**Usage Guide: Setting Up Clean Architecture in .NET (PPTify)**

**Table of Contents**

1. Prerequisites
2. Project Structure Setup
3. Creating the Solution and Projects
4. Adding Projects to the Solution
5. Setting Up Layered References
6. Creating Folder Structure with Files
7. Building and Running the Solution

**1. Prerequisites**

Ensure the following tools are installed on your system:

* .NET SDK (>= .NET 6.0)
* Visual Studio or VS Code
* Git (optional)
* PowerShell or CMD (for scripting)

**2. Project Structure Setup**

Create a root folder for your solution:

mkdir PPTify

cd PPTify

Create subfolders:

mkdir src tests docs scripts

**3. Creating the Solution and Projects**

Generate the solution and projects:

dotnet new sln -n PPTify

dotnet new webapi -n PPTify.Api -o src\Api\PPTify.Api

dotnet new classlib -n PPTify.Application -o src\Application\PPTify.Application

dotnet new classlib -n PPTify.Domain -o src\Domain\PPTify.Domain

dotnet new classlib -n PPTify.Infrastructure -o src\Infrastructure\PPTify.Infrastructure

dotnet new classlib -n PPTify.Shared -o src\Shared\PPTify.Shared

Create test projects:

dotnet new xunit -n PPTify.Tests.Unit -o tests\PPTify.Tests.Unit

dotnet new xunit -n PPTify.Tests.Integration -o tests\PPTify.Tests.Integration

dotnet new xunit -n PPTify.Tests.Functional -o tests\PPTify.Tests.Functional

**4. Adding Projects to the Solution**

Add all projects to the solution file:

dotnet sln add src\Api\PPTify.Api\PPTify.Api.csproj

dotnet sln add src\Application\PPTify.Application\PPTify.Application.csproj

dotnet sln add src\Domain\PPTify.Domain\PPTify.Domain.csproj

dotnet sln add src\Infrastructure\PPTify.Infrastructure\PPTify.Infrastructure.csproj

dotnet sln add src\Shared\PPTify.Shared\PPTify.Shared.csproj

dotnet sln add tests\PPTify.Tests.Unit\PPTify.Tests.Unit.csproj

dotnet sln add tests\PPTify.Tests.Integration\PPTify.Tests.Integration.csproj

dotnet sln add tests\PPTify.Tests.Functional\PPTify.Tests.Functional.csproj

**5. Setting Up Layered References**

Establish references according to Clean Architecture:

dotnet add src\Application\PPTify.Application\PPTify.Application.csproj reference src\Domain\PPTify.Domain\PPTify.Domain.csproj

dotnet add src\Infrastructure\PPTify.Infrastructure\PPTify.Infrastructure.csproj reference src\Application\PPTify.Application\PPTify.Application.csproj

dotnet add src\Infrastructure\PPTify.Infrastructure\PPTify.Infrastructure.csproj reference src\Domain\PPTify.Domain\PPTify.Domain.csproj

dotnet add src\Infrastructure\PPTify.Infrastructure\PPTify.Infrastructure.csproj reference src\Shared\PPTify.Shared\PPTify.Shared.csproj

dotnet add src\Api\PPTify.Api\PPTify.Api.csproj reference src\Application\PPTify.Application\PPTify.Application.csproj

dotnet add src\Api\PPTify.Api\PPTify.Api.csproj reference src\Infrastructure\PPTify.Infrastructure\PPTify.Infrastructure.csproj

dotnet add src\Api\PPTify.Api\PPTify.Api.csproj reference src\Shared\PPTify.Shared\PPTify.Shared.csproj

**6. Creating Folder Structure with Files**

Create feature folders and empty files:

mkdir src\Application\PPTify.Application\Features\ExampleFeature\Commands

mkdir src\Application\PPTify.Application\Features\ExampleFeature\Queries

mkdir src\Application\PPTify.Application\Features\ExampleFeature\Handlers

mkdir src\Application\PPTify.Application\Features\ExampleFeature\Validators

mkdir src\Domain\PPTify.Domain\Entities

mkdir src\Domain\PPTify.Domain\ValueObjects

mkdir src\Infrastructure\PPTify.Infrastructure\Persistence\Repositories

Use the type nul > filename.cs command to add placeholder files:

type nul > src\Application\PPTify.Application\Features\ExampleFeature\Commands\CreateExampleCommand.cs

type nul > src\Domain\PPTify.Domain\Entities\ExampleEntity.cs

**7. Building and Running the Solution**

Build the solution:

dotnet build PPTify.sln

Run the API project:

dotnet run --project src\Api\PPTify.Api

The Web API should now be running and accessible via Swagger or Postman.

This usage guide helps any new developer or contributor set up the solution structure, link layers, and start development following the clean architecture pattern.