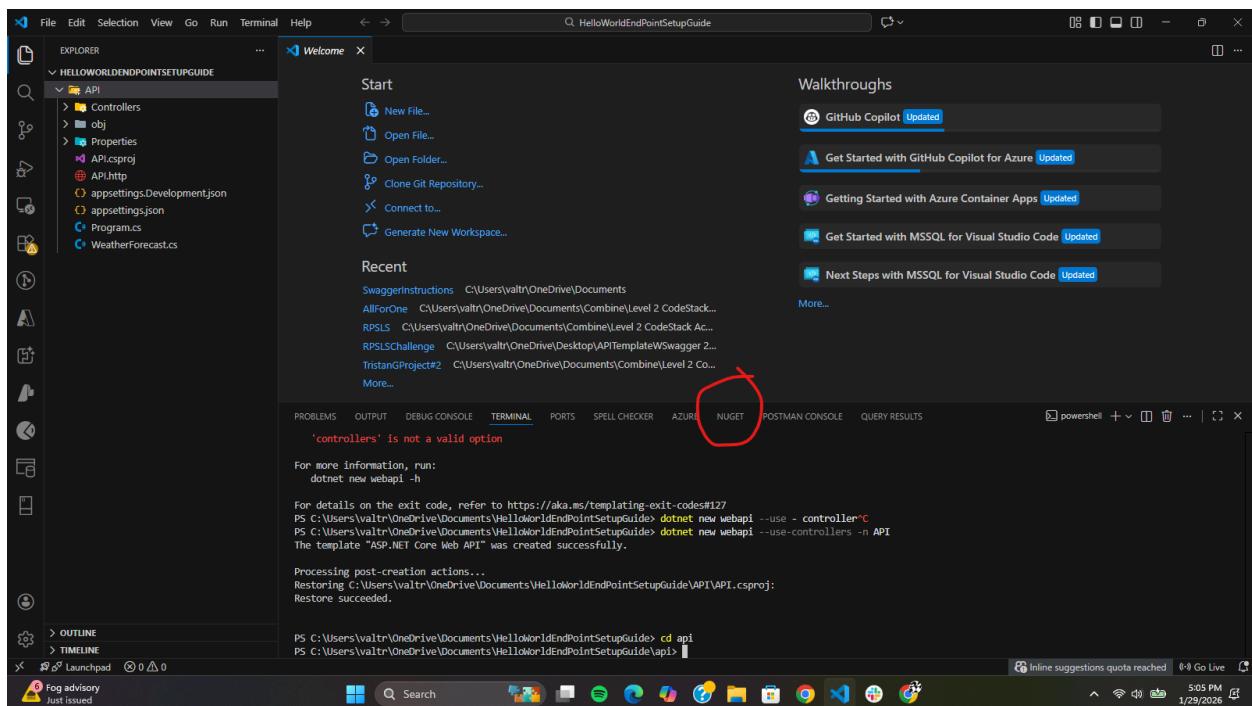


1. Project Initialization

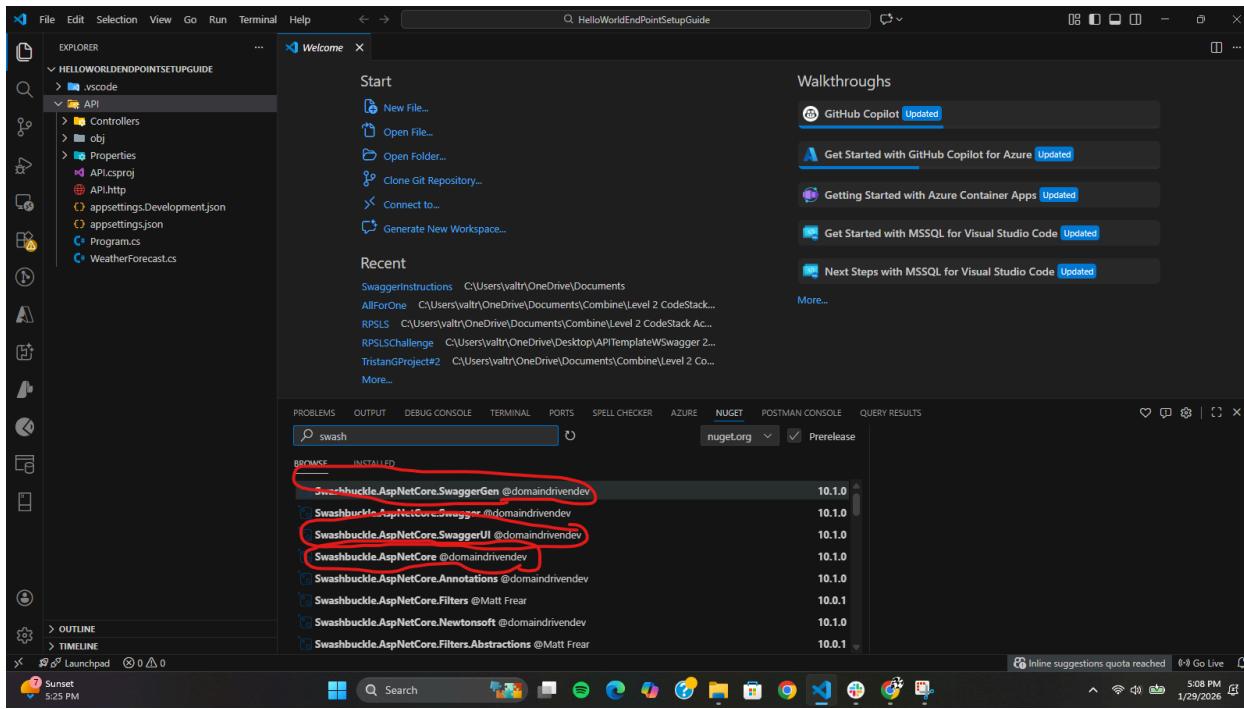
- Open the VS Code terminal (`Ctrl + ``).
- Create the project with the necessary structure:
 - **Command:** `dotnet new webapi --use-controllers -n API`
 - **Note:** The `-n API` creates a specific folder “API” being the name of the folder, and `--use-controllers` sets up the controller folder automatically and links it to the `program.cs`
- Navigate into the new folder: `cd API`

2. Install Dependencies (NuGet)

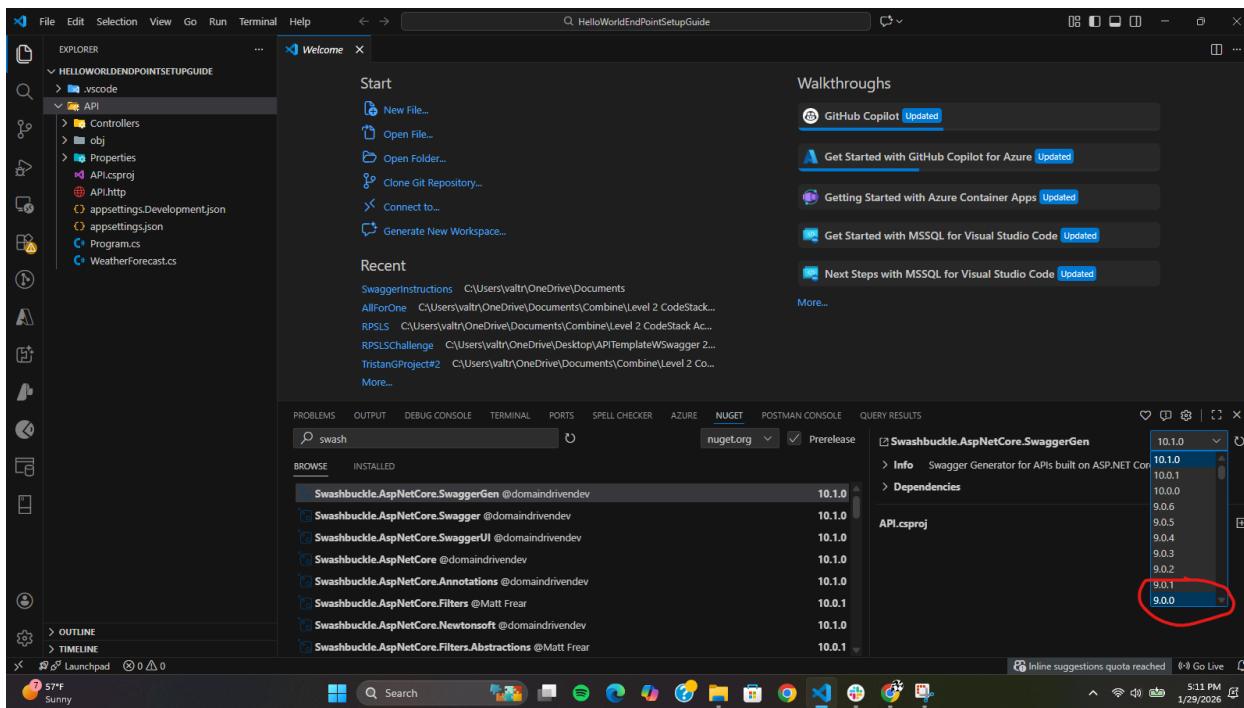
- Open the **NuGet** tab in VS Code.



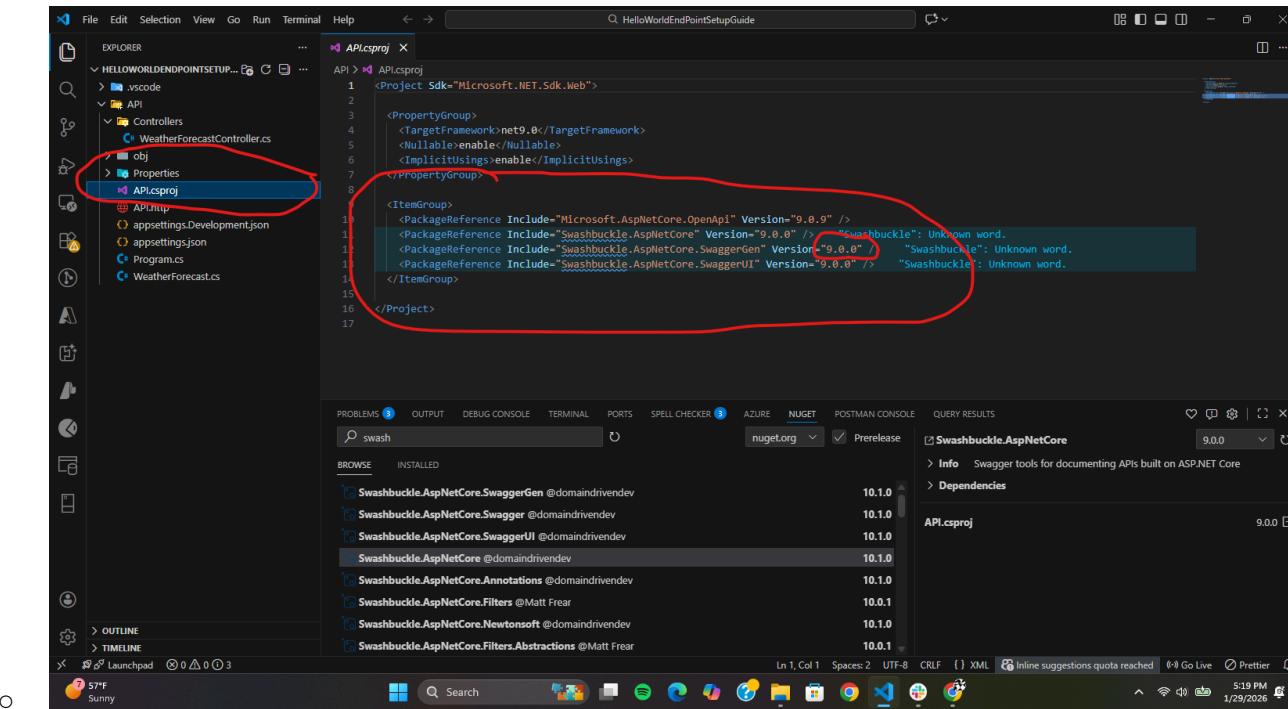
- Search for "**swash**".
- Download the **Swashbuckle** packages, SwaggerGen, Swagger, SwaggerUI



- Crucial: Ensure you select Version 9.0.0.



- Verify installation by checking your .csproj file.



3. Configure Launch Settings

- Open **Properties > launchSettings.json**.

```
1 {
2     "$schema": "https://json.schemastore.org/launchsettings.json",
3     "profiles": {
4         "http": {
5             "commandName": "Project",
6             "dotnetRunMessages": true,
7             "launchBrowser": false,
8             "applicationUrl": "http://localhost:5266",
9             "environmentVariables": {
10                 "ASPNETCORE_ENVIRONMENT": "Development"
11             }
12         },
13         "https": {
14             "commandName": "Project",
15             "dotnetRunMessages": true,
16             "launchBrowser": false,
17             "applicationUrl": "https://localhost:7209;http://localhost:5266",
18             "environmentVariables": {
19                 "ASPNETCORE_ENVIRONMENT": "Development"
20             }
21         }
22     }
23 }
```

- Modify the profiles (both sections) to ensure the browser opens Swagger automatically:
 - Change "launchBrowser": false to true.
 - Add/Update "launchUrl": "swagger".

```
1 {
2     "$schema": "https://json.schemastore.org/launchsettings.json",
3     "profiles": {
4         "http": {
5             "commandName": "Project",
6             "dotnetRunMessages": true,
7             "launchBrowser": true,
8             "launchUrl": "swagger",
9             "applicationUrl": "http://localhost:5266",
10            "environmentVariables": {
11                "ASPNETCORE_ENVIRONMENT": "Development"
12            }
13        },
14        "https": {
15            "commandName": "Project",
16            "dotnetRunMessages": true,
17            "launchBrowser": false,
18            "launchUrl": "swagger",
19            "applicationUrl": "https://localhost:7209;http://localhost:5266",
20            "environmentVariables": {
21                "ASPNETCORE_ENVIRONMENT": "Development"
22            }
23        }
24    }
25 }
```

4. Configure Program.cs

You need to register Swagger services and middleware. So go to your program.cs file

Add to Builder Services:

C#

```
builder.Services.AddEndpointsApiExplorer();
```

```
builder.Services.AddSwaggerGen();
```

•

Update the App Environment: Replace `app.MapOpenApi()`; with:

C#

```
app.UseSwagger();
```

```
app.UseSwaggerUI();
```

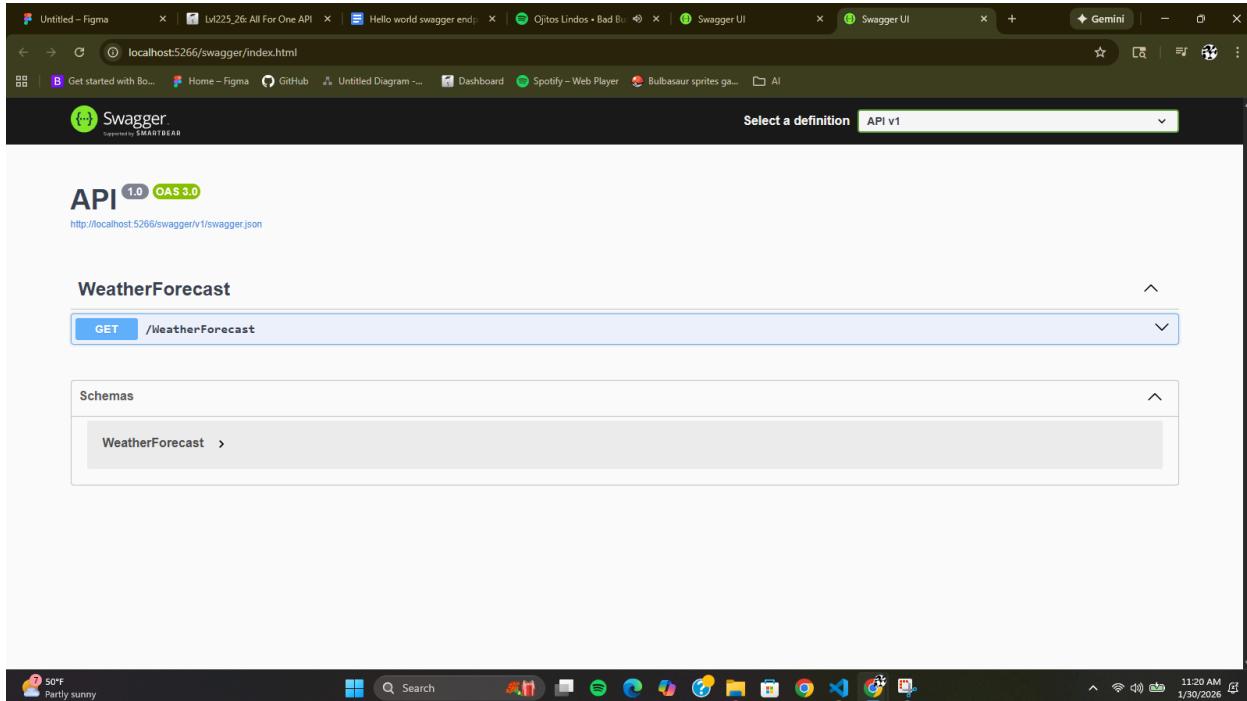
The screenshot shows the Visual Studio Code interface with the 'Program.cs' file open. The code is as follows:

```
API > C# Program.cs > Program > <stop-level-statements-entry-point>
1 var builder = WebApplication.CreateBuilder(args);
2
3 // Add services to the container.
4
5 builder.Services.AddControllers();
6 Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnet/openapi
7 builder.Services.AddEndpointsApiExplorer();
8 builder.Services.AddSwaggerGen();
9
10 var app = builder.Build();
11
12 // Configure the HTTP request pipeline.
13 if (app.Environment.IsDevelopment())
14 {
15     app.UseSwagger();
16     app.UseSwaggerUI();
17 }
18
19 app.UseHttpsRedirection();
20
21 app.UseAuthorization();
22
23 app.MapControllers();
24
25 app.Run();
26
```

Two sections of the code are circled with red highlights: the first section contains the three calls to `builder.Services.Add*` methods, and the second section contains the conditional block starting with `if (app.Environment.IsDevelopment())`.

•

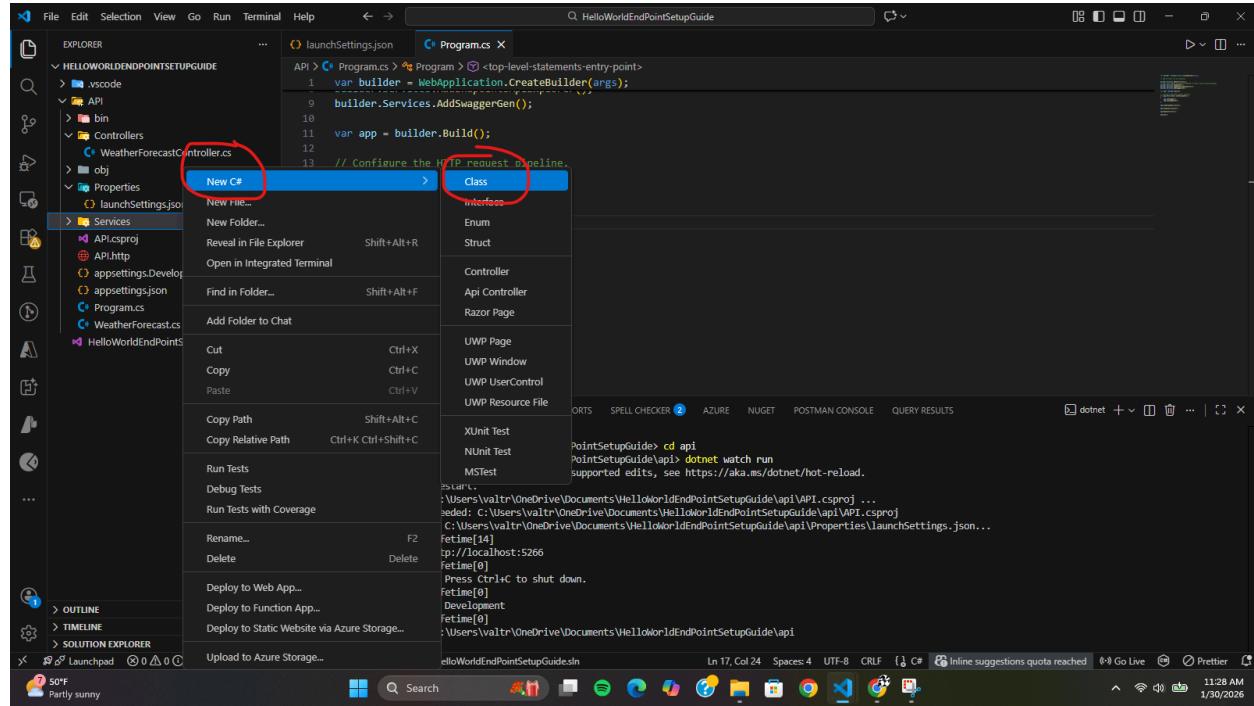
- **Test Setup:** Run `dotnet watch run` in the terminal. The Swagger UI should launch in your browser.



5. Create Services & Controllers

- **Services:**
 - Create a folder named "**Services**".

- Create a "New C# Class". Name it ending in "Service" (e.g., `HelloWorldService`).



- Adjust format: Remove curly brackets for the namespace and add a semi-colon at the end (file-scoped namespace).

A screenshot of the Visual Studio IDE interface. The title bar says "HelloWorldEndPointSetupGuide". The left sidebar shows a project structure with files like .vscode, API, bin, Controllers (containing WeatherForecastController.cs), obj, Properties (containing launchSettings.json), Services (containing HelloWorldService.cs), API.csproj, API.https, appsettings.Development.json, appsettings.json, Program.cs, WeatherForecast.cs, and HelloWorldEndPointSetupGuide.sln. The main code editor window displays the file "HelloWorldService.cs" with the following code:

```
API > Services > HelloWorldService.cs ...
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Threading.Tasks;
5
6 namespace API.Services
7 {
8     public class HelloWorldService
9     {
10
11 }
12 }
```

The first two lines of the code are circled in red. The "PROBLEMS" tab in the bottom right shows multiple errors related to the file, such as "CS1514: { expected" and "CS1514: ; expected".

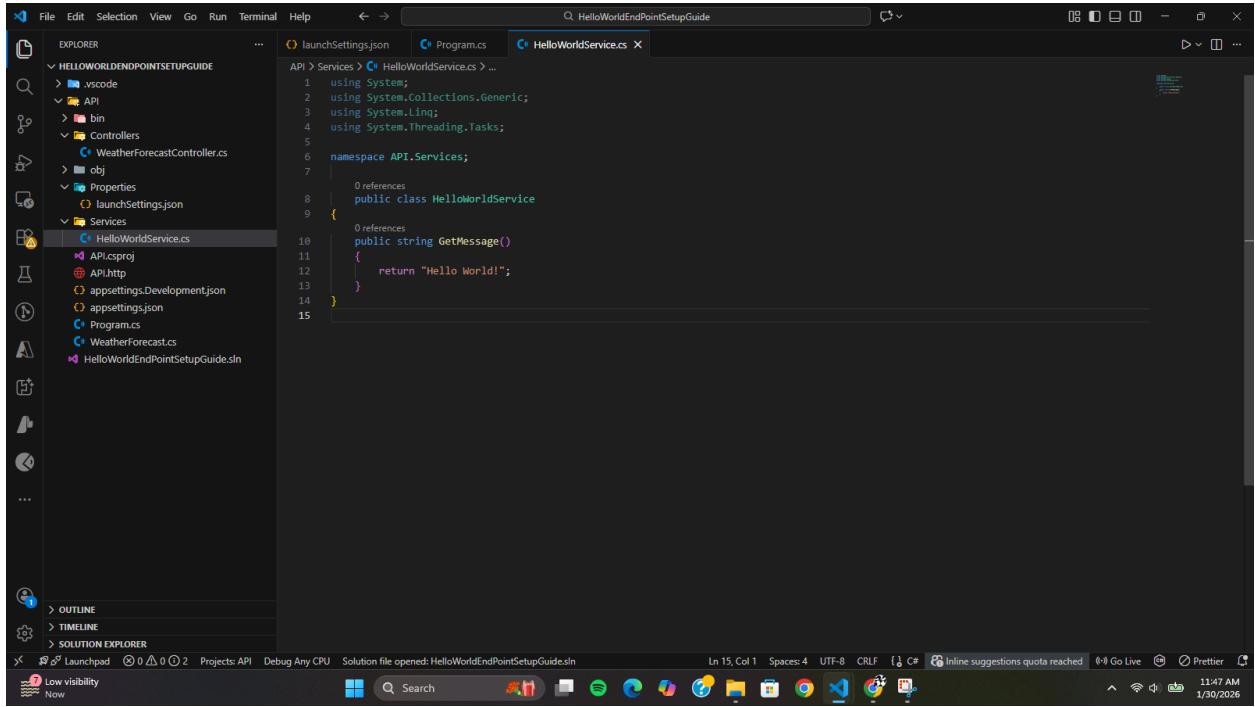
It should look something like this

A screenshot of the Visual Studio IDE interface, identical to the previous one but with the code fixed. The "HelloWorldService.cs" file now contains:

```
API > Services > HelloWorldService.cs ...
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Threading.Tasks;
5
6 namespace API.Services;
7
8     public class HelloWorldService
9     {
10
11 }
```

The code editor shows no errors in the "PROBLEMS" tab.

And when you've coded out your Service file it should look something like this



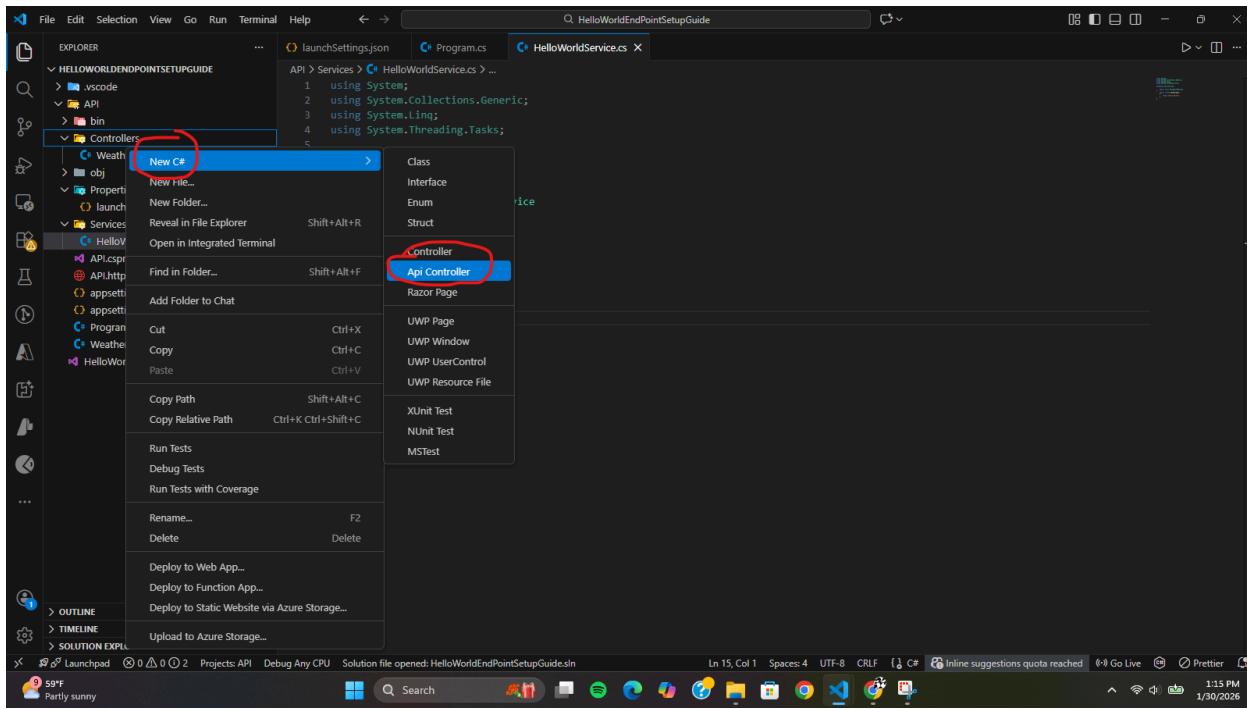
The screenshot shows the Visual Studio code editor with the following details:

- File Explorer:** Shows the project structure under "HelloworldEndPointSetupGuide". It includes ".vscode", "bin", "Controllers" (containing "WeatherForecastController.cs"), "obj", "Properties" (containing "launchSettings.json"), "Services" (containing "HelloWorldService.cs"), "API.csproj", "API.http", "appsettings.Development.json", "appsettings.json", "Program.cs", and "WeatherForecast.cs".
- Code Editor:** Displays the "HelloWorldService.cs" file with the following C# code:

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5
6  namespace API.Services;
7
8  {
9      public class HelloWorldService
10     {
11         public string GetMessage()
12         {
13             return "Hello World!";
14         }
15     }
}
```
- Solution Explorer:** Shows the solution configuration: "Launched" (0), "0", "0", "2".
- Bottom Status Bar:** Shows "Projects: API" and "Debug Any CPU".
- System Tray:** Shows icons for Task View, Search, Taskbar, File Explorer, Edge, and others.
- Bottom Right:** Shows the date and time: "11/17 AM 1/30/2026".

- **Controllers:**

- Right-click the existing "**Controllers**" folder.
- Select "New C# Api Controller". Name it ending in "Controller" (e.g., **HelloWorldController**).



- Adjust format to use file-scoped namespace (same as the Service file).

And when your done your Controller file should look something like this

```

File Edit Selection View Go Run Terminal Help <- > Q HelloWorldEndPointSetupGuide C Program.cs C WeatherForecastController.cs C HelloWorldService.cs C HelloWorldController.cs
EXPLORER ... launchSettings.json API > Controllers C HelloWorldController.cs > HelloWorldController > Get ...
> .vscode
> API
> bin
> Controllers
C HelloWorldController.cs
C WeatherForecastController.cs
> obj
Properties
> launchSettings.json
Services
C HelloWorldService.cs
API.csproj
API.Http
C appsettings
C appsettings.json
C Program.cs
C WeatherForecast.cs
HelloWorldEndPointSetupGuide.sln
C HelloWorldController.cs
C WeatherForecastController.cs
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Threading.Tasks;
5
6 using API.Services;
7 using Microsoft.AspNetCore.Mvc;
8
9 namespace API.Controllers;
10
11 [ApiController]
12 [Route("api/{controller}")]
13 public class HelloWorldController : ControllerBase
14 {
15     private readonly HelloWorldService service;
16
17     public HelloWorldController(HelloWorldService service)
18     {
19         this.service = service;
20     }
21
22     [HttpGet]
23     public IActionResult Get()
24     {
25         return Ok(service.GetMessage());
26     }
27

```

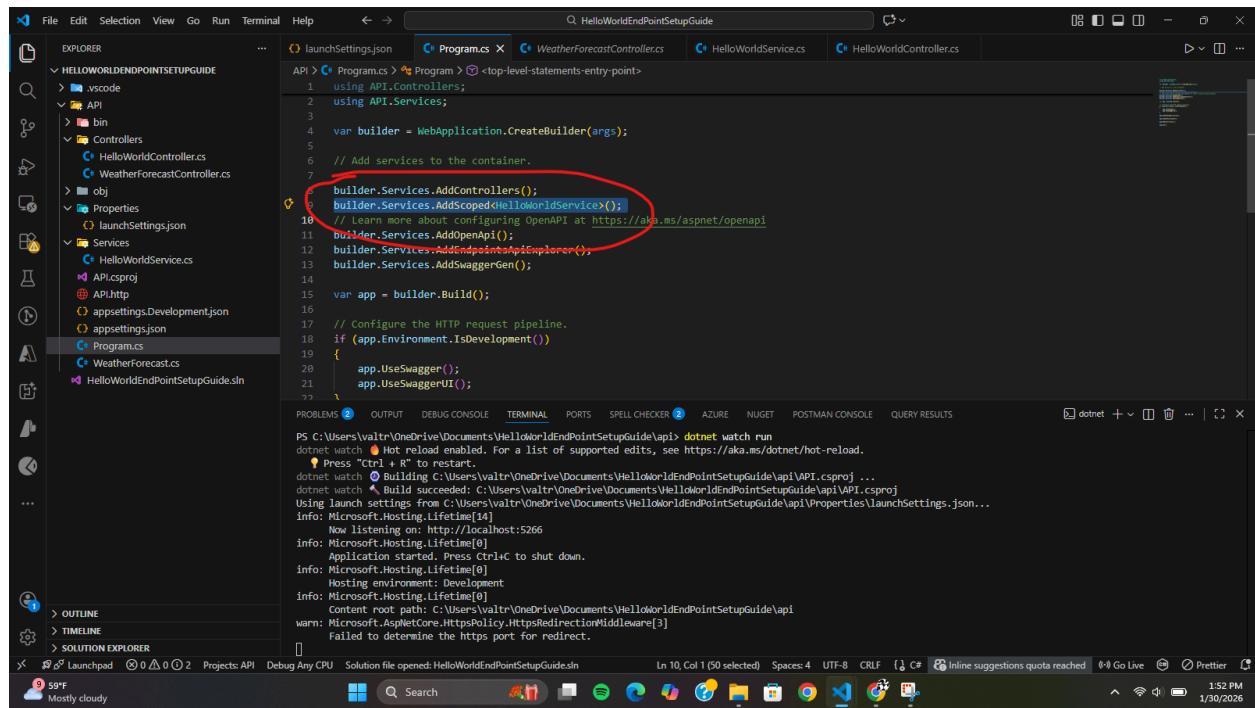
6. Dependency Injection & Final Run

- Register the Service:

- Go back to `Program.cs`.
- Add this line under `AddControllers`:

C#

```
builder.Services.AddScoped<HelloWorldService>();
```



The screenshot shows the Visual Studio IDE with the project 'HELLOWORLDENDPOINTSETUPGUIDE' open. The code editor displays `Program.cs` with the following code:

```
API > C# Program.cs > Program > <top-level-statements-entry-point>
1 using API.Controllers;
2 using API.Services;
3
4 var builder = WebApplication.CreateBuilder(args);
5
6 // Add services to the container.
7
8 builder.Services.AddControllers();
9 builder.Services.AddScoped<HelloWorldService>();
10 // Learn more about configuring OpenAPI at https://aka.ms/aspnet/openapi
11 builder.Services.AddOpenApi();
12 builder.Services.AddSwaggerUI();
13 builder.Services.AddSwaggerGen();
14
15 var app = builder.Build();
16
17 // Configure the HTTP request pipeline.
18 if (app.Environment.IsDevelopment())
19 {
20     app.UseSwagger();
21     app.UseSwaggerUI();
22 }
```

The line `builder.Services.AddScoped<HelloWorldService>();` is circled in red. The output window below shows the command `dotnet watch run` being executed, and the application is running on `http://localhost:5266`.

- Execute:

- Run `dotnet watch run`.
- Remember to `cd` in your project!

Untitled - Figma | Dashboard | Hello world swagger endpoint | Spotify - Liked Songs | Swagger UI | Gemini

localhost:5266/swagger/index.html

Swagger UI

Select a definition API v1

API 1.0 OAS 3.0
http://localhost:5266/swagger/v1/swagger.json

HelloWorld

GET /api/HelloWorld

WeatherForecast

GET /WeatherForecast

Schemas

WeatherForecast >

- In the browser, click **Get** -> **Try it out** -> **Execute** to test your endpoint.

Untitled - Figma | Dashboard | Hello world swagger endpoint | Spotify - Liked Songs | Swagger UI | Gemini

localhost:5266/swagger/index.html

Swagger UI

HelloWorld

GET /api/HelloWorld

Parameters

No parameters

Responses

curl -X 'GET' \n'http://localhost:5266/api/HelloWorld' \n-H 'accept: */*' Cancel

Request URL

http://localhost:5266/api/HelloWorld

Server response

Code Details

200 Response body

Hello World! Download

Good job you just did the hello world mini challenge with endpoints and setup swagger!