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

**Step 1: VS Code -> create 2 projects**

Project 1: API\_Final -> xử lý BE, API

Project 2: UI\_Final

**Step2: 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();

**Step 4: 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**
8. **1. Add Entity Framework Core and SQL Server Packages: add API\_Final project & Domain project**

**cd ../Domain**

**dotnet add package Microsoft.EntityFrameworkCore**

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

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

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

2. 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" \o ") 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);

}

}

}