Create html page of shopping site

<!DOCTYPE html>

<html lang="en">

<head>

<title>Shopping Site</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.4/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

<style>

.navbar {

margin-bottom: 50px;

border-radius: 0;

}

.jumbotron {

margin-bottom: 0;

}

footer {

background-color: #f2f2f2;

padding: 25px;

}

</style>

</head>

<body>

<div>

<div class="container text-center">

<h1>Apni Dukan</h1>

</div>

</div>

<nav class="navbar navbar-inverse">

<div class="container-fluid">

<div class="navbar-header">

<button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#myNavbar">

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-brand" href="#">Logo</a>

</div>

<div class="collapse navbar-collapse" id="myNavbar">

<ul class="nav navbar-nav">

<li class="active"><a href="#">Home</a></li>

<li><a href="#">Products</a></li>

<li><a href="#">Deals</a></li>

<li><a href="#">Stores</a></li>

<li><a href="#">Contact</a></li>

</ul>

<ul class="nav navbar-nav navbar-right">

<li><a href="#"><span class="glyphicon glyphicon-user"></span> Your Account</a></li>

<li><a href="#"><span class="glyphicon glyphicon-shopping-cart"></span> Cart</a></li>

</ul>

</div>

</div>

</nav>

<div class="container">

<div class="row">

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

<div class="panel panel-primary">

<div class="panel-heading">BLACK FRIDAY DEAL</div>

<div class="panel-body"><img src="https://placehold.it/150x80?text=IMAGE" class="img-responsive" style="width:100%" alt="Image"></div>

<div class="panel-footer">Buy 50 mobiles and get a gift card</div>

</div>

</div>

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

<div class="panel panel-danger">

<div class="panel-heading">BLACK FRIDAY DEAL</div>

<div class="panel-body"><img src="https://placehold.it/150x80?text=IMAGE" class="img-responsive" style="width:100%" alt="Image"></div>

<div class="panel-footer">Buy 50 mobiles and get a gift card</div>

</div>

</div>

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

<div class="panel panel-success">

<div class="panel-heading">BLACK FRIDAY DEAL</div>

<div class="panel-body"><img src="https://placehold.it/150x80?text=IMAGE" class="img-responsive" style="width:100%" alt="Image"></div>

<div class="panel-footer">Buy 50 mobiles and get a gift card</div>

</div>

</div>

</div>

</div><br>

<div class="container">

<div class="row">

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

<div class="panel panel-primary">

<div class="panel-heading">BLACK FRIDAY DEAL</div>

<div class="panel-body"><img src="https://placehold.it/150x80?text=IMAGE" class="img-responsive" style="width:100%" alt="Image"></div>

<div class="panel-footer">Buy 50 mobiles and get a gift card</div>

</div>

</div>

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

<div class="panel panel-primary">

<div class="panel-heading">BLACK FRIDAY DEAL</div>

<div class="panel-body"><img src="https://placehold.it/150x80?text=IMAGE" class="img-responsive" style="width:100%" alt="Image"></div>

<div class="panel-footer">Buy 50 mobiles and get a gift card</div>

</div>

</div>

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

<div class="panel panel-primary">

<div class="panel-heading">BLACK FRIDAY DEAL</div>

<div class="panel-body"><img src="https://placehold.it/150x80?text=IMAGE" class="img-responsive" style="width:100%" alt="Image"></div>

<div class="panel-footer">Buy 50 mobiles and get a gift card</div>

</div>

</div>

</div>

</div><br><br>

<footer class="container-fluid text-center">

<p>Online Store Copyright</p>

<form class="form-inline">

Get deals:

<input type="email" class="form-control" size="50" placeholder="Email Address">

<button type="button" class="btn btn-danger">Sign Up</button>

</form>

</footer>

</body>

</html>

UnitTests

AppointmentServiceTests

using HospitalClinicApp.contexts;

using HospitalClinicApp.Interfaces;

using HospitalClinicApp.Models;

using HospitalClinicApp.Repositories;

using HospitalClinicApp.Services;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace HospitalTest

{

internal class AppointmentServiceTest

{

ClinicContexts context;

[SetUp]

public void Setup()

{

var dbContextOption = new DbContextOptionsBuilder<ClinicContexts>().UseInMemoryDatabase(databaseName: "dbDummyClinic").Options;

context = new ClinicContexts(dbContextOption);

}

#region Add appointment

[Test]

public void checkAvalibilityTest()

{

// Arrange

IRepository<int, Appointment> appointmentRepository = new AppointmentRepository(context);

DateTime dt2 = new DateTime(2023, 06, 7, 5, 10, 20);

//Action

Appointment appointment = (new Appointment { AppointmentNumber = 1, PatientId = 1, DoctorId = 1, AppointmentDateTime = dt2 });

appointmentRepository.Add(appointment);

IAppointmentService appointmentservice = new AppointmentService(appointmentRepository);

var result = new Appointment { AppointmentNumber = 1, PatientId = 1, DoctorId = 1, AppointmentDateTime = dt2 };

//Assert

Assert.NotNull(appointment.AppointmentDateTime);

}

#endregion

#region Add

[Test]

public void AddTest()

{

// Arrange

IRepository<int, Appointment> appointmentRepository = new AppointmentRepository(context);

DateTime dt2 = new DateTime(2023, 06, 7, 5, 10, 20);

//Action

Appointment appointment = (new Appointment { AppointmentNumber = 1, PatientId = 1, DoctorId = 1, AppointmentDateTime = dt2 });

appointmentRepository.Add(appointment);

IAppointmentService appointmentservice = new AppointmentService(appointmentRepository);

var result = new Appointment { AppointmentNumber = 1, PatientId = 1, DoctorId = 1, AppointmentDateTime = dt2 };

//Assert

Assert.AreEqual(result,appointment);

}

#endregion

#region cancel appointment

[Test]

public void cancelAppointmentTest()

{

// Arrange

IRepository<int, Appointment> appointmentRepository = new AppointmentRepository(context);

DateTime dt2 = new DateTime(2023, 06, 7, 5, 10, 20);

//Action

Appointment appointment = (new Appointment { AppointmentNumber = 1, PatientId = 1, DoctorId = 1, AppointmentDateTime = dt2 });

appointmentRepository.Add(appointment);

IAppointmentService appointmentservice = new AppointmentService(appointmentRepository);

Appointment result = appointmentRepository.Delete(appointment.AppointmentNumber=1);

//Assert

Assert.IsNotNull(result);

}

#endregion

}

}

PatientServiceTests

using HospitalClinicApp.contexts;

using HospitalClinicApp.Interfaces;

using HospitalClinicApp.Models;

using HospitalClinicApp.Repositories;

using HospitalClinicApp.Services;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace HospitalTest

{

public class PatientServiceTest

{

ClinicContexts context;

[SetUp]

public void Setup()

{

var dbContextOption = new DbContextOptionsBuilder<ClinicContexts>().UseInMemoryDatabase(databaseName: "dbDummyClinic").Options;

context = new ClinicContexts(dbContextOption);

}

#region Add Patient

[Test]

public void AddTest()

{

// Arrange

IRepository<int, Patient> patientRepository = new PatientRepository(context);

IPatientService patientService = new PatientService(patientRepository);

//Action

var pat = new Patient {Id = 1 , Name = "Deva", Email = "Deva@gmail.com" , Age=22 , Phone="8856904770" };

var result = patientService.Add(pat);

var data = new Patient { Id = 1, Name = "Deva", Email = "Deva@gmail.com", Age = 22, Phone = "8856904770" };

//Assert

Assert.AreEqual(data.Id, result.Id);

}

#endregion

#region GeatAllPatient

[Test]

public void GetAllPatientTest()

{

// Arrange

IRepository<int, Patient> patientRepository = new PatientRepository(context);

IPatientService patientService = new PatientService(patientRepository);

Patient? patient = new Patient { Id = 1, Name = "Deva", Email = "Deva@gmail.com", Age = 22, Phone = "8856904770" };

patientRepository.Add(patient);

//Action

var result = patientService.GetAllPatient();

// Assert

Assert.NotNull(result);

Assert.AreEqual(1, result.Count);

}

#endregion

#region

[Test]

public void UpdatePatientTest()

{

// Arrange

IRepository<int, Patient> patientRepository = new PatientRepository(context);

IPatientService patientService = new PatientService(patientRepository);

//Action

var pat = new Patient { Id = 1, Name = "Deva", Email = "Deva@gmail.com", Age = 22, Phone = "8856904770" };

patientService.Add(pat);

Patient? result = patientService.UpdatePatient(new Patient { Id = 1, Name = "lakshman", Email = "xyz@gmail.com", Age = 2, Phone = "7972051076" });

var data = new Patient { Id = 1, Name = "lakshman", Email = "xyz@gmail.com", Age = 2, Phone = "7972051076" };

//Assert

Assert.AreEqual(data,result);

}

#endregion

}

}

DoctorServiceTests

using HospitalClinicApp.contexts;

using HospitalClinicApp.Interfaces;

using HospitalClinicApp.Models.DTO;

using HospitalClinicApp.Models;

using HospitalClinicApp.Repositories;

using HospitalClinicApp.Services;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace HospitalTest

{

public class DoctorServiceTest

{

ClinicContexts context;

//Gets executed for every test

#region LoginTest

[SetUp]

public void Setup()

{

var dbContextOption = new DbContextOptionsBuilder<ClinicContexts>().UseInMemoryDatabase(databaseName: "dbDummyClinic").Options;

context = new ClinicContexts(dbContextOption);

}

[Test]

public void AddTest()

{

// Arrange

IRepository<int, Doctor> doctorRepository = new DoctorRepository(context);

IDoctorService doctorService = new DoctorService(doctorRepository);

//Action

var doct = new Doctor { Id = 1, Name = "Deva", Email = "Deva@gmail.com", Experience = 2, Phone = "8856904770", Speciality = "surgen", Pic = "-" };

var result = doctorService.Add(doct);

var data = new Doctor { Id = 1, Name = "Deva", Email = "Deva@gmail.com", Experience = 2, Phone = "8856904770", Speciality = "surgen", Pic = "-" };

//Assert

Assert.AreEqual(data.Id, result.Id);

}

#endregion

#region UpdateSpecialization

[Test]

public void UpdateSpecializationTest()

{

//Arrange

IRepository<int, Doctor> doctorRepository = new DoctorRepository(context);

IDoctorService doctorService = new DoctorService(doctorRepository);

Doctor? doct = new Doctor { Id = 1, Name = "Deva", Email = "Deva@gmail.com", Experience = 2, Phone = "8856904770", Speciality = "surgen", Pic = "-" };

doctorRepository.Add(doct);

var result = doctorService.UpdateSpecialization(new DoctorSpecialDTO { Id=1 , Speciality = "dermatologist" });

var data = "dermatologist";

//Assert

Assert.AreEqual(data, result.Speciality);

}

#endregion

#region GeatAllDoctor

[Test]

public void GetAllDoctorTest()

//Arrange

{

IRepository<int, Doctor> doctorRepository = new DoctorRepository(context);

IDoctorService doctorService = new DoctorService(doctorRepository);

Doctor? doctor = new Doctor { Id = 1, Name = "Deva", Email = "Deva@gmail.com", Experience = 2, Phone = "8856904770", Speciality = "surgen", Pic = "-" };

doctorRepository.Add(doctor);

//Action

var result = doctorService.GetAllDoctor();

// Assert

Assert.NotNull(result);

Assert.AreEqual(1, result.Count);

}

#endregion

#region UpdatePhone

[Test]

public void UpdatePhoneTest()

{

//Arrange

IRepository<int, Doctor> doctorRepository = new DoctorRepository(context);

IDoctorService doctorService = new DoctorService(doctorRepository);

Doctor? doct = new Doctor { Id = 1, Name = "Deva", Email = "Deva@gmail.com", Experience = 2, Phone = "8856904770", Speciality = "surgen", Pic = "-" };

doctorRepository.Add(doct);

var result = doctorService.UpdatePhone(new Doctor { Id = 1, Phone="885690477056" });

var data = "885690477056";

//Assert

Assert.AreEqual(data, result.Phone);

}

#endregion

#region Update Email

[Test]

public void UpdateEmailTest()

{

//Arrange

IRepository<int, Doctor> doctorRepository = new DoctorRepository(context);

IDoctorService doctorService = new DoctorService(doctorRepository);

Doctor? doct = new Doctor { Id = 1, Name = "Deva", Email = "Deva@gmail.com", Experience = 2, Phone = "8856904770", Speciality = "surgen", Pic = "-" };

doctorRepository.Add(doct);

var result = doctorService.UpdateEmail(new Doctor { Id = 1, Email="abc@gmail.com" });

var data = "abc@gmail.com";

//Assert

Assert.AreEqual(data, result.Email);

}

#endregion

}

}

Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to *target*. Leet Code

class Solution {

   public int[] twoSum(int[] nums, int target) {

        int[] ans = new int[2];

        int n = nums.length;

        for (int i = 0; i < n; i++) {

            for (int j = i + 1; j < n; j++) {

                if (nums[j] == target- nums[i]) {

                    ans[0] = i;

                    ans[1] = j;

                    return ans;

                }

            }

        }

        return ans;

    }

}