetAppointment();

return View();

}

#endregion

#region cancel Appointment = /Appointment/cancelAppointment

[HttpGet]

public IActionResult cancelAppointment(int id)

{

var appointments = \_repository.GetById(id);

return View(appointments);

}

[HttpPost]

public IActionResult cancelAppointment(int id, Appointment appointment)

{

ViewBag.Message = "";

try

{

\_repository.Delete(id);

return RedirectToAction("Index");

}

catch (Exception e)

{

Debug.WriteLine(e.Message);

ViewBag.Message = "Unable to delete appointment";

}

return View(appointment);

}

#endregion

}

}

using DoctorApp.Interfaces;

using DoctorApp.Models;

using DoctorApp.Repositories;

using DoctorApp.Services;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using System.Diagnostics;

namespace DoctorApp.Controllers

{

public class DoctorController : Controller

{

private readonly IRepository<int, Doctor> \_repository;

private readonly DoctorService \_doctorService;

static List<Doctor> doctors = new List<Doctor>();

public DoctorController(IRepository<int, Doctor> repository, DoctorsRepository repository1,

DoctorService doctorService)

{

\_repository = repository;

\_doctorService = doctorService;

}

#region index == /Index

public IActionResult Index()

{

return View(\_repository.GetAll());

}

#endregion

#region Add == /Doctor/AddDoctor

[HttpGet]

public IActionResult AddDoctor()

{

//ViewBag.doctorList = GetDoctor();

return View();

}

[HttpPost]

public IActionResult AddDoctor(Doctor doctor)

{

//ViewBag.doctorList = GetDoctor();

\_doctorService.AddDoctor(doctor);

return RedirectToAction("Index");

}

//private List<SelectListItem> GetDoctor()

//{

// List<SelectListItem> doctorList = new List<SelectListItem>();

// var doct = \_docRe.GetAll();

// foreach (var d in doct)

// {

// doctorList.Add(

// new SelectListItem

// { Text = d.DoctorName, Value = d.DoctorId.ToString() }

// );

// }

// return doctorList;

//}

#endregion

#region update Experience = /Doctor/UpdateDoctorExp

[HttpGet]

public IActionResult UpdateDoctorExp(int id)

{

var doct = \_repository.GetById(id);

return View(doct);

}

[HttpPost]

public IActionResult UpdateDoctorExp(int id, Doctor doctor)

{

ViewBag.Message = "";

try

{

\_doctorService.UpdateDoctorExp(id , doctor);

\_repository.Update(doctor);

return RedirectToAction("Index");

}

catch (Exception e)

{

Debug.WriteLine(e.Message);

ViewBag.Message = "doctor not found";

}

return View(doctor);

}

#endregion

#region update Speciality == /Doctor/UpdateDoctorSpeciality

[HttpGet]

public IActionResult UpdateDoctorSpeciality(int id)

{

var doctorexp = \_repository.GetById(id);

return View(doctorexp);

}

[HttpPost]

public IActionResult UpdateDoctorSpeciality(int id, Doctor doctor)

{

ViewBag.Message = "";

try

{

\_doctorService.UpdateDoctorSpeciality(id , doctor);

\_repository.Update(doctor);

return RedirectToAction("Index");

}

catch (Exception e)

{

Debug.WriteLine(e.Message);

ViewBag.Message = "doctor not found";

}

return View(doctor);

}

#endregion

#region delete = /Doctor/Delete

[HttpGet]

public IActionResult Delete(int id)

{

var doctor = \_repository.GetById(id);

return View(doctor);

}

[HttpPost]

public IActionResult Delete(int id, Doctor doctor)

{

ViewBag.Message = "";

try

{

\_repository.Delete(id);

return RedirectToAction("Index");

}

catch (Exception e)

{

Debug.WriteLine(e.Message);

ViewBag.Message = "Unable to delete doctor";

}

return View(doctor);

}

#endregion

#region update Phone = /Doctor/UpdateDoctorPhone

[HttpGet]

public IActionResult UpdateDoctorPhone(int id)

{

var doctorexp = \_repository.GetById(id);

return View(doctorexp);

}

[HttpPost]

public IActionResult UpdateDoctorPhone(int id, Doctor doctor)

{

ViewBag.Message = "";

try

{

\_doctorService.UpdateDoctorPhone(id, doctor);

\_repository.Update(doctor);

return RedirectToAction("Index");

}

catch (Exception e)

{

Debug.WriteLine(e.Message);

ViewBag.Message = "doctor not found";

}

return View(doctor);

}

#endregion

}

}

using DoctorApp.Interfaces;

using DoctorApp.Models;

using DoctorApp.Repositories;

using DoctorApp.Services;

using Microsoft.AspNetCore.Mvc;

using DoctorApp.Services;

using System.Diagnostics;

using Microsoft.AspNetCore.Mvc.Rendering;

namespace DoctorApp.Controllers

{

public class PatientController : Controller

{

private readonly IRepository<int, Patient> \_repository;

private readonly PatientService \_patientService;

static List<Patient> patient = new List<Patient>();

public PatientController(IRepository<int, Patient> repository, PatientRepository repository1, PatientService patientService1)

{

\_repository = repository;

\_patientService = patientService1;

}

#region Index = Patient/Index

public IActionResult Index()

{

return View(\_repository.GetAll());

}

#endregion

#region Add Patient == /Patient/Add

[HttpGet]

public IActionResult AddPatient()

{

//ViewBag.patientList = GetPatient();

return View();

}

[HttpPost]

public IActionResult AddPatient(Patient patient)

{

//ViewBag.patientList = GetPatient();

\_repository.Add(patient);

return RedirectToAction("Index");

}

//private List<SelectListItem> GetPatient()

//{

// List<SelectListItem> patientList = new List<SelectListItem>();

// var doct = \_Prepo.GetAll();

// foreach (var p in doct)

// {

// patientList.Add(

// new SelectListItem

// { Text = p.PatientName, Value = p.PatientId.ToString() }

// );

// }

// return patientList;

//}

#endregion

#region update Patient

[HttpGet]

public IActionResult UpdatePatient(int id)

{

var pat = \_repository.GetById(id);

return View(pat);

}

[HttpPost]

public IActionResult UpdatePatient( int id ,Patient patient)

{

ViewBag.Message = "";

try

{

\_patientService.UpdatePatient(patient);

return RedirectToAction("Index");

}

catch (Exception e)

{

Debug.WriteLine(e.Message);

ViewBag.Message = "patient not found";

}

return View(patient);

}

#endregion

}

}

Doctor view

@model DoctorApp.Models.Doctor

@{

ViewData["Title"] = "AddDoctor";

}

<h1>AddDoctor</h1>

<h4>Doctor</h4>

<hr />

<div class="row">

<div class="col-md-4">

<**form** **asp-action**="AddDoctor">

<**div** **asp-validation-summary**="ModelOnly" class="text-danger"></**div**>

@\* <div class="form-group">

<label asp-for="DoctorId" class="control-label"></label>

<input asp-for="DoctorId" class="form-control" />

<span asp-validation-for="DoctorId" class="text-danger"></span>

</div> \*@

<div class="form-group">

<**label** **asp-for**="DoctorName" class="control-label"></**label**>

<**input** **asp-for**="DoctorName" class="form-control" />

<**span** **asp-validation-for**="DoctorName" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="YearOfExp" class="control-label"></**label**>

<**input** **asp-for**="YearOfExp" class="form-control" />

<**span** **asp-validation-for**="YearOfExp" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Speciality" class="control-label"></**label**>

<**input** **asp-for**="Speciality" class="form-control" />

<**span** **asp-validation-for**="Speciality" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Doctorphone" class="control-label"></**label**>

<**input** **asp-for**="Doctorphone" class="form-control" />

<**span** **asp-validation-for**="Doctorphone" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="DoctorEmail" class="control-label"></**label**>

<**input** **asp-for**="DoctorEmail" class="form-control" />

<**span** **asp-validation-for**="DoctorEmail" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="pic" class="control-label"></**label**>

<**input** **asp-for**="pic" class="form-control" />

<**span** **asp-validation-for**="pic" class="text-danger"></**span**>

</div>

<div class="form-group">

<input type="submit" value="Create" class="btn btn-primary" />

</div>

</**form**>

</div>

</div>

<div>

<**a** **asp-action**="Index">Back to List</**a**>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

@model DoctorApp.Models.Doctor

@{

ViewData["Title"] = "Delete";

}

<h1>Delete</h1>

<h3>Are you sure you want to delete this?</h3>

<div>

<h4>Doctor</h4>

<hr />

<dl class="row">

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.DoctorId)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.DoctorId)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.DoctorName)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.DoctorName)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.YearOfExp)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.YearOfExp)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Speciality)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Speciality)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.Doctorphone)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.Doctorphone)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.DoctorEmail)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.DoctorEmail)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.pic)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.pic)

</dd>

</dl>

<**form** **asp-action**="Delete">

<input type="submit" value="Delete" class="btn btn-danger" /> |

<**a** **asp-action**="Index">Back to List</**a**>

</**form**>

</div>

@model IEnumerable<DoctorApp.Models.Doctor>

@{

ViewData["Title"] = "Index";

}

<h1>Index</h1>

<p>

<**a** **asp-action**="Create">Create New</**a**>

</p>

<table class="table">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.DoctorId)

</th>

<th>

@Html.DisplayNameFor(model => model.DoctorName)

</th>

<th>

@Html.DisplayNameFor(model => model.YearOfExp)

</th>

<th>

@Html.DisplayNameFor(model => model.Speciality)

</th>

<th>

@Html.DisplayNameFor(model => model.Doctorphone)

</th>

<th>

@Html.DisplayNameFor(model => model.DoctorEmail)

</th>

<th>

@Html.DisplayNameFor(model => model.pic)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model) {

<tr>

<td>

@Html.DisplayFor(modelItem => item.DoctorId)

</td>

<td>

@Html.DisplayFor(modelItem => item.DoctorName)

</td>

<td>

@Html.DisplayFor(modelItem => item.YearOfExp)

</td>

<td>

@Html.DisplayFor(modelItem => item.Speciality)

</td>

<td>

@Html.DisplayFor(modelItem => item.Doctorphone)

</td>

<td>

@Html.DisplayFor(modelItem => item.DoctorEmail)

</td>

<td>

@Html.DisplayFor(modelItem => item.pic)

</td>

<td>

@Html.ActionLink("Edit", "Edit", new { /\* id=item.PrimaryKey \*/ }) |

@Html.ActionLink("Details", "Details", new { /\* id=item.PrimaryKey \*/ }) |

@Html.ActionLink("Delete", "Delete", new { /\* id=item.PrimaryKey \*/ })

</td>

</tr>

}

</tbody>

</table>

@model DoctorApp.Models.Doctor

@{

ViewData["Title"] = "UpdateDoctorExp";

}

<h1>UpdateDoctorExp</h1>

<h4>Doctor</h4>

<hr />

<div class="row">

<div class="col-md-4">

<**form** **asp-action**="UpdateDoctorExp">

<**div** **asp-validation-summary**="ModelOnly" class="text-danger"></**div**>

@\* <div class="form-group">

<label asp-for="DoctorId" class="control-label"></label>

<input asp-for="DoctorId" class="form-control" />

<span asp-validation-for="DoctorId" class="text-danger"></span>

</div> \*@

@\* <div class="form-group">

<label asp-for="DoctorName" class="control-label"></label>

<input asp-for="DoctorName" class="form-control" />

<span asp-validation-for="DoctorName" class="text-danger"></span>

</div> \*@

<div class="form-group">

<**label** **asp-for**="YearOfExp" class="control-label"></**label**>

<**input** **asp-for**="YearOfExp" class="form-control" />

<**span** **asp-validation-for**="YearOfExp" class="text-danger"></**span**>

</div>

@\* <div class="form-group">

<label asp-for="Speciality" class="control-label"></label>

<input asp-for="Speciality" class="form-control" />

<span asp-validation-for="Speciality" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Doctorphone" class="control-label"></label>

<input asp-for="Doctorphone" class="form-control" />

<span asp-validation-for="Doctorphone" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="DoctorEmail" class="control-label"></label>

<input asp-for="DoctorEmail" class="form-control" />

<span asp-validation-for="DoctorEmail" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="pic" class="control-label"></label>

<input asp-for="pic" class="form-control" />

<span asp-validation-for="pic" class="text-danger"></span>

</div> \*@

<div class="form-group">

<input type="submit" value="Save" class="btn btn-primary" />

</div>

</**form**>

</div>

</div>

<div>

<**a** **asp-action**="Index">Back to List</**a**>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

@model DoctorApp.Models.Doctor

@{

ViewData["Title"] = "UpdateDoctorPhone";

}

<h1>UpdateDoctorPhone</h1>

<h4>Doctor</h4>

<hr />

<div class="row">

<div class="col-md-4">

<**form** **asp-action**="UpdateDoctorPhone">

<**div** **asp-validation-summary**="ModelOnly" class="text-danger"></**div**>

@\* <div class="form-group">

<label asp-for="DoctorId" class="control-label"></label>

<input asp-for="DoctorId" class="form-control" />

<span asp-validation-for="DoctorId" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="DoctorName" class="control-label"></label>

<input asp-for="DoctorName" class="form-control" />

<span asp-validation-for="DoctorName" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="YearOfExp" class="control-label"></label>

<input asp-for="YearOfExp" class="form-control" />

<span asp-validation-for="YearOfExp" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Speciality" class="control-label"></label>

<input asp-for="Speciality" class="form-control" />

<span asp-validation-for="Speciality" class="text-danger"></span>

</div> \*@

<div class="form-group">

<**label** **asp-for**="Doctorphone" class="control-label"></**label**>

<**input** **asp-for**="Doctorphone" class="form-control" />

<**span** **asp-validation-for**="Doctorphone" class="text-danger"></**span**>

</div>

@\* <div class="form-group">

<label asp-for="DoctorEmail" class="control-label"></label>

<input asp-for="DoctorEmail" class="form-control" />

<span asp-validation-for="DoctorEmail" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="pic" class="control-label"></label>

<input asp-for="pic" class="form-control" />

<span asp-validation-for="pic" class="text-danger"></span>

</div> \*@

<div class="form-group">

<input type="submit" value="Save" class="btn btn-primary" />

</div>

</**form**>

</div>

</div>

<div>

<**a** **asp-action**="Index">Back to List</**a**>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

@model DoctorApp.Models.Doctor

@{

ViewData["Title"] = "UpdateDoctorSpeciality";

}

<h1>UpdateDoctorSpeciality</h1>

<h4>Doctor</h4>

<hr />

<div class="row">

<div class="col-md-4">

<**form** **asp-action**="UpdateDoctorSpeciality">

<**div** **asp-validation-summary**="ModelOnly" class="text-danger"></**div**>

@\* <div class="form-group">

<label asp-for="DoctorId" class="control-label"></label>

<input asp-for="DoctorId" class="form-control" />

<span asp-validation-for="DoctorId" class="text-danger"></span>

</div> \*@

@\* <div class="form-group">

<label asp-for="DoctorName" class="control-label"></label>

<input asp-for="DoctorName" class="form-control" />

<span asp-validation-for="DoctorName" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="YearOfExp" class="control-label"></label>

<input asp-for="YearOfExp" class="form-control" />

<span asp-validation-for="YearOfExp" class="text-danger"></span>

</div> \*@

<div class="form-group">

<**label** **asp-for**="Speciality" class="control-label"></**label**>

<**input** **asp-for**="Speciality" class="form-control" />

<**span** **asp-validation-for**="Speciality" class="text-danger"></**span**>

</div>

@\* <div class="form-group">

<label asp-for="Doctorphone" class="control-label"></label>

<input asp-for="Doctorphone" class="form-control" />

<span asp-validation-for="Doctorphone" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="DoctorEmail" class="control-label"></label>

<input asp-for="DoctorEmail" class="form-control" />

<span asp-validation-for="DoctorEmail" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="pic" class="control-label"></label>

<input asp-for="pic" class="form-control" />

<span asp-validation-for="pic" class="text-danger"></span>

</div> \*@

<div class="form-group">

<input type="submit" value="Save" class="btn btn-primary" />

</div>

</**form**>

</div>

</div>

<div>

<**a** **asp-action**="Index">Back to List</**a**>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

Patient views

@model DoctorApp.Models.Patient

@{

ViewData["Title"] = "AddPatient";

}

<h1>AddPatient</h1>

<h4>Patient</h4>

<hr />

<div class="row">

<div class="col-md-4">

<**form** **asp-action**="AddPatient">

<**div** **asp-validation-summary**="ModelOnly" class="text-danger"></**div**>

@\* <div class="form-group">

<label asp-for="PatientId" class="control-label"></label>

<input asp-for="PatientId" class="form-control" />

<span asp-validation-for="PatientId" class="text-danger"></span>

</div> \*@

<div class="form-group">

<**label** **asp-for**="PatientName" class="control-label"></**label**>

<**input** **asp-for**="PatientName" class="form-control" />

<**span** **asp-validation-for**="PatientName" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Age" class="control-label"></**label**>

<**input** **asp-for**="Age" class="form-control" />

<**span** **asp-validation-for**="Age" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="PatientPhone" class="control-label"></**label**>

<**input** **asp-for**="PatientPhone" class="form-control" />

<**span** **asp-validation-for**="PatientPhone" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="PatientEmail" class="control-label"></**label**>

<**input** **asp-for**="PatientEmail" class="form-control" />

<**span** **asp-validation-for**="PatientEmail" class="text-danger"></**span**>

</div>

<div class="form-group">

<input type="submit" value="Create" class="btn btn-primary" />

</div>

</**form**>

</div>

</div>

<div>

<**a** **asp-action**="Index">Back to List</**a**>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

@model IEnumerable<DoctorApp.Models.Patient>

@{

ViewData["Title"] = "Index";

}

<h1>Index</h1>

<p>

<**a** **asp-action**="Create">Create New</**a**>

</p>

<table class="table">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.PatientId)

</th>

<th>

@Html.DisplayNameFor(model => model.PatientName)

</th>

<th>

@Html.DisplayNameFor(model => model.Age)

</th>

<th>

@Html.DisplayNameFor(model => model.PatientPhone)

</th>

<th>

@Html.DisplayNameFor(model => model.PatientEmail)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model) {

<tr>

<td>

@Html.DisplayFor(modelItem => item.PatientId)

</td>

<td>

@Html.DisplayFor(modelItem => item.PatientName)

</td>

<td>

@Html.DisplayFor(modelItem => item.Age)

</td>

<td>

@Html.DisplayFor(modelItem => item.PatientPhone)

</td>

<td>

@Html.DisplayFor(modelItem => item.PatientEmail)

</td>

<td>

@Html.ActionLink("Edit", "Edit", new { /\* id=item.PrimaryKey \*/ }) |

@Html.ActionLink("Details", "Details", new { /\* id=item.PrimaryKey \*/ }) |

@Html.ActionLink("Delete", "Delete", new { /\* id=item.PrimaryKey \*/ })

</td>

</tr>

}

</tbody>

</table>

@model DoctorApp.Models.Patient

@{

ViewData["Title"] = "UpdatePatient";

}

<h1>UpdatePatient</h1>

<h4>Patient</h4>

<hr />

<div class="row">

<div class="col-md-4">

<**form** **asp-action**="UpdatePatient">

<**div** **asp-validation-summary**="ModelOnly" class="text-danger"></**div**>

@\* <div class="form-group">

<label asp-for="PatientId" class="control-label"></label>

<input asp-for="PatientId" class="form-control" />

<span asp-validation-for="PatientId" class="text-danger"></span>

</div> \*@

<div class="form-group">

<**label** **asp-for**="PatientName" class="control-label"></**label**>

<**input** **asp-for**="PatientName" class="form-control" />

<**span** **asp-validation-for**="PatientName" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="Age" class="control-label"></**label**>

<**input** **asp-for**="Age" class="form-control" />

<**span** **asp-validation-for**="Age" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="PatientPhone" class="control-label"></**label**>

<**input** **asp-for**="PatientPhone" class="form-control" />

<**span** **asp-validation-for**="PatientPhone" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="PatientEmail" class="control-label"></**label**>

<**input** **asp-for**="PatientEmail" class="form-control" />

<**span** **asp-validation-for**="PatientEmail" class="text-danger"></**span**>

</div>

<div class="form-group">

<input type="submit" value="Save" class="btn btn-primary" />

</div>

</**form**>

</div>

</div>

<div>

<**a** **asp-action**="Index">Back to List</**a**>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

Appointment views

@model DoctorApp.Models.Appointment

@{

ViewData["Title"] = "BookAppointment";

}

<h1>BookAppointment</h1>

<h4>Appointment</h4>

<hr />

<div class="row">

<div class="col-md-4">

<**form** **asp-action**="BookAppointment">

<**div** **asp-validation-summary**="ModelOnly" class="text-danger"></**div**>

@\* <div class="form-group">

<label asp-for="AppointmentNumber" class="control-label"></label>

<input asp-for="AppointmentNumber" class="form-control" />

<span asp-validation-for="AppointmentNumber" class="text-danger"></span>

</div> \*@

<div class="form-group">

<**label** **asp-for**="DateTime" class="control-label"></**label**>

<**input** **asp-for**="DateTime" class="form-control" />

<**span** **asp-validation-for**="DateTime" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="DoctorId" class="control-label"></**label**>

<**input** **asp-for**="DoctorId" class="form-control" />

<**span** **asp-validation-for**="DoctorId" class="text-danger"></**span**>

</div>

<div class="form-group">

<**label** **asp-for**="PatientId" class="control-label"></**label**>

<**input** **asp-for**="PatientId" class="form-control" />

<**span** **asp-validation-for**="PatientId" class="text-danger"></**span**>

</div>

<div class="form-group">

<input type="submit" value="Create" class="btn btn-primary" />

</div>

</**form**>

</div>

</div>

<div>

<**a** **asp-action**="Index">Back to List</**a**>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

@model DoctorApp.Models.Appointment

@{

ViewData["Title"] = "cancelAppointment";

}

<h1>cancelAppointment</h1>

<h3>Are you sure you want to delete this?</h3>

<div>

<h4>Appointment</h4>

<hr />

<dl class="row">

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.AppointmentNumber)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.AppointmentNumber)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.DateTime)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.DateTime)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.DoctorId)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.DoctorId)

</dd>

<dt class = "col-sm-2">

@Html.DisplayNameFor(model => model.PatientId)

</dt>

<dd class = "col-sm-10">

@Html.DisplayFor(model => model.PatientId)

</dd>

</dl>

<**form** **asp-action**="cancelAppointment">

<input type="submit" value="Delete" class="btn btn-danger" /> |

<**a** **asp-action**="Index">Back to List</**a**>

</**form**>

</div>

@model IEnumerable<DoctorApp.Models.Appointment>

@{

ViewData["Title"] = "CheckAvailability";

}

<h1>CheckAvailability</h1>

<p>

<**a** **asp-action**="Create">Create New</**a**>

</p>

<table class="table">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.AppointmentNumber)

</th>

<th>

@Html.DisplayNameFor(model => model.DateTime)

</th>

<th>

@Html.DisplayNameFor(model => model.DoctorId)

</th>

<th>

@Html.DisplayNameFor(model => model.PatientId)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model) {

<tr>

<td>

@Html.DisplayFor(modelItem => item.AppointmentNumber)

</td>

<td>

@Html.DisplayFor(modelItem => item.DateTime)

</td>

<td>

@Html.DisplayFor(modelItem => item.DoctorId)

</td>

<td>

@Html.DisplayFor(modelItem => item.PatientId)

</td>

<td>

@Html.ActionLink("Edit", "Edit", new { /\* id=item.PrimaryKey \*/ }) |

@Html.ActionLink("Details", "Details", new { /\* id=item.PrimaryKey \*/ }) |

@Html.ActionLink("Delete", "Delete", new { /\* id=item.PrimaryKey \*/ })

</td>

</tr>

}

</tbody>

</table>