Course End Project

**Create a Prototype for Joe’s Pizza Portal Using Selenium, NUnit, and SpecFlow to Develop and Test. Deploy it into Azure VM Using Jenkins**

**Source Code:**

**Pizza.cs**

namespace PhaseEndProjectPizza.Models

{

public class Pizza

{

public int PizzaId { get; set; }

public string? PizzaType { get; set; }

public double Price { get; set; }

}

}

**Order.cs**

namespace PhaseEndProjectPizza.Models

{

public class Order

{

public string? Pizza { get; set; }

public int Quantity { get; set; }

public double Amount { get; set; }

}

}

**ConfirmOrder.cs**

namespace PhaseEndProjectPizza.Models

{

public class ConfirmOrder

{

public string? OrderId { get; set; }

public string? Pizza { get; set; }

public int Quantity { get; set; }

public double Amount { get; set; }

}

}

**PizzaController.cs**

using Microsoft.AspNetCore.Mvc;

using PhaseEndProjectPizza.Models;

namespace PhaseEndProjectPizza.Controllers

{

public class PizzaController : Controller

{

// Dummy data for pizza types

private static readonly List<Pizza> PizzaTypes = new List<Pizza>

{

new Pizza { PizzaId = 1, PizzaType = "Margherita", Price = 300.45 },

new Pizza { PizzaId = 2, PizzaType = "Cheese Pizza", Price = 400.50 },

new Pizza { PizzaId = 3, PizzaType = "Panneer Pizza", Price = 200.50 },

new Pizza { PizzaId = 4, PizzaType = "Mushroom Pizza", Price = 450.50 },

new Pizza { PizzaId = 5, PizzaType = "Plain Pizza", Price = 99.50 }

// Add more pizza types as needed

};

[HttpGet]

public IActionResult PizzaSelection()

{

// Pass pizza types to the PizzaSelection view

return View(PizzaTypes);

}

[HttpGet]

public IActionResult OrderCheckout(string pizzaType)

{

// Get the selected pizza based on the pizza type

var selectedPizza = PizzaTypes.FirstOrDefault(pizza => pizza.PizzaType == pizzaType);

if (selectedPizza == null)

{

// Handle invalid pizza type

return RedirectToAction("PizzaSelection");

}

// Pass data to the OrderCheckout view

var model = new Order

{

Pizza = pizzaType

};

return View(model);

}

[HttpPost]

public IActionResult OrderConfirmation(string pizzaType, int quantity)

{

// Your logic to process the order (save to database, etc.)

// Retrieve pizza details based on the selected pizza in the order

var selectedPizza = PizzaTypes.FirstOrDefault(pizza => pizza.PizzaType == pizzaType);

if (selectedPizza == null)

{

// Handle invalid pizza type

return RedirectToAction("PizzaSelection");

}

// Calculate the total order amount

var orderAmount = CalculateOrderAmount(selectedPizza.Price, quantity);

// For simplicity, assuming you save the order and retrieve the order details

var confirmedOrder = new ConfirmOrder

{

OrderId = GenerateOrderId(),

Pizza = selectedPizza.PizzaType,

Quantity = quantity,

Amount = orderAmount

};

// Redirect to OrderConfirmation view with the confirmed order details

return View("OrderConfirmation", confirmedOrder);

}

public string GenerateOrderId()

{

// Replace this with your actual logic to generate a unique order ID

// For simplicity, returning a dummy order ID

return Guid.NewGuid().ToString();

}

public double CalculateOrderAmount(double pizzaPrice, int quantity)

{

// Replace this with your actual logic to calculate the order amount

// For simplicity, returning a fixed amount for each pizza type

return pizzaPrice \* quantity;

}

public void ConfigureServices(ServiceCollection services)

{

throw new NotImplementedException();

}

}

}

**PizzaPort.feature**

Feature: PizzaPort

*As a pizza enthusiast*

*I want to order pizzas*

*So that I can enjoy delicious pizzas at home*

Scenario: Order a Pizza

Given I am on the Pizza Selection page

When I select "Pepperoni Pizza" from the menu

And I proceed to checkout

Then I should be on the Order Checkout page

Scenario: Invalid Pizza Type

Given I am on the Pizza Selection page

When I select an invalid pizza type "InvalidPizza"

Then I should be redirected to the Pizza Selection page

**PizzaPortStepDefinitions.cs**

using System;

using Microsoft.AspNetCore.Mvc;

using NUnit.Framework;

using PhaseEndProjectPizza.Controllers;

using PhaseEndProjectPizza.Models;

using TechTalk.SpecFlow;

using Assert = NUnit.Framework.Assert;

namespace SpecFlowPizza.StepDefinitions

{

[Binding]

public class PizzaStepDefinitions

{

private readonly PizzaController pizzaController;

private IActionResult actionResult;

private ViewResult viewResult;

public PizzaStepDefinitions()

{

pizzaController = new PizzaController();

}

[Given(@"I am on the Pizza Selection page")]

public void GivenIAmOnThePizzaSelectionPage()

{

actionResult = pizzaController.PizzaSelection();

}

[When(@"I select ""([^""]\*)"" from the menu")]

public void WhenISelectFromTheMenu(string pizzaType)

{

actionResult = pizzaController.OrderCheckout(pizzaType);

}

[When(@"I proceed to checkout")]

public void WhenIProceedToCheckout()

{

viewResult = actionResult as ViewResult;

Assert.IsNotNull(viewResult);

}

[Then(@"I should be on the Order Checkout page")]

public void ThenIShouldBeOnTheOrderCheckoutPage()

{

Assert.IsNotNull(viewResult);

}

[When(@"I select an invalid pizza type ""([^""]\*)""")]

public void WhenISelectAnInvalidPizzaType(string invalidPizza)

{

actionResult = pizzaController.OrderCheckout(invalidPizza);

}

[Then(@"I should be redirected to the Pizza Selection page")]

public void ThenIShouldBeRedirectedToThePizzaSelectionPage()

{

var redirectToActionResult = actionResult as RedirectToActionResult;

Assert.IsNotNull(redirectToActionResult);

Assert.AreEqual("PizzaSelection", redirectToActionResult.ActionName);

}

}

}

**PizzaControllerTests.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using Microsoft.AspNetCore.Mvc;

using NUnit.Framework;

using PhaseEndProjectPizza.Controllers;

using PhaseEndProjectPizza.Models;

namespace ClassLibraryPizza

{

[TestFixture]

public class PizzaControllerTests

{

private PizzaController pizzaController;

[SetUp]

public void Setup()

{

// Assuming you have any necessary setup logic here

pizzaController = new PizzaController();

}

[Test]

public void PizzaSelection\_ReturnsView()

{

// Arrange & Act

var result = pizzaController.PizzaSelection();

// Assert

Assert.IsNotNull(result);

Assert.IsInstanceOf<ViewResult>(result);

}

[Test]

public void OrderCheckout\_WithValidPizzaType\_ReturnsView()

{

// Arrange & Act

var pizzaType = "Margherita";

var result = pizzaController.OrderCheckout(pizzaType);

// Assert

Assert.IsNotNull(result);

Assert.IsInstanceOf<ViewResult>(result);

// Additional assertion for the model passed to the view

var model = (result as ViewResult)?.Model as Order;

Assert.IsNotNull(model);

Assert.AreEqual(pizzaType, model.Pizza);

}

[Test]

public void OrderCheckout\_WithInvalidPizzaType\_RedirectsToPizzaSelection()

{

// Arrange & Act

var invalidPizzaType = "InvalidPizzaType";

var result = pizzaController.OrderCheckout(invalidPizzaType) as RedirectToActionResult;

// Assert

Assert.IsNotNull(result);

Assert.AreEqual("PizzaSelection", result.ActionName);

}

[Test]

public void OrderConfirmation\_WithValidOrder\_ReturnsView()

{

// Arrange & Act

var pizzaType = "Margherita";

var quantity = 2;

var result = pizzaController.OrderConfirmation(pizzaType, quantity);

// Assert

Assert.IsNotNull(result);

Assert.IsInstanceOf<ViewResult>(result);

// Additional assertion for the model passed to the view

var model = (result as ViewResult)?.Model as ConfirmOrder;

Assert.IsNotNull(model);

Assert.AreEqual(pizzaType, model.Pizza);

Assert.AreEqual(quantity, model.Quantity);

}

[Test]

public void OrderConfirmation\_WithInvalidPizzaType\_RedirectsToPizzaSelection()

{

// Arrange & Act

var invalidPizzaType = "InvalidPizzaType";

var quantity = 3;

var result = pizzaController.OrderConfirmation(invalidPizzaType, quantity) as RedirectToActionResult;

// Assert

Assert.IsNotNull(result);

Assert.AreEqual("PizzaSelection", result.ActionName);

}

// Add more tests as needed for your specific actions in PizzaController

}

}