----------------------------Step by Step to create Final Assignment

**- Create a new .Net Core minimal API**

**- Create a new class model Category (ID, Name)**

**- Create one endpoint to get category data (hard-coded data)**

**- Run and test endpoint with postman**

1. Create a new .Net Core minimal API

dotnet new web -o API\_Final

1. **API\_Final project** Create a new class model Category (ID, Name)

---create Models folder -> create Category class:

namespace API\_Final.Models

{

    public class Category

    {

        public int ID { get; set; }

        public string? Name { get; set; }

    }

}

**3:** API\_Final Project: Create one endpoint to get category data (hard-coded data)

+ Update the Program.cs file in the API\_Final project to use the correct namespace for the Category class.

using Final\_Assignment\_Dung.Domain.Models;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

    app.UseSwagger();

    app.UseSwaggerUI();

}

//app.UseHttpsRedirection();

app.MapGet("/categories", () =>

{

    var categories = new List<Category>

    {

        new Category { ID = 1, Name = "Electronics" },

        new Category { ID = 2, Name = "Books" },

        new Category { ID = 3, Name = "Clothing" },

        new Category { ID = 4, Name = "Home & Kitchen" }

    };

    return categories;

})

.WithName("GetCategories")

.WithOpenApi();

app.Run();

**Connect SQL**

Using API\_Final project

1. **Add Entity Framework Core and SQL Server Packages**.
2. **Configure Services in Program.cs**.
3. **Create the ApplicationDbContext**.
4. **Update the Category Model**.
5. **Add a Migration and Update the Database**.
6. **Create a Controller to Retrieve Data from the Database**.
7. **Run and Test the Endpoint with Postman**

1. Add Entity Framework Core and SQL Server Packages: add API\_Final project

**dotnet add package Microsoft.EntityFrameworkCore**

**dotnet add package Microsoft.EntityFrameworkCore.SqlServer**

**dotnet add package Microsoft.EntityFrameworkCore.Design**

**2.** Configure Services in Program.cs:Update the Program.cs file to configure Entity Framework Core and SQL Server.

1. Seed hard-coded category data to Entity Framework Core: In ApplicationDbContext.cs:

using Microsoft.EntityFrameworkCore;

using API\_Final.Models;

namespace API\_Final.Data

{

public class ApplicationDbContext : DbContext

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options)

{

}

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

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

// Seed data

modelBuilder.Entity<Category>().HasData(

new Category { ID = 1, Name = "Electronics" },

new Category { ID = 2, Name = "Books" },

new Category { ID = 3, Name = "Clothing" },

new Category { ID = 4, Name = "Home & Kitchen" }

);

}

}

}

1. Change the get category endpoint to retrieve data from Entity Framework Core: In Program.cs:

app.MapGet("/categories", async (ApplicationDbContext context) =>

{

var categories = await context.Categories.ToListAsync();

return categories;

})

.WithName("GetCategories")

.WithOpenApi();

1. Run and test the endpoint with Postman: Ensure your database is up-to-date with the latest migrations:

dotnet ef migrations add InitialCreate

dotnet ef database update

1. Run your application:

dotnet run --project API\_Final

---------------------------------UI------------------------------------------

**- Create a .Net MVC Project**

**- Create a new menu nav bar called category**

**- Create a razor page and render hardcoded category data in table format**

**- Run the project"**

**1.** Create a New .NET MVC Project: Open a terminal and run the following command to create a new .NET MVC project:

dotnet new mvc -n UI\_Final

cd UI\_Final

2. Create a New Menu Nav Bar Called "Category": Update the \_Layout.cshtml file to add a new menu item for "Category".

file:///c%3A/Dung/C%23.NET/Final\_Assignment\_Dung/UI\_Final/Views/Shared/\_Layout.cshtml

 <li class="nav-item">

                            <a class="nav-link text-dark" asp-area="" asp-controller="Category" asp-action="Index">Category</a>

                        </li>

3. Create a Razor Page to Render Hardcoded Category Data in Table Format: Create a new Razor page called Index.cshtml in the Views/Category folder.

file:///c%3A/Dung/C%23.NET/Final\_Assignment\_Dung/UI\_Final/Views/Category/Index.cshtml

@model IEnumerable<UI\_Final.Models.Category>

@{

ViewData["Title"] = "Category";

}

<h1>Category</h1>

<table class="table">

<thead>

<tr>

<th>ID</th>

<th>Name</th>

</tr>

</thead>

<tbody>

@foreach (var category in Model)

{

<tr>

<td>@category.ID</td>

<td>@category.Name</td>

</tr>

}

</tbody>

</table>

5. Create the corresponding model for the Razor page.

file:///c%3A/Dung/C%23.NET/Final\_Assignment\_Dung/UI\_Final/Models/Category.cs

namespace UI\_Final.Models

{

public class Category

{

public int ID { get; set; }

public string Name { get; set; }

}

}

1. Create a controller to handle the Category page. file:///c%3A/Dung/C%23.NET/Final\_Assignment\_Dung/UI\_Final/Controllers/CategoryController.cs

using Microsoft.AspNetCore.Mvc;

using UI\_Final.Models;

using System.Collections.Generic;

namespace UI\_Final.Controllers

{

public class CategoryController : Controller

{

public IActionResult Index()

{

var categories = new List<Category>

{

new Category { ID = 1, Name = "Electronics" },

new Category { ID = 2, Name = "Books" },

new Category { ID = 3, Name = "Clothing" },

new Category { ID = 4, Name = "Home & Kitchen" }

};

return View(categories);

}

}

}

**- Integrate with HttpClientFactory for .Net MVC Project**

**- Use HttpClient to call category endpoint of .Net Core Minimal API to get all categories data**

**- Replace the hardcoded category data with data retrieved from API**

**- Run 2 projects and verify the results**

**1. Configure HttpClientFactory in the .NET MVC Project: Update the Program.cs file to add HttpClientFactory services.**

Program.cs

using System.Net.Http;

using System.Net.Security;

using System.Security.Cryptography.X509Certificates;

using UI\_Final.Services;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllersWithViews();

**builder.Services.AddHttpClient<CategoryService>()**

.ConfigurePrimaryHttpMessageHandler(() =>

{

return new HttpClientHandler

{

ServerCertificateCustomValidationCallback = (message, cert, chain, errors) => true

};

});

var app = builder.Build();

// Configure the HTTP request pipeline.

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();

**2. Create the Category Endpoint in the .NET Core Minimal API**

Ensure that the .NET Core Minimal API has an endpoint to get all categories data.

file:///c%3A/Dung/C%23.NET/Final\_Assignment\_Dung/API\_Final/Program.cs

using Microsoft.EntityFrameworkCore;

using API\_Final.Data;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddDbContext<ApplicationDbContext>(options =>

    options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

    app.UseSwagger();

    app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.MapGet("/categories", async (ApplicationDbContext context) =>

{

    var categories = await context.Categories.ToListAsync();

    return categories;

})

.WithName("GetCategories")

.WithOpenApi();

app.Run();

3. Use HttpClient to Call the Category Endpoint and Retrieve Data: Create a service to call the category endpoint and retrieve data.

Đoạn mã này là một phần của file CategoryService.cs trong dự án ASP.NET Core MVC. Nó định nghĩa một dịch vụ (CategoryService) để gọi API và lấy dữ liệu danh mục từ một API bên ngoài

CategoryService là một dịch vụ để gọi API và lấy dữ liệu danh mục từ một API bên ngoài.

Nó sử dụng HttpClient để gửi yêu cầu GET đến URL http://localhost:5225/categories và chuyển đổi phản hồi JSON thành một danh sách các đối tượng Category.

Phương thức GetCategoriesAsync là một phương thức bất đồng bộ, trả về một Task chứa danh sách các đối tượng Category.

file:///c%3A/Dung/C%23.NET/Final\_Assignment\_Dung/UI\_Final/Services/CategoryService.csCategoryService.cs

using System.Collections.Generic;

using System.Net.Http;

using System.Net.Http.Json;

using System.Threading.Tasks;

using UI\_Final.Models;

namespace UI\_Final.Services

{

public class CategoryService

{

private readonly HttpClient \_httpClient;

public CategoryService(HttpClient httpClient)

{

\_httpClient = httpClient;

}

public async Task<List<Category>> GetCategoriesAsync()

{

var categories = await \_httpClient.GetFromJsonAsync<List<Category>>("**http://localhost:5225/categories**");

return categories ?? new List<Category>();

}

}

}

4 **Register the CategoryService in the Program.cs file.**

file:///c%3A/Dung/C%23.NET/Final\_Assignment\_Dung/UI\_Final/Program.cs

builder.Services.AddHttpClient<CategoryService>();

3. Replace the Hardcoded Category Data with Data Retrieved from the API

Update the CategoryController to use the CategoryService to retrieve data from the API.

Phương thức này có nhiệm vụ lấy danh sách các danh mục (categories) từ dịch vụ [CategoryService](vscode-file://vscode-app/c:/Users/dungnguyentm/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.esm.html) và trả về một view để hiển thị danh sách này

CategoryController.cs

using Microsoft.AspNetCore.Mvc;

using UI\_Final.Models;

using UI\_Final.Services;

using System.Collections.Generic;

using System.Threading.Tasks;

namespace UI\_Final.Controllers

{

public class CategoryController : Controller

{

private readonly CategoryService \_categoryService;

public CategoryController(CategoryService categoryService)

{

\_categoryService = categoryService;

}

public async Task<IActionResult> Index()

{

var categories = await \_categoryService.GetCategoriesAsync();

return View(categories);

}

}

}

4. Run API project then run MVC project

**Create a Class Library Project.**

**Create CategoryDto in the Class Library.**

**Add Reference to the Class Library Project in Both API and .NET MVC Projects.**

**Use CategoryDto from the Class Library in Both API and .NET MVC Projects.**

Implementation

1. Create a Class Library Project

Open a terminal and run the following command to create a new class library project:

dotnet new classlib -n SharedLibrary

2. Create CategoryDto in the Class Library

Navigate to the SharedLibrary project directory and create a new file named CategoryDto.cs with the following content:

file:///c%3A/Dung/C%23.NET/Final\_Assignment\_Dung/SharedLibrary/CategoryDto.cs

namespace SharedLibrary

{

public class CategoryDto

{

public int ID { get; set; }

public string Name { get; set; }

}

}

* Giải thích: [CategoryDto](vscode-file://vscode-app/c:/Users/dungnguyentm/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.esm.html) là viết tắt của "Data Transfer Object" (Đối tượng Chuyển Dữ liệu), thường được sử dụng để truyền dữ liệu giữa các lớp hoặc các tầng trong ứng dụng.
* Lớp [CategoryDto](vscode-file://vscode-app/c:/Users/dungnguyentm/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.esm.html) này có thể được sử dụng trong cả dự án API và .NET MVC để truyền dữ liệu danh mục một cách nhất quán và tránh việc phải định nghĩa lại lớp này trong từng dự án.

3. Add Reference to the Class Library Project in Both API and .NET MVC Projects

Navigate to the root directory of your solution and run the following commands to add references to the SharedLibrary project in both the API and .NET MVC projects:

dotnet add API\_Final/API\_Final.csproj reference SharedLibrary/SharedLibrary.csproj

dotnet add UI\_Final/UI\_Final.csproj reference SharedLibrary/SharedLibrary.csproj

Lệnh này thêm một tham chiếu từ dự án API ([API\_Final](vscode-file://vscode-app/c:/Users/dungnguyentm/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.esm.html)) đến dự án thư viện lớp ([SharedLibrary](vscode-file://vscode-app/c:/Users/dungnguyentm/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.esm.html)). Điều này cho phép dự án API sử dụng các lớp và thành phần được định nghĩa trong dự án thư viện lớp.

Lệnh này thêm một tham chiếu từ dự án .NET MVC ([UI\_Final](vscode-file://vscode-app/c:/Users/dungnguyentm/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.esm.html)) đến dự án thư viện lớp ([SharedLibrary](vscode-file://vscode-app/c:/Users/dungnguyentm/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.esm.html)). Điều này cho phép dự án .NET MVC sử dụng các lớp và thành phần được định nghĩa trong dự án thư viện lớp.

4. Use CategoryDto from the Class Library in Both API and .NET MVC Projects

Update the API project to use CategoryDto from the SharedLibrary.

file:///c%3A/Dung/C%23.NET/Final\_Assignment\_Dung/API\_Final/Program.cs

using Microsoft.EntityFrameworkCore;

using API\_Final.Data;

**using SharedLibrary;**

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddDbContext<ApplicationDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.MapGet("/categories", async (ApplicationDbContext context) =>

{

var categories = await context.Categories

.Select(c => new **CategoryDto** { ID = c.ID, Name = c.Name })

.ToListAsync();

return categories;

})

.WithName("GetCategories")

.WithOpenApi();

app.Run();

Update the .NET MVC project to use CategoryDto from the SharedLibrary.

file:///c%3A/Dung/C%23.NET/Final\_Assignment\_Dung/UI\_Final/Services/CategoryService.cs

using System.Collections.Generic;

using System.Net.Http;

using System.Net.Http.Json;

using System.Threading.Tasks;

using SharedLibrary;

namespace UI\_Final.Services

{

public class CategoryService

{

private readonly HttpClient \_httpClient;

public CategoryService(HttpClient httpClient)

{

\_httpClient = httpClient;

}

public async Task<List<**CategoryDto**>> GetCategoriesAsync()

{

var categories = await \_httpClient.GetFromJsonAsync<List<**CategoryDto**>>("https://localhost:5225/categories");

return categories ?? new List<**CategoryDto**>();

}

}

}

Bằng cách làm theo các bước này, ta sẽ tạo một dự án thư viện lớp, định nghĩa CategoryDto trong thư viện lớp, thêm tham chiếu đến dự án thư viện lớp trong cả dự án API và .NET MVC, và sử dụng CategoryDto từ thư viện lớp thay vì tạo riêng biệt trong cả hai dự án.

5. Update the View to Use CategoryDto from SharedLibrary

Update the view to expect a model of type IEnumerable<SharedLibrary.CategoryDto>.

file:///c%3A/Dung/C%23.NET/Final\_Assignment\_Dung/UI\_Final/Views/Category/Index.cshtml

**@model IEnumerable<SharedLibrary.CategoryDto>**

@{

ViewData["Title"] = "Category";

}

<h1>Category</h1>

<table class="table">

<thead>

<tr>

<th>ID</th>

<th>Name</th>

</tr>

</thead>

<tbody>

@foreach (var category in Model)

{

<tr>

<td>@category.ID</td>

<td>@category.Name</td>

</tr>

}

</tbody>

</table>

1. **@model IEnumerable<SharedLibrary.CategoryDto>**:
   * Khai báo rằng mô hình được sử dụng trong view này là một danh sách các đối tượng [CategoryDto](vscode-file://vscode-app/c:/Users/dungnguyentm/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.esm.html).
   1. Update the Controller to Use CategoryDto from SharedLibrary

Ensure that the controller is using CategoryDto from SharedLibrary.

file:///c%3A/Dung/C%23.NET/Final\_Assignment\_Dung/UI\_Final/Controllers/CategoryController.cs

using Microsoft.AspNetCore.Mvc;

using UI\_Final.Services;

using System.Collections.Generic;

using System.Threading.Tasks;

**using SharedLibrary;**

namespace UI\_Final.Controllers

{

public class CategoryController : Controller

{

private readonly CategoryService \_categoryService;

public CategoryController(CategoryService categoryService)

{

\_categoryService = categoryService;

}

public async Task<IActionResult> Index()

{

var categories = await \_categoryService.GetCategoriesAsync();

return View(categories);

}

}

}

Update file:///c%3A/Dung/C%23.NET/Final\_Assignment\_Dung/API\_Final/Data/ApplicationDbContext.cs

Note:

Delete API\_Final/Models/Category.cs & API\_Final/Models/Category.cs

Remove hardcode in API\_Final/Data/ApplicationDbContext.cs

using Microsoft.EntityFrameworkCore;

using SharedLibrary;

namespace API\_Final.Data

{

    public class ApplicationDbContext : DbContext

    {

        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

            : base(options)

        {

        }

        public DbSet<CategoryDto> Categories { get; set; }

    }

}

1. Run comment: dotnet build for 3 projects then dotnet run for API project then is UI project—

--------- ------------------------------------------------------------------------------------------------------

**- Start building Home Page: Category Menu, Featured Product (Hard-coded data)**

1. Tại SharedLibrary project: create a file ProductDto.cs

namespace SharedLibrary

{

    public class ProductDto

    {

        public int ID { get; set; }

        public string? Name { get; set; }

        public decimal Price { get; set; }

        public string? PriceCurrency { get; set; }

        public string? ImageUrl { get; set; }

    }

}

1. Tạo một file mới có tên HomeViewModel.cs trong thư mục Models của dự án file:///c%3A/Dung/C%23.NET/Final\_Assignment\_Dung/UI\_Final.

file:///c%3A/Dung/C%23.NET/Final\_Assignment\_Dung/UI\_Final/Models/HomeViewModel.cs

using System.Collections.Generic;

using SharedLibrary;

namespace UI\_Final.Models

{

    public class HomeViewModel

    {

        public List<CategoryDto> Categories { get; set; } = new List<CategoryDto>();

        public List<ProductDto> FeaturedProducts { get; set; } = new List<ProductDto>();

    }

}

Cập nhật HomeController để sử dụng [HomeViewModel](vscode-file://vscode-app/c:/Users/dungnguyentm/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.esm.html) từ namespace UI\_Final.Models.

Add hard code data cho Feature product

#### file:///c%3A/Dung/C%23.NET/Final\_Assignment\_Dung/UI\_Final/Controllers/HomeController.cs

using System.Diagnostics;

using Microsoft.AspNetCore.Mvc;

using UI\_Final.Models;

using SharedLibrary;

using UI\_Final.Services;

namespace UI\_Final.Controllers

{

    public class HomeController : Controller

    {

        private readonly ILogger<HomeController> \_logger;

        private readonly CategoryService \_categoryService;

        public HomeController(ILogger<HomeController> logger, CategoryService categoryService)

        {

            \_logger = logger;

            \_categoryService = categoryService;

        }

        public async Task<IActionResult> Index()

        {

            var categories = await \_categoryService.GetCategoriesAsync();

            var featuredProducts = new List<ProductDto>

            {

                new ProductDto { ID = 1, Name = "Dép quai ngang Puffy Wide-Strap", Price = 1290000M, PriceCurrency = "VND", ImageUrl = "/images/depquaingangm1.jpg" },

                new ProductDto { ID = 2, Name = "Túi xách hình thang Metallic-Accent Double Handle", Price = 2390000M, PriceCurrency = "VND", ImageUrl = "/images/tuim1.jpg" },

                new ProductDto { ID = 3, Name = "Dép quai chéo Strappy Buckled", Price = 1450000M, PriceCurrency = "VND", ImageUrl = "/images/depquaiden1.jpg" }

            };

            var model = new HomeViewModel

            {

                Categories = categories,

                FeaturedProducts = featuredProducts

            };

            return View(model);

        }

        public async Task<IActionResult> CategoryList()

        {

            var categories = await \_categoryService.GetCategoriesAsync();

            return PartialView("\_CategoryList", categories);

        }

        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 });

        }

    }

}

**Display Category list trên header Home page**

1. Tạo new file in folders UI\_Final>View>Shared>**Components>CategoryList>Default.cshtml**

@model IEnumerable<SharedLibrary.CategoryDto>

@foreach (var category in Model)

{

    <li class="nav-item">

        <a class="nav-link text-dark" href="#">@category.Name</a>

    </li>

}

1. Create folder: View Component in UI\_Final project, add file CategoryListViewComponent .cs sẽ lấy danh sách các danh mục từ dịch vụ CategoryService và hiển thị chúng trong một view

using Microsoft.AspNetCore.Mvc;

using SharedLibrary;

using UI\_Final.Services;

using System.Threading.Tasks;

namespace UI\_Final.ViewComponents

{

    public class CategoryListViewComponent : ViewComponent

    {

        private readonly CategoryService \_categoryService;

        public CategoryListViewComponent(CategoryService categoryService)

        {

            \_categoryService = categoryService;

        }

        public async Task<IViewComponentResult> InvokeAsync()

        {

            var categories = await \_categoryService.GetCategoriesAsync();

            return View(categories);

        }

    }

}

1. Update layout: UI\_Final > Views>Shared>\_Layout.cshtml

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="utf-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>@ViewData["Title"] - Dung Shop</title>

    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />

    <link rel="stylesheet" href="~/UI\_Final.styles.css" asp-append-version="true" />

</head>

<body>

    <header>

        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">

            <div class="container-fluid">

                <!-- Cột chứa navbar-brand và danh sách Category -->

                <div class="col-12">

                    <!-- Hàng chứa navbar-brand -->

                    <div class="row">

                        <div class="col-12">

                            <a class="navbar-brand fw-bold fs-4" asp-area="" asp-controller="Home" asp-action="Index" style="width: 298px; height: 20px;">Dung Shop</a>

                        </div>

                    </div>

                    <!-- Hàng rỗng để tạo khoảng cách -->

                    <div class="row">

                        <div class="col-12">

                        </div>

                    </div>

                    <!-- Hàng chứa danh sách Category -->

                    <div class="row">

                        <div class="col-12">

                            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse" aria-controls="navbarSupportedContent"

                                    aria-expanded="false" aria-label="Toggle navigation">

                                <span class="navbar-toggler-icon"></span>

                            </button>

                            <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">

                                <ul class="navbar-nav flex-grow-1">

                                    <li class="nav-item">

                                        <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Trang chủ</a>

                                    </li>

                                    <!-- Sử dụng View Component để hiển thị danh sách Category -->

                                    @await Component.InvokeAsync("CategoryList")

                                </ul>

                            </div>

                        </div>

                    </div>

                </div>

            </div>

        </nav>

    </header>

    <div class="container">

        <main role="main" class="pb-3">

            @RenderBody()

        </main>

    </div>

    <footer class="border-top footer text-muted">

        <div class="container">

            &copy; 2023 - Dung Shop - <a asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>

        </div>

    </footer>

    <script src="~/lib/jquery/dist/jquery.min.js"></script>

    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>

    <script src="~/js/site.js" asp-append-version="true"></script>

</body>

</html>

**Display Feature Product:**

* 1. Thêm CSS để kiểm soát việc ngắt dòng của tên sản phẩm: Thêm CSS vào file site.css hoặc một file CSS tùy chỉnh khác để kiểm soát việc ngắt dòng của tên sản phẩm.

file:///c%3A/Dung/C%23.NET/Final\_Assignment\_Dung/UI\_Final/wwwroot/css/site.css

.product-title {

  font-size: 12.25px; /\* Đặt kích thước phông chữ \*/

  display: -webkit-box;

  -webkit-line-clamp: 2; /\* Hiển thị tối đa 2 dòng \*/

  line-clamp: 2; /\* Standard property for compatibility \*/

  -webkit-box-orient: vertical;

  overflow: hidden;

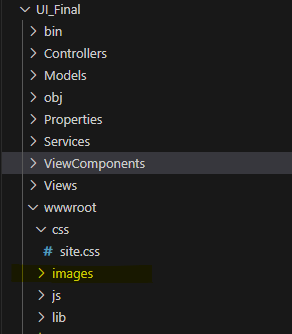
  text-overflow: ellipsis;

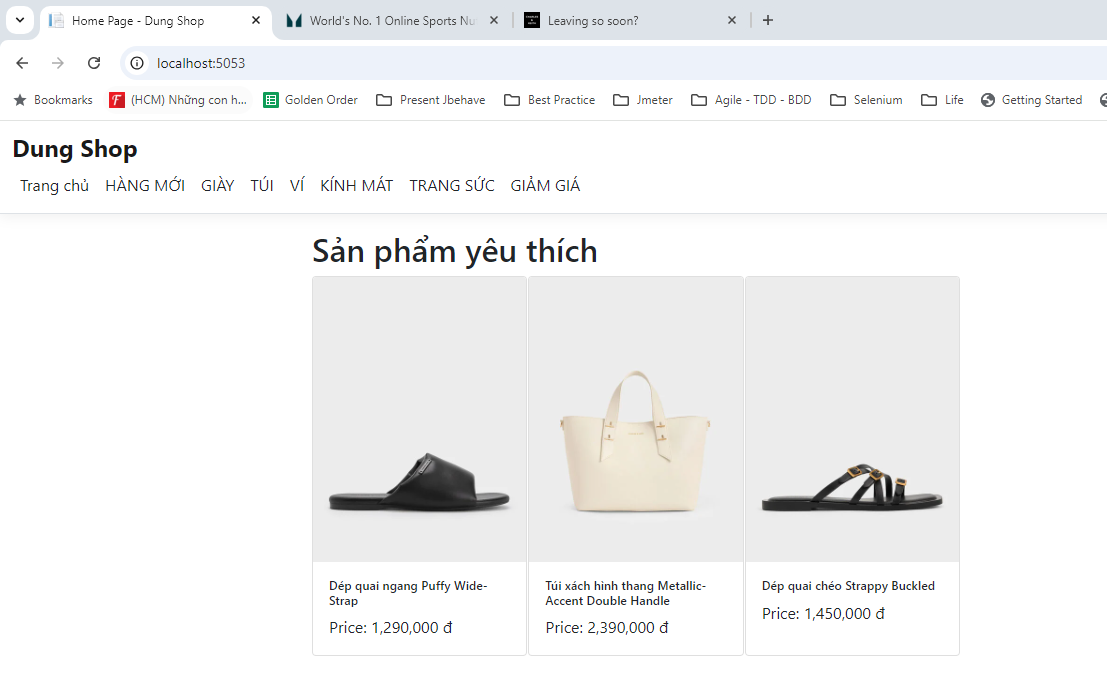
  white-space: normal;

}

* 1. Cập nhật file file:///c%3A/Dung/C%23.NET/Final\_Assignment\_Dung/UI\_Final/Views/Home/Index.cshtml để thêm lớp CSS cho tên sản phẩm. hiện products lên Home page
* @model UI\_Final.Models.HomeViewModel
* @{
* ViewData["Title"] = "Home Page";
* }
* <div class="col-md-6">
* <h2>Sản phẩm yêu thích</h2>
* <div class="d-flex flex-nowrap" style="gap: 1px; overflow-x: auto;"> <!-- Sử dụng Flexbox để hiển thị sản phẩm trên một dòng và thêm khoảng cách -->
* @foreach (var product in Model.FeaturedProducts)
* {
* <div class="card" style="width: 450px; height: 380px; margin-bottom: 1px;"> <!-- Điều chỉnh kích thước khu vực chứa sản phẩm -->
* <img src="@product.ImageUrl" class="card-img-top" alt="@product.Name ?? "No Image"" style="width: 100%; height: auto;" /> <!-- Điều chỉnh kích thước hình ảnh -->
* <div class="card-body">
* <h5 class="card-title product-title">@product.Name</h5> <!-- Thêm lớp CSS cho tên sản phẩm -->
* <p class="card-text">Price: @product.Price.ToString("N0") @(product.PriceCurrency == "VND" ? "đ" : product.PriceCurrency)</p> <!-- Hiển thị giá sản phẩm cùng với đơn vị tiền tệ -->
* </div>
* </div>
* }
* </div>
* </div>

**Chứa images**

****

****