

DEV TEAM

Senior & Junior developer working together

PROMPT COMPLEXITY



SENIOR
DEVELOPER

These prompts are designed for experienced developers who have a strong understanding of programming concepts, languages, and tools. The tasks in these prompts are more complex and may involve advanced features of a language or framework. They require a higher level of problem-solving skills and a deep understanding of software development principles. These prompts can include building complex systems, optimizing code, integrating multiple technologies, and more.



JUNIOR
DEVELOPER

These prompts are designed for beginners or those with less experience in programming. They are less complex and focus on basic programming concepts, introducing new developers to the fundamentals of a language or framework. These prompts can involve tasks like writing simple functions, understanding and fixing bugs, or learning to use basic features of a language or tool. They provide a supportive environment for learning and growth, helping new developers to build their skills and confidence.

TARGET APP – USER STORY

Our system works like a virtual store manager. It keeps a list of all your products, just like a catalog. You can add new products to this catalog, just like placing a new item on a store shelf. When you add a new product, you can include all the information a customer might want to know - like what the product is called, what it looks like, how much it costs, what category it falls under, and more.

If you need to update information about a product - maybe the price has changed, or you've got a new photo to show - you can do that, too. It's like updating the product's tag on the store shelf. And if a product is no longer available, you can remove it from the catalog, just like taking it off the shelf.

The system also helps you keep track of which products are in stock. If a product is running low or is out of stock, the system can let you know so you can restock it.

The system can also handle customer reviews and ratings for your products. It's like having a bulletin board in your store where customers can leave their feedback.

The best part is that our system is designed to grow with the business. As you add more types of products or need to include more information about your products, the system can handle it.

EXAMPLE 1:

.NET Application in Visual Studio 2022

TARGET APPLICATION – TECH STACK SELECTED

- Visual Studio 2022 (VS 2022): This is a powerful Integrated Development Environment (IDE) by Microsoft that supports various programming languages. It is used to write, debug, and test code. It provides various tools and features to facilitate efficient and effective software development.
- .NET 8: This is a free, open-source, cross-platform framework for building modern applications. It provides tools to build web, cloud, IoT, and desktop applications. For your web API, .NET 8 offers ASP.NET, a robust framework for building web apps and services, HTTP services, and more.
- Entity Framework Core: This is a lightweight, extensible, open-source, and cross-platform version of the popular Entity Framework data access technology. It is an Object-Relational Mapping (ORM) framework that simplifies data access by letting developers work with relational data using domain-specific objects, eliminating the need for most data access plumbing code.
- NUnit: This is a unit-testing framework for all .NET languages. It is used to write and execute tests to ensure that individual units of source code (like functions, methods, classes, etc.) are working correctly. This helps catch any bugs or errors early in the development cycle, making the code more reliable and robust.

CREATE PROJECT

Create a new project

Search for templates (Alt+S) Clear all

C# Windows Web

Recent project templates

- ASP.NET Core Web App (Model-View-Controller)
- NUnit Test Project
- Azure Functions
- ASP.NET Core Web API**
- Class Library
- Console App
- Blazor Server App
- Blank Solution
- Console App (.NET Framework)
- Class Library (.NET)

Blazor Web App

A project template for creating a Blazor web app that supports both server-side rendering and client interactivity. This template can be used for web apps with rich dynamic user interfaces (UIs).

C# Linux macOS Windows Blazor Cloud Web

ASP.NET Core Web App (Razor Pages)

A project template for creating an ASP.NET Core application with example ASP.NET Core Razor Pages content

C# Linux macOS Windows Cloud Service Web

ASP.NET Core Web API

A project template for creating a RESTful Web API using ASP.NET Core controllers or minimal APIs, with optional support for OpenAPI and authentication.

C# Linux macOS Windows API Cloud Service Web Web API

ASP.NET Core Web API (native AOT)

A project template for creating a RESTful Web API using ASP.NET Core minimal APIs published as native AOT.

C# Linux macOS Windows API Cloud Service Web

Next

CONFIGURE

Configure your new project

ASP.NET Core Web API C# Linux macOS Windows API Cloud Service Web Web API

Project name
WebApi

Location
C:\Demos\

Solution
Create new solution

Solution name ⓘ
WebApi

Place solution and project in the same directory

Project will be created in "C:\Demos\WebApi\WebApi"

Back Next

CONFIGURE

Additional information

ASP.NET Core Web API C# Linux macOS Windows API Cloud Service Web Web API

Framework ⓘ
↓ .NET 8.0 (Long Term Support) ↓

Authentication type ⓘ
↓ None ↓

Configure for HTTPS ⓘ

Enable Docker ⓘ

Docker OS ⓘ
↓ Linux ↓

Enable OpenAPI support ⓘ

Do not use top-level statements ⓘ

Use controllers ⓘ

Back Create

START THE PROMPTING – PROMPT 1




SENIOR
DEVELOPER


I need to create a database containing information about products. I want to use Entity Framework Core.

Can you remind me what are the steps and best practices to do it?

I need to create a database containing information about products. I want to use Entity Framework Core. Can you remind me what are the steps and best practices to do it?

 GitHub Copilot

RESULT

 GitHub Copilot

Sure, here are the steps and best practices to create a database using Entity Framework Core:

1. **Install Entity Framework Core:** You can install it via NuGet package manager. The main package you need is [Microsoft.EntityFrameworkCore](#). If you're using SQL Server, you'll also need [Microsoft.EntityFrameworkCore.SqlServer](#).
2. **Create Your Model:** This is a class that represents the data you want to store. For example, a [Product](#) class might have properties like [Id](#), [Name](#), [Description](#), and [Price](#).

```
public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Description { get; set; }
    public decimal Price { get; set; }
}
```



Insert in new file

Preview




1

RESULT - CONTINUED

3. **Create Your DbContext:** This is a class that manages the interaction between your models and the database. It includes `DbSet` properties for each type of entity you're working with.

```
public class ProductContext : DbContext
{
    public DbSet<Product> Products { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.UseSqlServer(@"Server=(localdb)\mssqllocaldb;Database=MyDatabase");
    }
}
```

 Insert in new file Preview

RESULT - CONTINUED

4. **Migrations:** Migrations are a way to keep your database schema in sync with your model classes. You can create a migration with the [Add-Migration](#) command in the Package Manager Console, and then apply it with the [Update-Database](#) command.
5. **CRUD Operations:** You can now perform Create, Read, Update, and Delete operations on your [Product](#) entities using the [ProductContext](#).

Here are some best practices:

- **Separation of Concerns:** Keep your [DbContext](#) and domain models in a separate project or at least separate folders.
- **Avoid Lazy Loading:** Lazy loading can lead to performance issues. Use eager loading with [Include](#) method or explicit loading instead.
- **Use Migrations for Schema Changes:** Always use migrations to apply schema changes to your database.
- **Don't Share DbContext Instances:** DbContext is not thread-safe. Always create a new instance when you need one.
- **Use Dependency Injection:** This allows you to manage DbContext lifetime and configuration in a centralized place.


PROMPT 2 - INSTALL REFERENCES



JUNIOR
DEVELOPER

What are commands for Nuget package installation for these two packages? I'm using package manager console.

what are commands for Nuget package installation for these two packages? i'm using package manager console.

 GitHub Copilot

You can install the required packages using the following commands in the Package Manager Console:

For [Microsoft.EntityFrameworkCore](#):

```
Install-Package Microsoft.EntityFrameworkCore
```



Preview

For [Microsoft.EntityFrameworkCore.SqlServer](#):

```
Install-Package Microsoft.EntityFrameworkCore.SqlServer
```



Preview

These commands will install the latest stable versions of the packages. If you need a specific version, you can specify it using the [-Version](#) parameter, like so:

```
Install-Package Microsoft.EntityFrameworkCore -Version 5.0.0
```



Preview

PROMPT 3 – FIX CODE ERRORS WITH **/FIX** COMMAND



JUNIOR
DEVELOPER

/fix

GitHub Copilot

The errors you're seeing are because the necessary namespaces for `DbContext`, `DbSet<>`, and `DbContextOptionsBuilder` are not included in your file. These classes are part of the `Microsoft.EntityFrameworkCore` namespace.

You can fix these errors by adding the following line at the top of your file:

```
using Microsoft.EntityFrameworkCore;
```

Preview

So your updated `ProductContext.cs` file should look like this:

```
using Microsoft.EntityFrameworkCore;

public class ProductContext : DbContext
{
    public DbSet<Product> Products { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.UseSqlServer(@"Server=(localdb)\mssqllocaldb;Database=...");
    }
}
```


PROMPT 5



SENIOR
DEVELOPER

For my local development I don't want to use actual sql server for db, rather I would like to use In memory database. Can you update my Program.cs class. Only use this in memory database for development environment. For other environments use sql server.

For my local development i don't want to use actual sql server for db, rather i would like to use In memory database. Can you update my Program.cs class. Only use this in memory database for development environment. For other environments use sql server.

 GitHub Copilot

Sure, you can use the `IsDevelopment` method to check if the current environment is development. If it is, you can add the in-memory database service; otherwise, add the SQL Server service. Here's how you can do it:

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();


if (builder.Environment.IsDevelopment())
{
    builder.Services.AddDbContext<ProductContext>(options =>
        options.UseInMemoryDatabase("MyDatabase"));
}
else
{
    builder.Services.AddDbContext<ProductContext>(options =>
        options.UseSqlServer(builder.Configuration.GetConnectionString("MyDataba
    }
}
```

USING /FIX COMMAND

/fix - Propose a fix for problems in the selected code

Helpful Resources

- [About GitHub Copilot Chat](#)
- [Getting Started with GitHub Copilot Chat](#)
- [Video Tutorial - GitHub Copilot Chat](#)
- [Code Refinement and Debugging with Copilot Chat](#)

 References



AI suggestions might be inaccurate.

fix

/


#



PROMPT 6- FIX ERRORS




JUNIOR
DEVELOPER

 GitHub Copilot

The error message is indicating that the `UseInMemoryDatabase` extension method is not available. This method is part of the `Microsoft.EntityFrameworkCore.InMemory` namespace, which is provided by the `Microsoft.EntityFrameworkCore.InMemory` NuGet package.


First, you need to install the `Microsoft.EntityFrameworkCore.InMemory` package. You can do this via the Package Manager Console with the following command:

```
Install-Package Microsoft.EntityFrameworkCore.InMemory
```

 Preview

Then, you need to add the following `using` directive at the top of your `Program.cs` file:

```
using Microsoft.EntityFrameworkCore;
```

 Preview

So, your updated `Program.cs` file should look like this:

```
using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
```

/fix

PROMPT 7




SENIOR
DEVELOPER

When developing local and using InMemory database i need some mock objects for my products. Can you, according to the best practice, generate me initialization logic for my local development. In memory database should contain 20 mocked product entities. #file:'Program.cs' #file:'ProductContext.cs'

PROMPT 7 - RESULT

When developing local and using InMemory database i need some mock objects for my products. Can you, according to the best practice, generate me initialisation logic for my local development. In memory database should contain 20 mocked product entities. `C# Program.cs` `C# ProductContext.cs`



 GitHub Copilot

Sure, you can seed your in-memory database with mock data by adding a database initializer. This initializer can check if the environment is development and if so, add the mock data. Here's how you can do it:

First, modify your `ProductContext` class to take `DbContextOptions` in its constructor:

```
public class ProductContext : DbContext
{
    public ProductContext(DbContextOptions<ProductContext> options) : base(options) { }

    public DbSet<Product> Products { get; set; }
}
```

 Insert in new file 

Then, in your `Program.cs`, after the `app.Run()` line, add the following code:

```
if (app.Environment.IsDevelopment())
{
    using (var scope = app.Services.CreateScope())
    {
        var context = scope.ServiceProvider.GetRequiredService<ProductContext>();
        if (!context.Products.Any())
        {
            for (int i = 1; i <= 20; i++)
            {

```

AI suggestions might be inaccurate.


PROMPT 8 – REPOSITORY PATTERN



SENIOR
DEVELOPER

In the future I would like to be able to easily change my db connection implementation. Can you generate repository pattern implementation around my dbContext so that I can access product list over interface, rather than directly.

In the future i would like to be able to easily change my db connection implementation. Can you generate repository pattern implementation around my dbContext so that i can access product list over interface, rather than directly.

 GitHub Copilot

Sure, you can use the Repository pattern to abstract away the details of accessing data from your `DbContext`. Here's how you can do it:

First, create an interface for your repository:

```
public interface IProductRepository
{
    Task<IEnumerable<Product>> GetProductsAsync();
    // Add other methods as needed, like GetById, Add, Update, Delete, etc.
}
```



Insert in new file

Preview

Then, create a class that implements this interface:

```
public class ProductRepository : IProductRepository
{
    private readonly ProductContext _context;

    public ProductRepository(ProductContext context)
    {
        _context = context;
    }

    public async Task<IEnumerable<Product>> GetProductsAsync()
    {
        return await _context.Products.ToListAsync();
    }

    // Implement other methods as needed
}
```



Insert in new file

Preview

In your [Program.cs](#), register your repository with the DI container:

```
builder.Services.AddScoped<IProductRepository, ProductRepository>();
```



Preview

Now, you can inject [IProductRepository](#) into your controllers or other services, and use it to access your products. This way, the rest of your application doesn't need to know about [DbContext](#) or Entity Framework, which makes it easier to change your database connection in the future.

PROMPT 9 – CONTROLLER + /FIX




SENIOR
DEVELOPER

Can you generate a controller code for my Products. I need usual CRUD methods implemented, and calling my IProductRepository object. Concrete instance would be provided via constructor injection. Include proper using statements and namespace used in application.

/fix with references to ProductsController, IProductRepository and ProductRepository

Can you generate a controller code for my Products. I need usual CRUD methods implemented, and calling my IProductRepository object. Concrete instance would be provided via constructor injection.

 GitHub Copilot

Sure, here's a basic implementation of a `ProductsController` that uses `IProductRepository`:

```
[ApiController]
[Route("[controller]")]
public class ProductsController : ControllerBase
{
    private readonly IProductRepository _productRepository;

    public ProductsController(IProductRepository productRepository)
    {
        _productRepository = productRepository;
    }

    [HttpGet]
    public async Task<IEnumerable<Product>> Get()
    {
        return await _productRepository.GetProductsAsync();
    }

    [HttpGet("{id}")]
    public async Task<ActionResult<Product>> Get(int id)
    {
```

```
/fix C# ProductsController.cs C# IRepository.cs C# ProductRepository.cs
```

GitHub Copilot

The errors you're seeing are because the `IProductRepository` interface and the `ProductRepository` class do not have the methods that are being called in the `ProductsController`.

Let's add the missing methods to the `IProductRepository` interface and implement them in the `ProductRepository` class.

Here's the updated `IProductRepository` interface:

```
public interface IProductRepository
{
    Task<IEnumerable<Product>> GetProductsAsync();
    Task<Product> GetProductByIdAsync(int id);
    Task AddProductAsync(Product product);
    Task UpdateProductAsync(Product product);
    Task DeleteProductAsync(Product product);
}
```



Insert in new file

Preview

And here's the updated `ProductRepository` class:

```
public class ProductRepository : IProductRepository
{
    private readonly ProductContext _context;

    public ProductRepository(ProductContext context)
    {
        _context = context;
    }

    public async Task<IEnumerable<Product>> GetProductsAsync()
    {
        return await _context.Products.ToListAsync();
    }

    public async Task<Product> GetProductByIdAsync(int id)
    {
        return await _context.Products.FindAsync(id);
    }
}
```

For the nullability error, you can change the return type of the `Get` method to `Task<ActionResult<Product>>` to match the nullable return type of `GetProductByIdAsync`. Here's the updated method:

```
[HttpGet("{id}")]
public async Task<ActionResult<Product>> Get(int id)
{
    var product = await _productRepository.GetProductByIdAsync(id);
    if (product == null)
    {
        return NotFound();
    }
    return product;
}
```



Preview


PROMPT 10 – FIX PRODUCTS NOT SHOWING



JUNIOR
DEVELOPER

Can you analyze why my mocked products from #file:'Program.cs' lines 40 to 60 are not showing?

Also break point on line 40 (if (app.Environment.IsDevelopment())) is never reached?

 GitHub Copilot

The issue is due to the placement of your code block for seeding the in-memory database. The `app.Run();` method starts the web application and blocks the calling thread until the host shuts down. This means that any code after `app.Run();` will not be executed until the application is shutting down.

To fix this, you should move your database seeding code to a place before `app.Run();`. Here's how you can do it:

```
var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();

    using (var scope = app.Services.CreateScope())
```

PROMPT 11 – CUSTOMER REQUEST




JUNIOR
DEVELOPER

Can you update #Product.cs with info from customer request:

"Here are some specifications regarding the properties I would like to input for my products and the types of products I want to sell: Properties for Products: Product Name: A clear and descriptive name that instantly identifies what the product is. Product Description: Detailed information about the product, including its features, usage, benefits, and any other relevant information. Product Category: The category or department the product belongs to (e.g., Electronics, Home and Kitchen, Apparel, etc.). Product Sub-category: A more specific classification within the main category (e.g., under Electronics, there could be Mobiles, Laptops, etc.). Product Price: The price of the product. There should be fields for both regular and sale prices. Product Images: High-quality images from different angles. There should be an option to zoom and view the product in a 360-degree view. Product Videos: Short clips demonstrating the product, if applicable. SKU (Stock Keeping Unit): Unique identifier that helps to manage the inventory. Stock Status: Information about the availability of the product (In Stock, Out of Stock, Limited Stock). Product Weight and Dimensions: Important for calculating shipping costs. Product Variants: Different variations like size, color, material, etc. Product Reviews and Ratings: Allow customers to leave feedback and rate products. Related Products: Suggestions for other products that customers might be interested in. SEO Fields: For meta title, meta description, and keywords to help in search engine optimization. Types of Products: Physical Products: These are tangible items such as clothes, electronics, furniture, etc. Digital Products: These are intangible items like ebooks, music, digital art, software, etc. Services: These include services like consulting, coaching, online courses, etc. Subscription-based Products: Products that require regular renewal like magazines, memberships, etc. Bundle Products: A combination of related products sold as a package. Customizable Products: Products that can be personalized based on customer preferences. I would like the platform to be flexible, allowing for additional product types and properties as the business grows and evolves."

Also update #Program.cs initialization of mocked products.

PROMPT 11 - RESULT

 GitHub Copilot

Based on the customer's request, here's an updated version of your `Product` class:

```
public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Description { get; set; }
    public string Category { get; set; }
    public string SubCategory { get; set; }
    public decimal RegularPrice { get; set; }
    public decimal? SalePrice { get; set; }
    public string ImageUrl { get; set; }
    public string VideoUrl { get; set; }
    public string SKU { get; set; }
    public string StockStatus { get; set; }
    public decimal Weight { get; set; }
    public string Dimensions { get; set; }
    public string Variants { get; set; }
}
```

PROMPT 11 – RESULT

And here's how you can update the initialization of mocked products in `Program.cs`:

```
if (app.Environment.IsDevelopment())
{
    using (var scope = app.Services.CreateScope())
    {
        var context = scope.ServiceProvider.GetRequiredService
        if (!context.Products.Any())
        {
            for (int i = 1; i <= 20; i++)
            {
                context.Products.Add(new Product
                {
                    Id = i,
                    Name = $"Product {i}",
                    Description = $"This is product {i}",
                    Category = "Electronics",
                    SubCategory = "Mobiles",
                    RegularPrice = i * 10,
                    SalePrice = i * 9,
                    ImageUrl = $"https://example.com/images/pr
                    VideoUrl = $"https://example.com/videos/pr
                    SKU = $"SKU{i}",
                    StockStatus = "In Stock"
                });
            }
        }
    }
}
```

PROMPT 12 – ADVANCED FILTERING + /FIX



SENIOR
DEVELOPER

I want to implement filtering in my ProductsController GET method. The following examples illustrate how i want it to work and the use and semantics of each of the logical operators.

Example: all products with a name equal to 'Milk'

GET `https://api.contoso.com/products?filter=name eq 'Milk'` Example: all products with a name not equal to 'Milk'


GET `https://api.contoso.com/products?filter=name ne 'Milk'` Example: all products with the name 'Milk' that also have a price less than 2.55:

GET `https://api.contoso.com/products?filter=name eq 'Milk' and price lt 2.55` Example: all products that either have the name 'Milk' or have a price less than 2.55:

GET `https://api.contoso.com/products?filter=name eq 'Milk' or price lt 2.55` Example: all products that have the name 'Milk' or 'Eggs' and have a price less than 2.55:

GET `https://api.contoso.com/products?filter=(name eq 'Milk' or name eq 'Eggs') and price lt 2.55`


PROMPT 12 - RESULT

 GitHub Copilot

To implement the filtering functionality as described in the examples, you can use the [System.Linq.Dynamic.Core](#) library. This library allows you to build dynamic LINQ queries using string expressions, which is perfect for implementing a flexible filtering system.

First, install the [System.Linq.Dynamic.Core](#) NuGet package:

```
Install-Package System.Linq.Dynamic.Core
```


 Preview

Then, modify your `Get` method in the `ProductsController` class to accept a `filter` query parameter and apply it to the products query:

```
[HttpGet]
public async Task<ActionResult<IEnumerable<Product>>> Get([FromQuery] string filter)
{
    IQueryable<Product> query = _productRepository.GetProducts();

    if (!string.IsNullOrEmpty(filter))
    {
        query = query.Where(filter);
    }

    return await query.ToListAsync();
}
```

 Preview

PROMPT 12 – RESULT AFTER /FIX



JUNIOR
DEVELOPER

Here's the corrected `Get` method:

```
[HttpGet]
public async Task<ActionResult<IEnumerable<Product>>> Get([FromQuery] string?
{
    IQueryable<Product> query = _productRepository.GetProducts();

    if (!string.IsNullOrEmpty(filter))
    {
        query = query.AsQueryable().Where(filter);
    }

    return await query.ToListAsync();
}
```



Preview



PROMPT 13 – ADD DOCKER FILE




SENIOR
DEVELOPER

Given the fact I'm using .NET 8, and `dockerfile`, generate me a Dockerfile which will create an image out of my application. Use multi staged build and the smallest, most optimal final base image. Use Linux images.

will .NET 8 app work with this docker file?

PROMPT 13 - RESULT

 GitHub Copilot

The Dockerfile provided in the previous response is for a .NET 6 application. If you're using .NET 8, you'll need to update the base images to the .NET 8 versions once they're available. As of now, .NET 8 is not yet released, so there are no official .NET 8 Docker images.

Here's an example of what the Dockerfile might look like for .NET 8:

```
# Stage 1: Build the application
FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build
WORKDIR /source

# Copy csproj and restore
COPY *.csproj .
RUN dotnet restore

# Copy everything else and build the app
COPY . .
RUN dotnet publish -c Release -o /app --no-restore

# Stage 2: Create the final image
FROM mcr.microsoft.com/dotnet/aspnet:8.0
WORKDIR /app
COPY --from=build /app ./
ENTRYPOINT ["dotnet", "WebApi.dll"]
```



Preview

PROMPT 14 – AZURE DEVOPS PIPELINES



SENIOR
DEVELOPER

I need to setup CI-CD pipeline in my Azure DevOps pipelines. Generate YAML azure devops pipeline which will build the code, execute unit tests, build an image and push it to container registry.

GitHub Copilot

Here is a basic example of an Azure Pipelines YAML file that builds a .NET application, runs unit tests, builds a Docker image, and pushes it to an Azure Container Registry:

```
trigger:
- main

variables:
  dockerRegistryServiceConnection: 'DockerRegistryServiceConnection'
  imageRepository: 'your-image-repository'
  containerRegistry: 'your-container-registry.azurecr.io'
  dockerfilePath: '$(Build.SourcesDirectory)/Dockerfile'
  tag: '$(Build.BuildId)'

stages:
- stage: Build
  displayName: Build and test
  jobs:
  - job: Build
    displayName: 'Build'
    pool:
      vmImage: 'ubuntu-latest'
    steps:
    - task: DotNetCoreCLI@2
      inputs:
```

PROMPT 15 – ADDING TO PIPELINE



SENIOR
DEVELOPER

Also i want to be able to use parameters to turn push of the images on/off when I manually run the pipeline

```
You can add parameters to your Azure Pipelines YAML file to control whether to push images to a registry. To add a pushImage parameter to your pipeline, add the following to your pipeline's trigger and parameters sections:
```

```
trigger:
- main

parameters:
- name: pushImage
  displayName: Push Docker Image
  type: boolean
  default: true
```

```
tags: $(tag)
- ${{ if eq(parameters.pushImage, true) }}:
- task: Docker@2
  inputs:
    containerRegistry: $(dockerRegistryServiceConnection)
    repository: $(imageRepository)
    command: 'push'
    tags: $(tag)
```


PROMPT 16 – UNIT TEST PROJECT



JUNIOR
DEVELOPER

How to create unit test project in my Visual Studio solution?

How to create unit test project in my Visual Studio solution?

 GitHub Copilot

To create a unit test project in your Visual Studio solution, follow these steps:

1. Open your solution in Visual Studio.
2. Right-click on the Solution in the Solution Explorer and select [Add -> New Project](#).
3. In the [Create a new project](#) dialog, search for [xUnit Test Project \(.NET Core\)](#) or [MSTest Test Project \(.NET Core\)](#) depending on your preference. Select the template and click [Next](#).
4. In the [Configure your new project](#) dialog, enter a name for your test project. Typically, it's the name of the project being tested followed by [.Tests](#). Click [Create](#).
5. Once the test project is created, you need to add a reference to the project you want to test. Right-click on your test project in the Solution Explorer, select [Add -> Reference](#).
6. In the [Reference Manager](#) dialog, under [Projects](#), select the project you want to test and click [OK](#).

Now, you have a new unit test project in your solution, and it's ready for you to add tests. You can create a new test by right-clicking on your test project in the Solution Explorer, selecting [Add -> New Item](#), choosing [xUnit Test](#) or [MSTest Test](#), and clicking [Add](#).

- ❖ [How can I run the unit tests from Visual Studio?](#)
- ❖ [Can I generate a code coverage report for my unit tests in Visual Studio?](#)


PROMPT 17 – ADDING UNIT TESTS



JUNIOR
DEVELOPER

In my unit test file #file:'UnitTest1.cs' generate an initial unit test for #file:'ProductsController.cs' Include in instructions all nuget packages and proper namespaces

In my unit test file `C# UnitTest1.cs` generate an initial unit test for `C# ProductsController.cs`

 GitHub Copilot

To create an initial unit test for the `ProductsController`, you'll need to mock the `IProductRepository` dependency. Here's an example of how you can do this using Moq and NUnit:

```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Moq;
using NUnit.Framework;
using WebApi.Controllers;
using WebApi.Models;

namespace WebApi.Tests
{
    public class Tests
    {
        private Mock<IProductRepository> _mockRepo;
        private ProductsController _controller;

        [SetUp]
        public void Setup()
```

PROMPT 18 – ADDITIONAL UNIT TESTS



JUNIOR
DEVELOPER

Add unit test which tests filtering by price in #UnitTests1.cs

Add unit test with tests filtering by price in `C# UnitTest1.cs`

GitHub Copilot

To add a unit test that tests filtering by price, you can modify the `Get_WhenCalled_ReturnsAllItems` test to return a list of products with different prices, and then filter the products by price in the `Get` method call. Here's how you can do it:

```
[Test]
public async Task Get_WhenCalledWithPriceFilter_ReturnsFilteredItems()
{
    // Arrange
    _mockRepo.Setup(repo => repo.GetProducts()).Returns(new List<Product>
    {
        new Product { Id = 1, Name = "Product1", RegularPrice = 100 },
        new Product { Id = 2, Name = "Product2", RegularPrice = 200 },
        new Product { Id = 3, Name = "Product3", RegularPrice = 300 }
    }.AsQueryable());

    // Act
    var result = await _controller.Get("RegularPrice < 250");

    // Assert
    Assert.That(result.Value, Is.TypeOf<List<Product>>());
    Assert.That(result.Value.Count, Is.EqualTo(2));
    Assert.That(result.Value.All(p => p.RegularPrice < 250));
}
```


PROMPT 19 -21 – UNIT TESTS



JUNIOR
DEVELOPER

Add unit test which tests creation of a new product in
`#file:'UnitTest1.cs'`

Add unit test which tests deletion of a product in `#file:'UnitTest1.cs'`

Add unit test which tests update of an existing product in
`#file:'UnitTest1.cs'`

PROMPT 22 – AZURE INFRASTRUCTURE



SENIOR
DEVELOPER

I will host this .NET 8 app container in azure app service. Generate bicep template for azure resource creation used for hosting this app. Also provide instructions on how to execute the bicep template against azure in markdown format in a README.md file.

I will host this .NET 8 app container in azure app service. Generate bicep template for azure resource creation used for hosting this app. Also provide instructions on how to execute the bicep template against azure in markdown format in a README.md file.

GitHub Copilot

Here is a basic Bicep template for creating an Azure App Service Plan and an App Service:

```
param name string
param location string = resourceGroup().location
param skuName string = 'F1'
param skuTier string = 'Free'

resource appServicePlan 'Microsoft.Web/serverfarms@2021-02-01' =
  name: name
  location: location
  sku: {
    name: skuName
    tier: skuTier
  }
}

resource appService 'Microsoft.Web/sites@2021-02-01' = {
  name: name
  location: location
  properties: {
    serverFarmId: appServicePlan.id
  }
}
```

Here is the README.md file:

```
# Azure App Service Deployment

This repository contains a Bicep template for deploying an Azure

## Prerequisites

- Azure CLI
- Bicep CLI

## Deployment Steps

1. Login to your Azure account:



```
az login
```



2. Create a resource group:



```
az group create --name --location
```



3. Deploy the Bicep template:
```

HANDS-ON EXERCISES & CHALLENGES

Let's check what we've learned!

CHALLENGE 1

- Your task is to design and implement an application that is capable of storing vital information about Users. The details to be stored include, but are not limited to, the user's First Name, Last Name, Email, and User id. You will need to approach this task by utilizing Entity Framework Core and the latest version of Visual Studio, which is Visual Studio 2022.
- In the process, you will need to create a database model, set up a database context, and configure Entity Framework in the application's startup. The application should be capable of performing CRUD (Create, Read, Update, Delete) operations on the user's data. Remember to ensure data integrity and security when handling the user's information.

CHALLENGE 2

- Your task in this challenge is to enhance your local development experience by setting up a local database and utilizing the Entity Framework (EF) Core Seed method. This task will help you simulate a real-world scenario where you would interact with data stored in a local database rather than an in-memory database. The EF Core Seed method should be used to populate your local database with initial data.

The goal here is to migrate from using an in-memory database and mocked objects to a more realistic setup featuring a local database and actual data objects generated via the EF Core Seed method. This shift will provide a more practical understanding of how your application interacts with stored data.

CHALLENGE 3

- Your mission in this challenge is to implement a Users Controller for your REST API endpoint. This controller will act as the intermediary between the User model and the corresponding views, handling incoming browser requests, retrieving necessary model data, and returning appropriate responses.

To be more specific, your Users Controller should be capable of managing all interactions with User data. This includes the implementation of Create, Read, Update, and Delete (CRUD) operations, which are fundamental in any data-driven application.

CHALLENGE 4

- Your objective in this challenge is to extend the User object with additional properties as per the customer's requirements. The customer has provided a detailed story outlining the specific properties they want to track for their users, and the types of users they anticipate interacting with.

By incorporating these properties, you will be adding a new level of depth and complexity to your User object. This detailed information will facilitate a richer, more personalized user experience, and enable more effective user management.

The properties to be added to the User object include but are not limited to: User Name, User Profile, User Category, User Sub-category, User Interaction History, User Images, User Videos, UID, Account Status, User Preferences, User Variants, User Feedback and Ratings, Connected Users, and SEO Fields.

CHALLENGE 5

- Your task for this challenge is to implement a unit test for the User Controller in your application. Unit tests are critical to ensure the reliability and functionality of your code. They help you identify and fix bugs, prevent future errors, and improve the overall design of your code.
- In this case, you're required to design a unit test that verifies the functionality of the User Controller. This includes creating tests for all of the CRUD operations (Create, Read, Update, Delete) that the User Controller performs. Ensure that the tests cover all possible edge cases and unexpected user inputs.

CHALLENGE 6

- The task for this challenge is to create a Docker image for your application. Docker is a platform that allows developers to package applications into containers — standardized executable components that combine application source code with all the operating system (OS) libraries and dependencies required to run the code in any environment.

Creating a Docker image for your application ensures that it can run consistently on any machine that has Docker installed, regardless of the underlying operating system. This is crucial for simplifying deployment and scaling of your application.

APPLICATION IN VISUAL STUDIO CODE

TECH STACK OVERVIEW

- Visual Studio Code: a popular and versatile code editor with a wide range of extensions for various languages and frameworks. VS Code's rich ecosystem of extensions allows developers to customize their environment for various languages and runtimes (e.g., Python, Java, C++).
- Flask: a lightweight Python web framework for building web applications quickly and easily. Flask provides sensible defaults and conventions for getting started quickly, allowing developers to choose their tools and libraries.
- pytest-flask is an extension for the pytest test runner specifically designed for testing Flask applications. It simplifies testing and development by providing useful tools. Features include fixtures for creating Flask apps, test clients, and CLI runners. The extension integrates seamlessly with Flask, allowing developers to write effective tests using the pytest framework.
- Materialize is a responsive front-end framework based on Google's Material Design guidelines. It combines the principles of Material Design with the flexibility of CSS to create a visually attractive and easy-to-use framework.


PROMPT 1



JUNIOR
DEVELOPER

I need to implement a simple web application using python programming language version 3.11.4. First tell me the steps i need to do to use python virtual environment for development. Include what commands are needed to be executed for package instalation etc. I would like Root directory of my app would be C:\Demos\PyWebApp

PROMPT 1 - RESULT

 GitHub Copilot

Here are the steps to set up a Python virtual environment for your web application:

1. Open the terminal in Visual Studio Code.
2. Navigate to your root directory using the `cd` command:

```
cd C:\Demos\PyWebApp
```

3. Create a new virtual environment using the `venv` module in Python:

```
python -m venv env
```

4. Activate the virtual environment. On Windows, the command is:

```
.\env\Scripts\activate
```

5. Now that the virtual environment is activated, you can install packages using `pip`. For example, to install Flask (a lightweight web application framework), you would use:

```
pip install flask
```

6. You can verify the installed packages with:

```
pip freeze
```


PROMPT 2



JUNIOR
DEVELOPER

Now i need to initialize this folder as local git repository so i can track changes against my code base. Can you give me step by step instructions on how to do it using VS Code?

PROMPT 2 - RESULT

 GitHub Copilot

Sure, here are the steps to initialize your folder as a local Git repository using Visual Studio Code:

1. Open the terminal in Visual Studio Code.
2. Navigate to your root directory using the `cd` command:

```
cd C:\Demos\PyWebApp
```

3. Initialize a new Git repository with the `git init` command:

```
git init
```

4. Now, your folder is a Git repository. You can start tracking


PROMPT 3



JUNIOR
DEVELOPER

I need simple help world page with simple menu for navigation. Can you recommend me which libraries and frameworks to use and give me detailed step by step instructions on how to setup and run the project in Visual Studio Code.

PROMPT 3 RESULT-

 GitHub Copilot

For a simple web application in Python, Flask is a great choice. It's a lightweight and easy-to-use framework that's perfect for small projects. Here's how you can set up a simple "Hello World" page with a navigation menu:

1. If you haven't already, install Flask in your virtual environment:

```
pip install flask
```

2. Create a new file in your project directory named `app.py` and open it in VS Code.
3. In `app.py`, import Flask and create a new Flask web server from the Flask module:

```
from flask import Flask  
app = Flask(__name__)
```

4. Define a route for the home page ("/) of your web application:

```
@app.route('/')  
def home():  
    return "Hello, World!"
```

5. Now, you can run your application. In the terminal, set the `FLASK_APP` environment variable to your application:

```
set FLASK_APP=app.py
```

6. Then, run the application with the `flask run` command:

```
flask run
```

7. Open a web browser and go to `http://127.0.0.1:5000/` to see your application.


PROMPT 4



JUNIOR
DEVELOPER

I've noticed a large number of git changes originating from .env folder of my python virtual environment. Can you generate a .gitignore file so that those changes are excluded from source control?

PROMPT 4 – RESULT

 GitHub Copilot

Used 1 reference >

Sure, you can create a `.gitignore` file in the root directory of your project to ignore the `env` folder. Here's how you can do it:

1. In the root directory of your project, create a new file named `.gitignore`.
2. Open the `.gitignore` file in VS Code.
3. Add the following lines to the `.gitignore` file:

```
# Python virtual environment
env/
```

4. Save and close the `.gitignore` file.

Now, Git will ignore the `env` folder when tracking changes. You can add more directories or files to the `.gitignore` file as needed. Each new line is a new rule. For example, to ignore all

PROMPT 5




SENIOR
DEVELOPER

I need to implement a display table for list of products. I've received this from my C# developers on how the object is going to look like. Can you use this C# class as a reference and create a mock json file which will be served in application, so i can use it as a fake endpoint for future integration? Here's C# Product class:

PROMPT 5 - CONTINUED

```
public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Description { get; set; }
    public string Category { get; set; }
    public string SubCategory { get; set; }
    public decimal RegularPrice { get; set; }
    public decimal? SalePrice { get; set; }
    public string ImageUrl { get; set; }
    public string VideoUrl { get; set; }
    public string SKU { get; set; }
    public string StockStatus { get; set; }
    public decimal Weight { get; set; }
    public string Dimensions { get; set; }
    public string Variants { get; set; }
    public string Reviews { get; set; }
    public string Ratings { get; set; }
    public string RelatedProducts { get; set; }
    public string MetaTitle { get; set; }
    public string MetaDescription { get; set; }
    public string Keywords { get; set; }
    public string ProductType { get; set; }
}
```

PROMPT 5 - RESULT

 GitHub Copilot

Used 1 reference >

Sure, based on the C# class you provided, you can create a mock JSON file that represents a list of products. Here's an example of how you can structure the JSON file:

```
[
  {
    "Id": 1,
    "Name": "Product 1",
    "Description": "This is product 1",
    "Category": "Category1",
    "SubCategory": "SubCategory1",
    "RegularPrice": 100.0,
    "SalePrice": 80.0,
    "ImageUrl": "http://example.com"
```

You can save this JSON in a file named `products.json` in your project directory. Then, you can serve this file as a static file in your Flask application. Here's how you can do it:

1. In `app.py`, import `send_from_directory` from Flask:

```
from flask import Flask, send_from_directory
```

2. Define a route for the products endpoint that sends the `products.json` file:

```
@app.route('/products')
def products():
    return send_from_directory('products.json')
```

Now, when you go to `http://127.0.0.1:5000/products` in your


PROMPT 6



JUNIOR
DEVELOPER

When I run application I get the following error: "TypeError: send_from_directory() missing 1 required positional argument: 'path' "

PROMPT 6 RESULT

 GitHub Copilot

Used 1 reference >

The `send_from_directory` function requires two arguments: the directory name and the filename. In your case, it seems like you're trying to serve the `products.json` file from the same directory as your `app.py` file. Here's how you can fix the error:

Replace the `/products` route in your `app.py` file with the following code:

```
@app.route('/products')
def products():
    return send_from_directory('.', 'products.json')
```

In this code, `'.'` represents the current directory (i.e., the directory where `app.py` is located), and `'products.json'` is the name of the file you want to serve.

PROMPT 7



SENIOR
DEVELOPER

Generate a method which will connect to url and fetch the list of products. For initial url value take <http://localhost:5000/products>. Also convert json body to a python list of products and generate python class for product. Here is JSON example of the products:

PROMPT 7 - RESULT

GitHub Copilot

Used 1 reference >

You can use the `requests` library in Python to fetch the list of products from the URL. Then, you can use the `json` library to convert the JSON response to a Python list of dictionaries. Here's how you can do it:

First, let's define the `Product` class:

```
class Product:
    def __init__(self, Id, Name, Description, Category, Sub
        self.Id = Id
        self.Name = Name
        self.Description = Description
        self.Category = Category
        self.SubCategory = SubCategory
        self.RegularPrice = RegularPrice
        self.SalePrice = SalePrice
```

```
import requests

def fetch_products(url):
    response = requests.get(url)
    product_dicts = response.json()

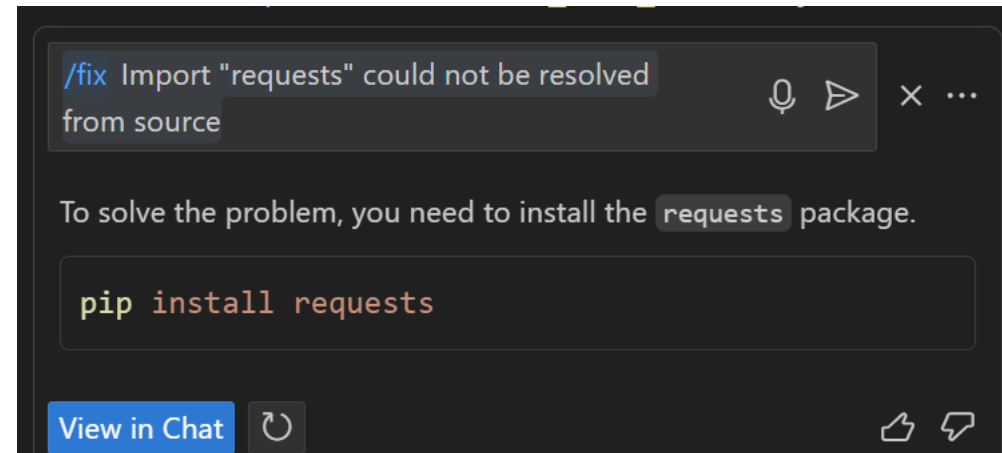
    products = []
    for product_dict in product_dicts:
        product = Product(**product_dict)
        products.append(product)

    return products
```

You can call this function with the URL of the products endpoint to get a list of `Product` objects:

```
products = fetch_products('http://localhost:5000/products')
```

PROMPT 8 – FIX ERRORS



PROMPT 9



SENIOR
DEVELOPER

On my home page, I need to display a table of products. By using a list of products fetched by method „fetch_products“ to generate html table containing Id, Name, Description and RegularPrice columns. Can you add the code directly in my app.py?

PROMPT 9 - RESULT

GitHub Copilot

Used 1 reference >

Sure, you can use Flask's built-in templating engine, Jinja2, to generate the HTML table. First, you need to import `render_template` from Flask and modify the `/` route to render a template. Then, you need to create a new template file in the `templates` directory.

Here's how you can modify your `app.py` file:

```
from flask import Flask, render_template
import requests

app = Flask(__name__)
```

```
def fetch_products(url):
    response = requests.get(url)
    product_dicts = response.json()

    products = []
    for product_dict in product_dicts:
        product = Product(**product_dict)
        products.append(product)

    return products

@app.route('/')
def home():
    products = fetch_products('http://localhost:5000')
    return render_template('home.html', products=products)
```

```
<table>
  <tr>
    <th>Id</th>
    <th>Name</th>
    <th>Description</th>
    <th>RegularPrice</th>
  </tr>
  {% for product in products %}
  <tr>
    <td>{{ product.Id }}</td>
    <td>{{ product.Name }}</td>
    <td>{{ product.Description }}</td>
    <td>{{ product.RegularPrice }}</td>
  </tr>
  {% endfor %}
</table>
```


PROMPT 10



JUNIOR
DEVELOPER

I need to display details for each of the products in the list. Can you generate additional page „/details“ which will receive product id from query string.

PROMPT 10 - RESULT

 GitHub Copilot

Used 1 reference >

Sure, you can create a new route `/details` that takes a product ID from the query string and displays the details of the corresponding product. Here's how you can modify your `app.py` file:

```
from flask import Flask, render_template, request
import requests

# ... rest of your code ...

@app.route('/details')
def details():
    product_id = request.args.get('id')
    products = fetch_products('http://
    product = next((product for product
    return render_template('details.ht
```

In this code, `request.args.get('id')` gets the product ID from the query string. The `next` function is used to find the product with the given ID in the `products` list. If no such product is found, `next` returns `None`.

Then, create a new file named `details.html` in the `templates` directory with the following content:

```
1>{{ product.Name }}</h1>
>{{ product.Description }}</p>
>Regular Price: {{ product.RegularPrice }}</p>
-- Add more fields as needed -->
```

In this template, `{{ product.Name }}`, `{{ product.Description }}`, and `{{ product.RegularPrice }}` are Jinja2 expressions that output the value of the corresponding attribute of the product.

Now, you can access the details of a product by visiting `/details?id=<product_id>`, where `<product_id>` is the ID of the product.

PROMPT 11



SENIOR
DEVELOPER

I need to add some professional looking desing. Can you give me step by step instructions on how to include Materialize framework in my python app and make my home and details template look more professional.

PROMPT 11 - RESULT

GitHub Copilot

Used 1 reference >

Sure, here are the steps to include the Materialize framework in your Python Flask application and make your templates look more professional:

- 1. Include Materialize CSS and JS in your templates:** You need to include Materialize CSS and JS files in your HTML templates. You can do this by adding the following lines in the `<head>` section of your HTML files:

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.min.css">
<script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materialize.min.js">
```

- 2. Use Materialize classes in your HTML:** Materialize provides a lot of classes that you can use to style your HTML elements. For example, to make a button look more professional, you can use the `btn` class:

How can I customize the navigation bar using Materialize framework in

- 3. Customize your home template:** For your home template, you can use Materialize's grid system to display your products in a grid layout. Here's an example:

```
ss="row">
or product in products %}
class="col s12 m6 l4">
<div class="card">
  <div class="card-image">
    
  <span class="card-title">{{ product.name }}
  </div>
  <div class="card-content">
    <p>{{ product.description }}</p>
  </div>
  <div class="card-action">
    <a href="/details?id={{ product.id }}">View
  </div>
</div>
```

```
<div class="card">
  <div class="card-image">
    
    <span class="card-title">{{ product.name }}
  </div>
  <div class="card-content">
    <p>{{ product.description }}</p>
  </div>
</div>
```

- 5. Include Materialize initialization code:** Some Materialize components require initialization. You can do this by adding the following script at the end of your HTML files:

```
<script>
  M.AutoInit();
</script>
```

PROMPT 12



JUNIOR
DEVELOPER

I need to implement unit tests for this python web app. Can you recommend me best framework for my case (using Flask) and also provide step by step instructions on how to setup project, which packages to install etc.

PROMPT 12 - RESULT

GitHub Copilot

Used 1 reference >

Sure, for testing Flask applications, the Flask extension `pytest-flask` is a good choice. It's a plugin for `pytest` that provides a bunch of useful tools to make testing Flask applications easier.

Here are the steps to set up your project for testing:

1. **Install the necessary packages:** You need to install `pytest`, `pytest-flask`, and `requests-mock` for mocking HTTP requests. You can install them using pip:

```
pip install pytest pytest-flask requests-mock
```

2. **Create a test file:** Create a new Python file in your project directory for your tests. You can name it `test_app.py`.

3. **Import the necessary modules:** At the top of your test file, import the necessary modules:

```
import pytest
import requests_mock
from flask import url_for
from app import app
```

4. **Create a pytest fixture for your app:** Pytest fixtures are functions that are run before each test function to which it is applied. You can use a fixture to initialize your Flask app:

```
@pytest.fixture
def client():
    app.config['TESTING'] = True
    with app.test_client() as client:
        yield client
```

5. **Write your tests:** Now you can write your tests. Here's an example of a test for the `/details` route:

```
def test_details(client, requests_mock):
    mock_response = [
        {"Id": 1, "Name": "Product 1", "Desc"},
        {"Id": 2, "Name": "Product 2", "Desc"}
    ]
    requests_mock.get('http://localhost:5000',
                      response=mock_response)
    response = client.get(url_for('details'))
    assert response.status_code == 200
    assert b"Product 1" in response.data
```

PROMPT 13 – FIX ERRORS



JUNIOR
DEVELOPER

When i run tests with pytest i get following error :

"E RuntimeError: Working outside of application context. This typically means that you attempted to use functionality that needed E the current application. To solve this, set up an application context E with app.app_context(). See the documentation for more information."

Can you provide code changes in form of entire file? Provide concrete code for my example without needing to change or replace anything.

PROMPT 13 - RESULT

Sure, the error you're seeing is because the `url_for` function is being called outside of an application context. This function needs the application context to generate URLs. You can solve this by creating an application context in your test. Here is the modified code:

```
import pytest
import requests_mock
from flask import url_for
from app import app

@pytest.fixture
def client():
    app.config['TESTING'] = True
    with app.app_context(): # Create an applicati
        with app.test_client() as client:
            yield client

def test_details(client, requests_mock):
```

PROMPT 14



JUNIOR
DEVELOPER

Now I get error:

RuntimeError: Unable to build URLs outside an active request without 'SERVER_NAME' configured. Also configure 'APPLICATION_ROOT' and 'PREFERRED_URL_SCHEME' as needed.

PROMPT 14 RESULT

This error occurs because Flask's `url_for` function needs to know the server name, application root, and URL scheme to generate URLs outside of a request context. You can solve this by setting these configuration values in your test setup. Here's how you can modify your code:

```
import pytest
import requests_mock
from flask import url_for
from app import app

@pytest.fixture
def client():
    app.config['TESTING'] = True
    app.config['SERVER_NAME'] = 'localhost:5000'
    app.config['APPLICATION_ROOT'] = '/' # Add th
    app.config['PREFERRED_URL_SCHEME'] = 'http' #
    with app.app_context(): # Create an applicati
```

PROMPT 15 – FIX ERRORS AGAIN NO 2



JUNIOR
DEVELOPER

Now i get following error:

```
"E TypeError: Product.init() missing 18 required positional arguments: 'Category', 'SubCategory', 'RegularPrice', 'SalePrice', 'ImageUrl', 'VideoUrl', 'SKU', 'StockStatus', 'Weight', 'Dimensions', 'Variants', 'Reviews', 'Ratings', 'RelatedProducts', 'MetaTitle', 'MetaDescription', 'Keywords', and 'ProductType'"
```


PROMPT 15 - RESULT

The error message indicates that the `Product` class requires 18 arguments for initialization, but they are not provided in the `mock_response`. You need to include these fields in your `mock_response`. Here's how you can modify your code:

```
def test_details(client, requests_mock):
    mock_response = [
        {
            "Id": 1,
            "Name": "Product 1",
            "Description": "Description 1",
            "Category": "Category 1",
            "SubCategory": "SubCategory 1",
            "RegularPrice": 100.0,
            "SalePrice": 90.0,
            "ImageUrl": "http://example.com/image1",
            "VideoUrl": "http://example.com/video1",
            "SKU": "SKU1"
        }
    ]
```

PROMPT 16



SENIOR
DEVELOPER

#file:home.html #file:app.py Add a refresh button to my home page. Html needs to be defined so it uses Materialized framework. Also click action needs to initiate refresh of products list. Refresh needs to be only for the list, not for the entire page so use asynchronous method. Also add a loader which will first wait for 2 seconds and then fade in the refreshed list. Keep fetching product from json file. Add custom classes to html elements to avoid similarity with public code.

PROMPT 16 RESULT

Used 1 reference >

Plan

1. Add a button for refreshing the products list in the home.html file. This button will have an onclick event that triggers a JavaScript function to refresh the products.
2. Define the JavaScript function that will be triggered when the refresh button is clicked. This function will use the fetch API to make an asynchronous GET request to the '/products' endpoint of the Flask app.
3. In the fetch_products function in app.py, add a delay of 2 seconds before returning the products list to simulate the loading time.
4. In the JavaScript function, add a loader that will be displayed while the products are being fetched. After the products are fetched, the loader will be hidden and the new products list will be displayed.
5. Use Materialize CSS classes for the refresh button and the

home.html

```
<button class="btn waves-effect waves-light" id="r
  <i class="material-icons right">refresh</i>
</button>
<div id="loader" class="preloader-wrapper big acti
  <div class="spinner-layer spinner-blue-only">
    <div class="circle-clipper left">
      <div class="circle"></div>
    </div><div class="gap-patch">
      <div class="circle"></div>
    </div><div class="circle-clipper right">
      <div class="circle"></div>
    </div>
  </div>
</div>
<script>
  async function refreshProducts() {
    document.getElementById('loader').style.di
```

PROMPT 17



SENIOR
DEVELOPER

```
#file:app.py Add logging to my application. Implement logging so that i can switch concrete logging destination afterwards. For development environment logg to console output. Generate me entire app.py code properly merged.
```

PROMPT 17 RESULT

GitHub Copilot

Used 1 reference >

Sure, I can help you add logging to your application. We will use Python's built-in `logging` module. Here's how you can do it:

```
import logging
from flask import Flask, send_from_directory, render_template
import requests
import time
from Product import Product

# Set up logging
logging.basicConfig(level=logging.DEBUG)
logger = logging.getLogger(__name__)

app = Flask(__name__)
```

```
@app.route('/products')
def products():
    logger.info('Products route accessed')
    time.sleep(2) # simulate loading time
    return send_from_directory('.', 'products.json')

@app.route('/details')
def details():
    logger.info('Details route accessed')
    product_id = request.args.get('id')
    products = fetch_products('http://localhost:5000/products')
    product = next((product for product in products if product['id'] == product_id))
    return render_template('details.html', product=product)

def fetch_products(url):
    logger.info(f'Fetching products from {url}')
    response = requests.get(url)
    product_dicts = response.json()
```

PROMPT 18



SENIOR
DEVELOPER

Can you add additional Test field to Product.py named „TestProperty“ of type string and also add corrections to all places, add properties to mocked file

#file:products.json , add properties to python class file:
#file:Product.py , add additional properties to test file:
#file:test_app.py and also for display in html templates:
#file:home.html and #file:details.html

PROMPT 18 RESULT

GitHub Copilot

Used 1 reference >

Sure, I can help you add the "TestProperty" field to your Product class and update all the relevant files. Here's how you can do it:

1. Update the Product class in `Product.py`:

```
class Product:
    def __init__(self, Id, Name, Description, Category):
        self.Id = Id
```

2. Add the "TestProperty" field to the product data in `products.json`:

```
[
  {
    "Id": 1,
    "Name": "Product 1",
    ...
    "ProductType": "Type1",
    "TestProperty": "Test value 1"
  },
  ...
]
```

3. Update the test data in `test_app.py`:

```
def test_details(client, requests_mock):
    mock_response = [
        {
            "Id": 1,
            "Name": "Product 1",
            ...
            "ProductType": "Type 1",
            "TestProperty": "Test value 1"
        }
        # Add a similar dictionary for the second
```

4. Display the "TestProperty" field in `home.html` and `details.html`:

In `home.html`:

```
<div class="card-content">
  ...<p>{{ product.Description }}</p>
  ...<p>{{ product.TestProperty }}</p>
</div>
```

In `details.html`:

```
<div class="card-content">
  <p>{{ product.Description }}</p>
  <p>{{ product.TestProperty }}</p>
</div>
```

HANDS-ON EXERCISES & CHALLENGES

Let's check what we've learned!

CHALLENGE 1

Your task in this challenge is to set up a local virtual Python environment for web application development. A virtual environment allows you to isolate your Python project and its dependencies from other projects, ensuring that each project has its own set of dependencies that won't disrupt or conflict with each other.

This setup is crucial for maintaining clean and organized Python projects, especially when these projects have different dependencies and versions. It also makes your projects more portable and easier to share with others.

CHALLENGE 2

Your task in this challenge is to create a simple "Hello World" application using Flask. Flask is a popular web framework in Python used for developing web applications. A "Hello World" application is often the first program written when learning a new language or framework, serving as a simple test and introduction.

CHALLENGE 3

Your task in this challenge is to create a list of mocked objects for display based on the given JSON data. Mock objects are simulated objects that mimic the behavior of real objects in controlled ways. They are useful for testing and for situations where creating real objects is impractical or impossible.

In this case, you're given a JSON data structure of product objects, and your task is to create corresponding Python objects.

JSON example can be found in Notes section below

CHALLENGE 4

Your task in this challenge is to use the product list from Challenge 3 on the home page of your application to display the products. You'll need to integrate the list of Product objects into your Flask application and create a way to display them on the home page.

CHALLENGE 5

Your task in this challenge is to add example unit tests to your project. Unit tests are an essential part of software development, designed to verify that individual units of source code (e.g., functions or methods) are working correctly.

CHALLENGE 6

Your task in this challenge is to create an Azure DevOps pipeline for your Flask application. Azure DevOps is a Microsoft product that provides version control, reporting, requirements management, project management, automated builds, testing and release capabilities. It helps in automating the software delivery process and provides continuous integration and delivery capabilities.