Project

Joe's Pizza Portal

Source code:

CartController:

```
using Microsoft.AspNetCore.Mvc;
using PizzaHub.Infrastructure;
using PizzaHub.Models;
using PizzaHub.Models.ViewModels;

namespace PizzaHub.Controllers
{
   public class CartController : Controller
   {
      private readonly DataContext _context;

      public CartController(DataContext context)
      {
         _context = context;
      }

      public IActionResult Index()
      {
         List<CartItem> cart =
HttpContext.Session.GetJson<List<CartItem>>("Cart") ?? new List<CartItem>();

         CartViewModel cartVM = new()
         {
            CartItems = cart,
            GrandTotal = cart.Sum(x => x.Quantity * x.Price)
         };

         return View(cartVM);
      }

      public async Task<IActionResult> Add(long id)
      {
         Product product = await _context.Products.FindAsync(id);
```

```
            List<CartItem> cart =
HttpContext.Session.GetJson<List<CartItem>>("Cart") ?? new List<CartItem>();

            CartItem cartItem = cart.Where(c => c.ProductId == id).FirstOrDefault();

            if (cartItem == null)
            {
                cart.Add(new CartItem(product));
            }
            else
            {
                cartItem.Quantity += 1;
            }

            HttpContext.Session.SetJson("Cart", cart);

            TempData["Success"] = "The product has been added!";

            return Redirect(Request.Headers["Referer"].ToString());
        }

        public async Task<IActionResult> Decrease(long id)
        {
            List<CartItem> cart =
HttpContext.Session.GetJson<List<CartItem>>("Cart");

            CartItem cartItem = cart.Where(c => c.ProductId == id).FirstOrDefault();

            if (cartItem.Quantity > 1)
            {
                --cartItem.Quantity;
            }
            else
            {
                cart.RemoveAll(p => p.ProductId == id);
            }

            if (cart.Count == 0)
            {
```

```
            HttpContext.Session.Remove("Cart");
        }
        else
        {
            HttpContext.Session.SetJson("Cart", cart);
        }

        TempData["Success"] = "The product has been removed!";

        return RedirectToAction("Index");
    }

    public async Task<IActionResult> Remove(long id)
    {
        List<CartItem> cart =
HttpContext.Session.GetJson<List<CartItem>>("Cart");

        cart.RemoveAll(p => p.ProductId == id);

        if (cart.Count == 0)
        {
            HttpContext.Session.Remove("Cart");
        }
        else
        {
            HttpContext.Session.SetJson("Cart", cart);
        }

        TempData["Success"] = "The product has been removed!";

        return RedirectToAction("Index");
    }

    public IActionResult Clear()
    {
        HttpContext.Session.Remove("Cart"); if
(HttpContext.Request.Headers["X-Requested-With"] != "XMLHttpRequest")
            return Redirect(Request.Headers["Referer"].ToString());

        return Ok();
```

```
        }
    }
}
```

HomeController:

```
using Microsoft.AspNetCore.Mvc;
using PizzaHub.Models;
using System.Diagnostics;

namespace PizzaHub.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;

        public HomeController(ILogger<HomeController> logger)
        {
            _logger = logger;
        }

        public IActionResult Index()
        {
            return View();
        }

        public IActionResult Privacy()
        {
            return View();
        }

        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None,
NoStore = true)]
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ??
HttpContext.TraceIdentifier });
        }
    }
}
```

ProductController:

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using PizzaHub.Infrastructure;
using PizzaHub.Models;

namespace PizzaHub.Controllers
{
   public class ProductsController : Controller
   {
      private readonly DataContext _context;

      public ProductsController(DataContext context)
      {
         _context = context;
      }

      public async Task<IActionResult> Index(string categorySlug = "", int p = 1)
      {
         int pageSize = 3;
         ViewBag.PageNumber = p;
         ViewBag.PageRange = pageSize;
         ViewBag.CategorySlug = categorySlug;

         if (categorySlug == "")
         {
            ViewBag.TotalPages =
(int)Math.Ceiling((decimal)_context.Products.Count() / pageSize);

            return View(await _context.Products.OrderByDescending(p =>
p.Id).Skip((p - 1) * pageSize).Take(pageSize).ToListAsync());
         }

         Category category = await _context.Categories.Where(c => c.Slug ==
categorySlug).FirstOrDefaultAsync();
         if (category == null) return RedirectToAction("Index");

         var productsByCategory = _context.Products.Where(p => p.CategoryId ==
category.Id);
```

```
            ViewBag.TotalPages =
(int)Math.Ceiling((decimal)productsByCategory.Count() / pageSize);

            return View(await productsByCategory.OrderByDescending(p =>
p.Id).Skip((p - 1) * pageSize).Take(pageSize).ToListAsync());
        }
    }
}
```

SeedData.cs:

```
using Microsoft.EntityFrameworkCore;
using PizzaHub.Models;

namespace PizzaHub.Infrastructure
{
    public class SeedData
    {
        public static void SeedDatabase(DataContext context)
        {
            context.Database.Migrate();

            if (!context.Products.Any())
            {
                Category veg = new Category { Name = "Veg", Slug = "veg" };
                Category non_veg = new Category { Name = "Non_Veg", Slug =
"non_veg" };

                context.Products.AddRange(
                    new Product
                    {
                        Name = "Margherita",
                        Slug = "margherita",
                        Description = "Tasty Cheese Loaded",
                        Price = 199,
                        Category = veg,
                        Image = "Margherita.jpg"
                    },
                    new Product
```

```
{
    Name = "Farmhouse",
    Slug = "farmhouse",
    Description = "Loaded with Vegetables",
    Price = 300,
    Category = veg,
    Image = "Farmhouse.jpg"
},
new Product
{
    Name = "Peppy Paneer",
    Slug = "peppy-paneer",
    Description = "Paneer Loaded",
    Price = 250,
    Category = veg,
    Image = "Peppy_Paneer.jpg"
},
new Product
{
    Name = "Veg Extravaganza",
    Slug = "veg-extravaganza",
    Description = "Extra Vegetables",
    Price = 250,
    Category = veg,
    Image = "Veg_Extravaganza.jpg"
},
new Product
{
    Name = "Chicken Dominator",
    Slug = "chicken-dominator",
    Description = "Chicken Dominator",
    Price = 300,
    Category = non_veg,
    Image = "Chicken Dominator.jpg"
},
new Product
{
    Name = "Chicken Sausage",
    Slug = "chicken-sausage",
    Description = "Chicken Sausage",
```

```
                Price = 350,
                Category = non_veg,
                Image = "Chicken Sausage.jpg"
            },
            new Product
            {
                Name = "Chicken Fiesta",
                Slug = "chicken-fiesta",
                Description = "Chicken Fiesta",
                Price = 350,
                Category = non_veg,
                Image = "Chicken Fiesta.jpg"
            },
            new Product
            {
                Name = "Chicken Delight",
                Slug = "chicken-delight",
                Description = "Chicken Delight",
                Price = 300,
                Category = non_veg,
                Image = "Chicken Delight.jpg"
            }
        );

        context.SaveChanges();
    }
  }
 }
}
```

DataContext.cs:

```
using Microsoft.EntityFrameworkCore;
using PizzaHub.Models;

namespace PizzaHub.Infrastructure
{
  public class DataContext : DbContext
  {
```

```
        public DataContext(DbContextOptions<DataContext> options) :
base(options) { }


        public DbSet<Product> Products { get; set; }
        public DbSet<Category> Categories { get; set; }
    }




}
```

Components:
SmallCart.cs:

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using PizzaHub.Models;
using PizzaHub.Models.ViewModels;

namespace PizzaHub.Infrastructure.Components
{
    public class SmallCartViewComponent: ViewComponent
    {
        public IViewComponentResult Invoke()
        {
            List<CartItem> cart =
HttpContext.Session.GetJson<List<CartItem>>("Cart");
            SmallCartViewModel smallCartVM;

            if (cart == null || cart.Count == 0)
            {
                smallCartVM = null;
            }
            else
            {
                smallCartVM = new()
                {
                    NumberOfItems = cart.Sum(x => x.Quantity),
```

```
                TotalAmount = cart.Sum(x => x.Quantity * x.Price)
            };
        }

        return View(smallCartVM);
    }
  }
}
```

Migrations:

```
using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace PizzaHub.Migrations
{
  public partial class initial : Migration
  {
    protected override void Up(MigrationBuilder migrationBuilder)
    {
      migrationBuilder.CreateTable(
        name: "Categories",
        columns: table => new
        {
          Id = table.Column<long>(type: "bigint", nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
          Name = table.Column<string>(type: "nvarchar(max)", nullable: true),
          Slug = table.Column<string>(type: "nvarchar(max)", nullable: true)
        },
        constraints: table =>
        {
          table.PrimaryKey("PK_Categories", x => x.Id);
        });

      migrationBuilder.CreateTable(
        name: "Products",
        columns: table => new
        {
          Id = table.Column<long>(type: "bigint", nullable: false)
```

```
                .Annotation("SqlServer:Identity", "1, 1"),
            Name = table.Column<string>(type: "nvarchar(max)", nullable: false),
            Slug = table.Column<string>(type: "nvarchar(max)", nullable: true),
            Description = table.Column<string>(type: "nvarchar(max)", nullable:
false),
            Price = table.Column<decimal>(type: "decimal(8,2)", nullable: false),
            CategoryId = table.Column<long>(type: "bigint", nullable: false),
            Image = table.Column<string>(type: "nvarchar(max)", nullable: true)
        },
        constraints: table =>
        {
            table.PrimaryKey("PK_Products", x => x.Id);
            table.ForeignKey(
                name: "FK_Products_Categories_CategoryId",
                column: x => x.CategoryId,
                principalTable: "Categories",
                principalColumn: "Id",
                onDelete: ReferentialAction.Cascade);
        });

    migrationBuilder.CreateIndex(
        name: "IX_Products_CategoryId",
        table: "Products",
        column: "CategoryId");
}

protected override void Down(MigrationBuilder migrationBuilder)
{
    migrationBuilder.DropTable(
        name: "Products");

    migrationBuilder.DropTable(
        name: "Categories");
}
}
}
```

CartItem.cs:

```
namespace PizzaHub.Models
{
    public class CartItem
    {
        public long ProductId { get; set; }
        public string ProductName { get; set; }
        public int Quantity { get; set; }
        public decimal Price { get; set; }
        public decimal Total
        {
            get { return Quantity * Price; }
        }
        public string Image { get; set; }

        public CartItem()
        {
        }

        public CartItem(Product product)
        {
            ProductId = product.Id;
            ProductName = product.Name;
            Price = product.Price;
            Quantity = 1;
            Image = product.Image;
        }

    }
}
```

Category.cs:

```
namespace PizzaHub.Models
{
    public class Category
    {
        public long Id { get; set; }

        public string Name { get; set; }
```

```
        public string Slug { get; set; }
    }
}
```

Product.cs:

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace PizzaHub.Models
{
    public class Product
    {
        public long Id { get; set; }

        [Required(ErrorMessage = "Please enter a value")]
        public string Name { get; set; }

        public string Slug { get; set; }
        [Required, MinLength(4, ErrorMessage = "Minimum length is 4")]
        public string Description { get; set; }
        [Required]
        [Range(0.01, double.MaxValue, ErrorMessage = "Please enter a value")]
        [Column(TypeName = "decimal(8, 2)")]
        public decimal Price { get; set; }

        public long CategoryId { get; set; }

        public Category Category { get; set; }

        public string Image { get; set; }
    }
}
```

Views:

Cart:

Paypalpartial.cshtml:

```
@*
    For more information on enabling MVC for empty projects, visit
http://go.microsoft.com/fwlink/?LinkID=397860
*@
@{
    int count = 1;
}

<form class="paypalform" action="https://www.paypal.com/us/cgi-bin/webscr"
method="post">
    <input type="hidden" name="cmd" value="_cart">
    <input type="hidden" name="upload" value="1">
    <input type="hidden" name="business" value="riyajhingani238@gmail.com">

    @foreach (var item in Model)
    {
        <input type="hidden" name="item_name_@count"
value="@item.ProductName">
        <input type="hidden" name="amount_@count" value="@item.Price">
        <input type="hidden" name="quantity_@count" value="@item.Quantity">
        count++;
    }

    <input type="hidden" name="currency_code" value="USD">
    <input type="image" src="http://www.paypal.com/en_US/i/btn/x-click-
but01.gif" name="submit" alt="Make payments with PayPal - it's fast, free and
secure!">
</form>
```

Index.cshtml:

```
@model CartViewModel

@{
    ViewData["Title"] = "Cart Overview";
}


@if (Model.CartItems.Count > 0)
{

  <div class="cartWrapper">

    <div class="cartbg d-none">
      <h3 class="text-center">Redirecting you to paypal</h3>
      </div>
    <table class="table">
        <tr>
            <th>Product</th>
            <th>Quantity</th>
            <th></th>
            <th>Price</th>
            <th>Total</th>
        </tr>
        @foreach (var item in Model.CartItems)
        {
            <tr>
                <td>@item.ProductName</td>
                <td>@item.Quantity</td>
                <td>
                    <a class="btn btn-primary btn-sm" asp-action="Add"
asp-route-id="@item.ProductId">+</a>
                    <a class="btn btn-info btn-sm" asp-action="Decrease"
asp-route-id="@item.ProductId">-</a>
                    <a class="btn btn-danger btn-sm" asp-action="Remove"
asp-route-id="@item.ProductId">Remove</a>
                </td>
                <td>@item.Price.ToString("C2")</td>
```

```
                    <td>@Model.CartItems.Where(x => x.ProductId ==
item.ProductId).Sum(x => x.Quantity * x.Price).ToString("C2")</td>
                </tr>
            }
            <tr>
                <td class="text-right" colspan="4">Grand Total:
@Model.GrandTotal.ToString("C2")</td>
            </tr>
            <tr>
                <td class="text-right" colspan="4">
                    <a class="btn btn-danger" asp-action="Clear">Clear Cart</a>
                    <a class="btn btn-primary checkout" href="#">Checkout</a>
                </td>
            </tr>
        </table>
    </div>


}
else
{
    <h3 class="display-4 text-center">Your cart is empty.</h3>
}

<partial name="~/Views/Cart/_PaypalPartial.cshtml" for="CartItems" />

@section Scripts {

    <script>
    $(function () {
        $("a.checkout").click(function (e) {
            e.preventDefault();
            $("div.cartbg").removeClass("d-none");
            $.get("/cart/clear", {}, function () {
                $("form.paypalform").submit();
            });

        });
    });
    </script>
```

```
}
```

Products:

Index.cshtml:

```
@model IEnumerable<Product>

@{
    ViewData["Title"] = "Products";
}

<h1 background-color="pink">Pizzas</h1>

<div class="row">
    @foreach (var item in Model)
    {
        <div class="col-4">
            <img src="/media/products/@item.Image" class="img-fluid" alt="" 
/>
            <h4>@item.Name</h4>
            <div>
                @Html.Raw(item.Description)
            </div>
            <p>
                @item.Price.ToString("C2")
            </p>
            <p>
                <a class="btn btn-primary" asp-controller="Cart" asp-
action="Add" asp-route-id="@item.Id">Add to cart</a>
            </p>
        </div>
    }


    @if (ViewBag.TotalPages > 1)
    {
        <div class="d-flex w-100 justify-content-center">
            <pagination page-count="@ViewBag.TotalPages"
              page-target="/products/@ViewBag.CategorySlug"
```

```
                page-number="@ViewBag.PageNumber"
                page-range="@ViewBag.PageRange">
            </pagination>
        </div>
    }

</div>
```

PizzaHub.feature:

```
Feature: Pizzahub

Placing a pizza order

@tag1
Scenario: pizza order should be placed by me
        Given i have to navigate to pizzahub page
        Then click on all pizza button and add pizza to cart
        Then click on veg pizza button and add pizza to cart
        Then click on view cart butoon to check order list
        Then add or remove pizzas
        When click on checkout button order should be placed
```

StepDefinition:

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using System;
using TechTalk.SpecFlow;

namespace SpecFlowCalculator.Specs.StepDefinitions
{
   [Binding]
   public class PizzahubStepDefinitions
   {
      public ChromeDriver driver;

      [Given(@"i have to navigate to pizzahub page")]
      public void GivenIHaveToNavigateToPizzahubPage()
```

```
{
    driver = new ChromeDriver();
    driver.Manage().Window.Maximize();
    driver.Navigate().GoToUrl("https://localhost:1007/");

}

[Then(@"click on all pizza button and add pizza to cart")]
public void GivenClickOnAllPizzaButtonAndAddPizzaToCart()
{
    driver.FindElement(By.LinkText("All Pizaa\'s")).Click();
    driver.FindElement(By.LinkText("Add to cart")).Click();
    driver.FindElement(By.LinkText("2")).Click();
    Thread.Sleep(1000);
    driver.FindElement(By.LinkText("Add to cart")).Click();
    driver.FindElement(By.LinkText("3")).Click();
    driver.FindElement(By.LinkText("Add to cart")).Click();
}

[Then(@"click on veg pizza button and add pizza to cart")]
public void ThenClickOnVegPizzaButtonAndAddPizzaToCart()
{
    driver.FindElement(By.LinkText("Veg")).Click();
    driver.FindElement(By.CssSelector(".col-4:nth-child(2) .btn")).Click();
    driver.FindElement(By.LinkText("Non_Veg")).Click();
    Thread.Sleep(1000);
    driver.FindElement(By.CssSelector(".col-4:nth-child(2) .btn")).Click();
}

[Then(@"click on view cart butoon to check order list")]
public void ThenClickOnViewCartButoonToCheckOrderList()
{
    driver.FindElement(By.LinkText("View Cart")).Click();
}

[Then(@"add or remove pizzas")]
public void ThenAddOrRemovePizzas()
{
    driver.FindElement(By.LinkText("Remove")).Click();
    driver.FindElement(By.LinkText("-")).Click();
```

```
            driver.FindElement(By.LinkText("+")).Click();
            driver.FindElement(By.CssSelector("tr:nth-child(3) .btn-primary")).Click();
            Thread.Sleep(1000);
            driver.FindElement(By.CssSelector("tr:nth-child(4) .btn-info")).Click();
        }

        [When(@"click on checkout button order should be placed")]
        public void WhenClickOnCheckoutButtonOrderShouldBePlaced()
        {
            driver.FindElement(By.LinkText("Checkout")).Click();
            driver.Close();
        }
    }
}
```