1. **Install & Configure Swagger in ASP.NET Core Web API**

### Step-by-Step Instructions:

#### ****a. Install NuGet Package****

* Open **Package Manager Console** in Visual Studio.
* Install-Package Swashbuckle.AspNetCore

#### ****b. Modify**** Program.cs ****to Add Swagger****

var builder = WebApplication.CreateBuilder(args);

// Add Swagger services

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

builder.Services.AddControllers();

var app = builder.Build();

// Use Swagger middleware

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(); // Default UI endpoint: /swagger

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

#### ****c. Add Response Metadata (Optional)****

In your controller methods, use [ProducesResponseType]:

[HttpGet]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status404NotFound)]

public IActionResult GetAll()

{

return Ok(new[] { "value1", "value2" });

}

Output

* Run your app.
* Navigate to:
* https://localhost:5001/swagger/index.html
* You’ll see a **UI listing your API endpoints**, including HTTP verbs and schema.

## 2. ****Demonstrate Usage of Postman****

### Structure in Postman:

#### a. ****Set Request Type****

Choose GET, POST, PUT, DELETE from dropdown next to URL bar.

#### b. ****Enter URL****

https://localhost:5001/api/values

#### c. ****Headers Tab****

For protected APIs, add:

Key: Authorization

Value: Bearer <your\_token\_here>

#### d. ****Body Tab****

Choose **raw → JSON**.

"value3"

#### e. ****Send Request****

* Click **Send**.
* The **response pane** will show the output (e.g., 200 OK, response data, headers, etc.).

**Postman Request Collection:**

* Click **Collections → New Collection**
* Name: My API Test
* Add a request: Right-click → Add Request
* Name the request, select method, URL, headers, body, etc.
* Click **Save to Collection**

### Center Pane Tabs in Postman:

| **Tab** | **Purpose** |
| --- | --- |
| **Params** | Query parameters |
| **Authorization** | Token or credentials |
| **Headers** | Custom headers |
| **Body** | Request payload |
| **Tests** | JS tests for response |
| **Pre-request Script** | Preprocessing script |

## 3. ****Routing,**** ActionName****, and**** Name ****Attribute in Web API****

### Basic Route Attribute

[Route("api/[controller]")]

[ApiController]

public class ItemsController : ControllerBase

{

[HttpGet("get-all", Name = "GetAllItems")]

public IActionResult GetAll() => Ok(new[] { "item1", "item2" });

}

### Name Attribute

Used to give a **user-friendly name** for a route.

Helps when generating links using Url.Link("GetAllItems", ...)

### 🔹 ActionName Attribute

Used to **differentiate multiple methods** with the same HTTP verb.

[HttpGet]

[ActionName("List")]

public IActionResult ListItems() => Ok("List of items");

[HttpGet("{id}")]

[ActionName("Get")]

public IActionResult GetItem(int id) => Ok($"Item {id}");

You can now access:

* /api/items/List
* /api/items/Get/1

## 4. Simple Web API: Read/Write Actions

### 🔹 Controller: ProductsController.cs

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

namespace WebAPI.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class ProductsController : ControllerBase

{

private static List<string> products = new List<string> { "Laptop", "Phone" };

[HttpGet]

public ActionResult<IEnumerable<string>> Get()

{

return Ok(products);

}

[HttpPost]

public ActionResult Add([FromBody] string product)

{

if (string.IsNullOrWhiteSpace(product))

return BadRequest("Invalid product");

products.Add(product);

return Ok("Added successfully");

}

}

}