* **Demonstrate Swagger installation to WebAPI and WebAPI listing on browser**
  + **Nuget package to download Swashbuckle.AspNetCore, Usage of ProducesResponseType to Web API method, AddSwaggerGen, UseSwaggerUI**

**Answer-->**

## What is Swagger?

**Swagger** (now called **OpenAPI**) provides:

* Interactive documentation
* Auto-generated UI for testing endpoints
* Helpful for both development and API consumers

## Step 1: ****Install Swagger using NuGet****

### NuGet Package:

Run this command in the **Package Manager Console**:

Install-Package Swashbuckle.AspNetCore

Or, if we're using the .NET CLI:

dotnet add package Swashbuckle.AspNetCore

## Step 2: ****Configure Swagger in**** Program.cs ****or**** Startup.cs

### If using Program.cs (ASP.NET Core 6+)

var builder = WebApplication.CreateBuilder(args);

// Add services

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(); //Add Swagger Generator

var app = builder.Build();

// Configure HTTP request pipelineif (app.Environment.IsDevelopment())

{

app.UseSwagger(); // Generate Swagger JSON

app.UseSwaggerUI(); // Enable Swagger UI

}

app.UseAuthorization();

app.MapControllers();

app.Run();

## Step 3: ****Run the API and View Swagger****

Run the project (F5 or dotnet run)

<https://localhost:<port>/swagger>

## Step 4: ****Use**** [ProducesResponseType] ****in Web API methods****

### Purpose:

Helps Swagger understand what responses to expect from your action methods.

### Example Controller:

[ApiController]

[Route("api/[controller]")]public class ProductController : ControllerBase

{

[HttpGet("{id}")]

[ProducesResponseType(typeof(Product), StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status404NotFound)]

public IActionResult GetProduct(int id)

{

var product = ProductStore.Find(p => p.Id == id);

if (product == null)

return NotFound();

return Ok(product);

}

}

* **Demonstrate the usage of Postman tool to hit WebAPI methods**
  + **Structure in Postman tool, Headers with Authorization, Body as JSON, Option to choose the type of request, Request collection and how to add a new request in the collection, Tabs in the center pane that corresponds to the request**

**Answer-->**

## What is Postman?

**Postman** is a GUI-based API testing tool that lets you:

* Send HTTP requests (GET, POST, PUT, DELETE, etc.)
* Add headers, body, params, and authentication
* View responses
* Group requests into collections

## Step-by-Step: Using Postman to Hit a WebAPI

### Example API Endpoint:

### POST https://localhost:5001/api/product

## 1. ****Postman Interface Structure****

### ****Left Panel**** – Collections, History

* + **Collections**: Organized groups of related API requests.
  + **History**: Recently made requests.

### ****Center Pane**** – Request Tabs

* + **Method dropdown** (GET/POST/etc.)
  + **URL bar** for endpoint
  + **Tabs** below: Params, Authorization, Headers, Body, Tests, etc.

### ****Bottom Panel**** – Response Area

* + Shows **status**, **time**, **body**, **headers**, etc.

## 2. ****Creating a Request****

### Choose HTTP Method:

* + Select GET, POST, PUT, DELETE, etc. from the dropdown next to the URL.

### Enter API URL:

<https://localhost:5001/api/product>

## 3. ****Set Headers (e.g., Authorization)****

* Click the **Headers** tab.
* Add a header:

| **Key** | **Value** |
| --- | --- |
| Content-Type | application/json |
| Authorization | Bearer <your\_token\_here> |

## 4. ****Add JSON Body (for POST/PUT)****

* Click on the **Body** tab.
* Choose **raw** and select **JSON** from the dropdown.
* Enter JSON like:

{

"name": "Tablet",

"price": 15000

}

## 5. ****Send the Request****

* Click the **Send** button.
* You’ll see the **response** at the bottom pane:
  + **Status**: 200 OK, 201 Created, etc.
  + **Response Body**
  + **Response Headers**

## 6. ****Create a Collection****

### Purpose:

Group related API calls (e.g., all Product APIs).

### Steps:

* + In the left pane, click **New > Collection**.
  + Give it a name like **ProductAPI**.
  + Click **Save to Collection** when saving a request.
  + Right-click collection to:
    - Add folders
    - Create new requests
    - Export collection
* **Demonstrate the usage of Route and Explain Name attribute in Http requests**
  + **Importance of user friendly name to action method, Explain the usage of ActionName to have more than 1 method with the same Action verbDemonstrate creation of a simple WebAPI - With Read, Write actions**

**Answer-->**

### [Route]:

Defines a **custom URL pattern** for an action method or controller.

### Name:

Gives a **friendly, reusable name** to a route (useful for generating URLs with CreatedAtRoute).

### ****Example****:

[HttpGet("{id}", Name = "GetProductById")]

public IActionResult GetById(int id)

{

var product = products.FirstOrDefault(p => p.Id == id);

if (product == null)

return NotFound();

return Ok(product);

}

### Used in:

return CreatedAtRoute("GetProductById", new { id = product.Id }, product);

## 2. ****Why Use**** Name ****Attribute?****

| **Feature** | **Benefit** |
| --- | --- |
| Name = "RouteName" | Enables reusing route in links or redirects |
| Friendly URLs | Makes routes readable and consistent |
| Better maintenance | Route updates only in one place |

## 3. ****Using**** [ActionName] ****Attribute****

[ActionName] allows **method overloading** using **same HTTP verb** but **different logic** or **parameter sets**.

### ****Example****:

[HttpGet]

[ActionName("Get")]public IActionResult GetAll()

{

return Ok(products);

}

[HttpGet("{id}")]

[ActionName("Get")]public IActionResult GetById(int id)

{

var product = products.FirstOrDefault(p => p.Id == id);

if (product == null)

return NotFound();

return Ok(product);

}

## 4. ****Complete Example: Read & Write API with Route, Name, ActionName****

using Microsoft.AspNetCore.Mvc;using System.Collections.Generic;using System.Linq;

namespace DemoWebAPI.Controllers

{

[ApiController]

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

public class ProductController : ControllerBase

{

// Simulated data

private static List<Product> products = new List<Product>

{

new Product { Id = 1, Name = "Laptop", Price = 45000 },

new Product { Id = 2, Name = "Phone", Price = 25000 }

};

// READ all

[HttpGet]

[ActionName("Get")]

public IActionResult GetAll()

{

return Ok(products);

}

// READ by ID

[HttpGet("{id}", Name = "GetProductById")]

[ActionName("Get")]

public IActionResult GetById(int id)

{

var product = products.FirstOrDefault(p => p.Id == id);

if (product == null)

return NotFound();

return Ok(product);

}

// WRITE - Create new

[HttpPost]

public IActionResult Create(Product product)

{

product.Id = products.Max(p => p.Id) + 1;

products.Add(product);

// Using named route to return Location header

return CreatedAtRoute("GetProductById", new { id = product.Id }, product);

}

}

// Product model

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

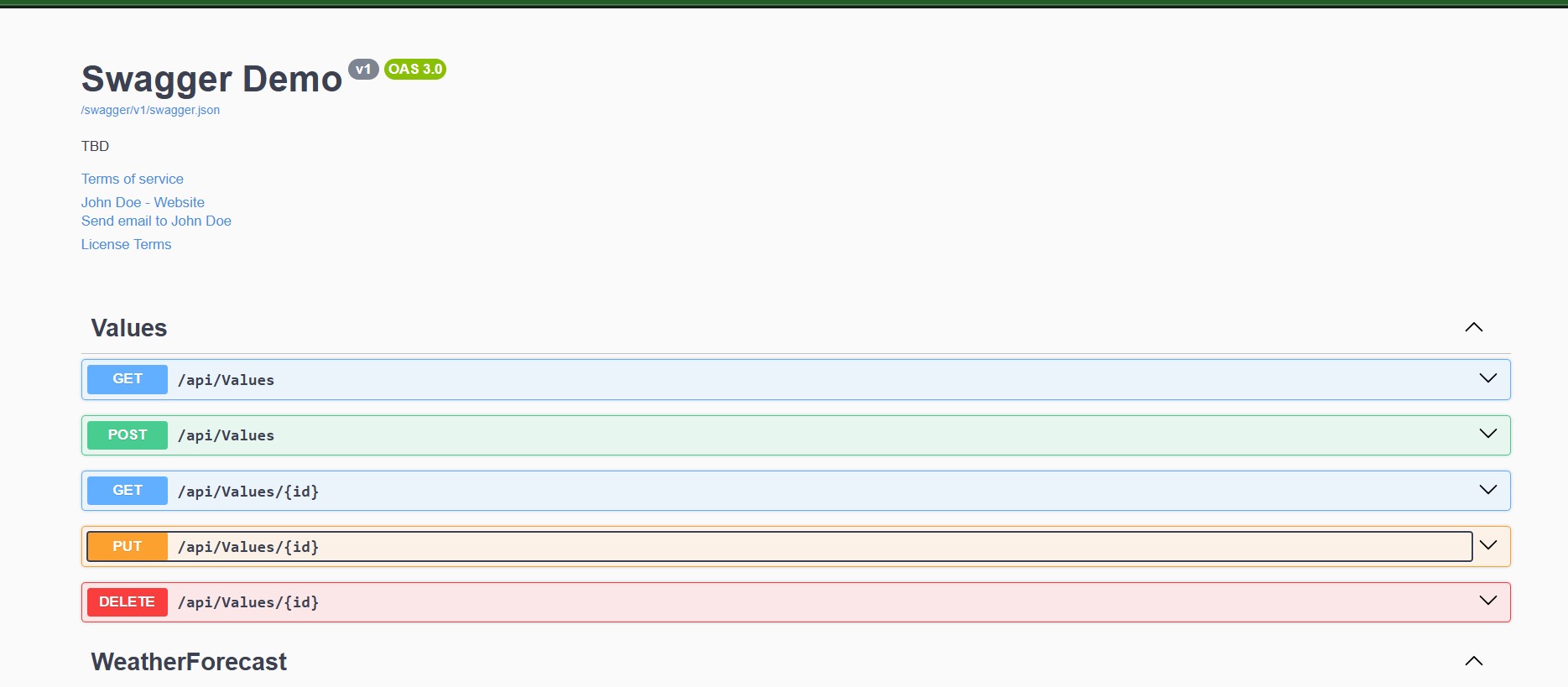
public double Price { get; set; }

}

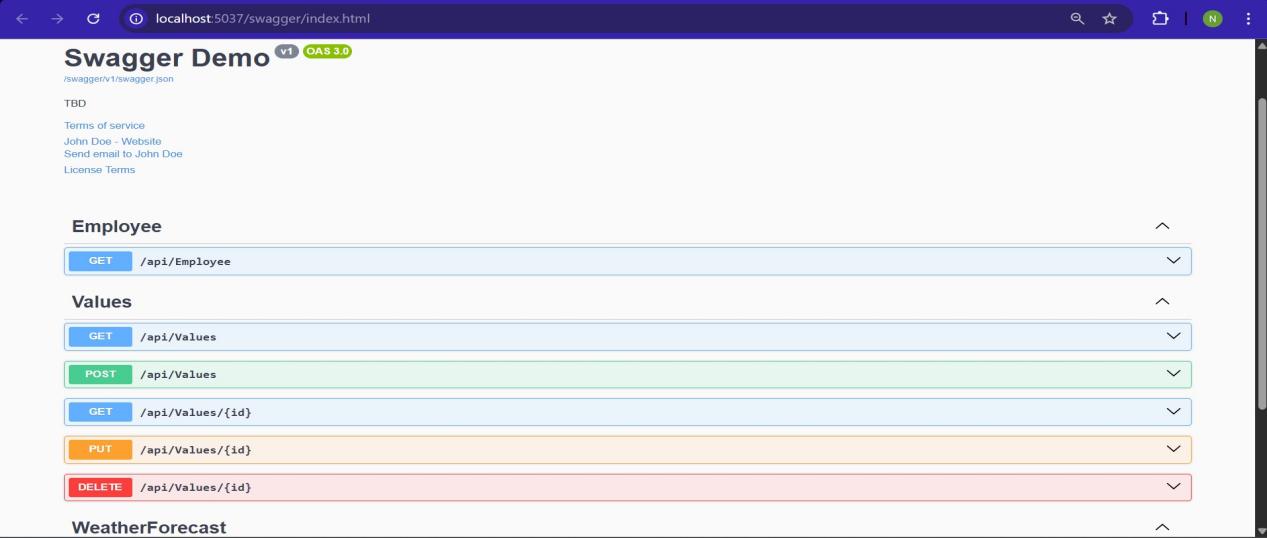
}

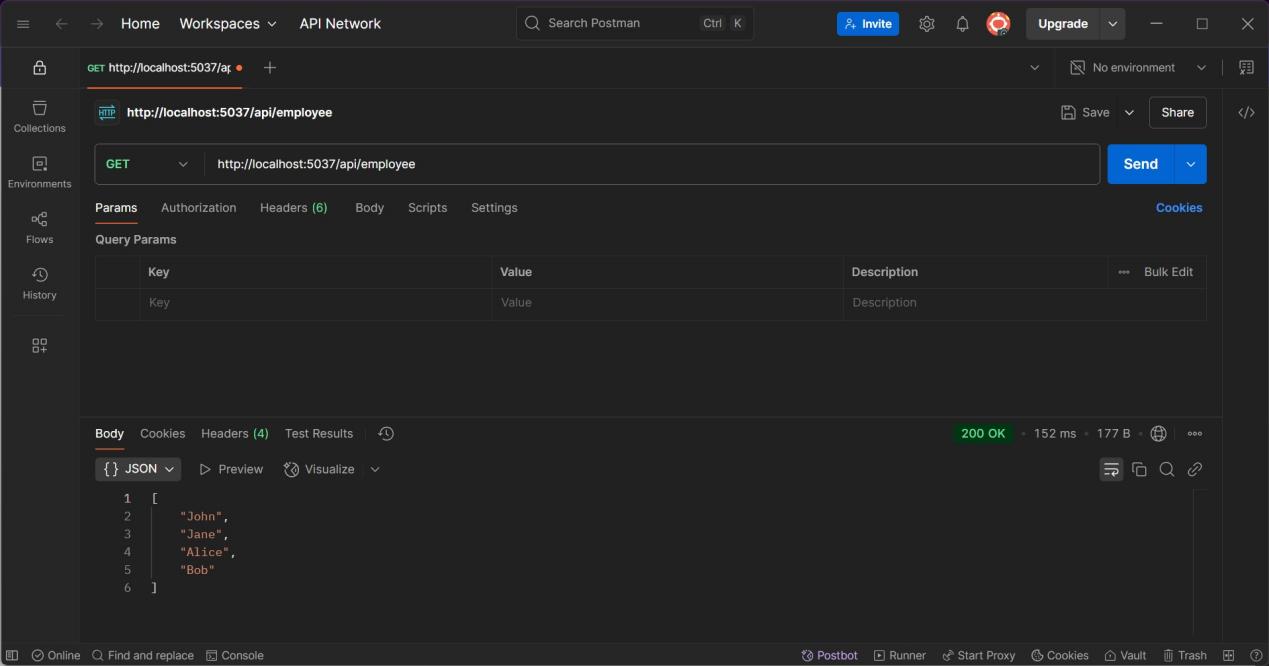
**Output:**

**1)**

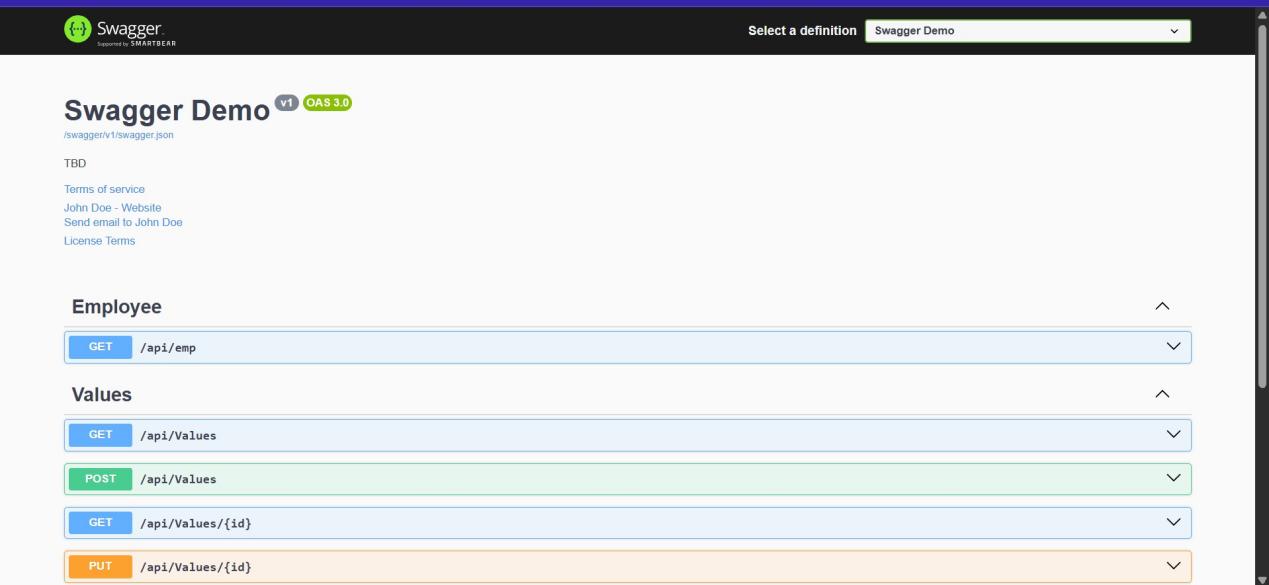
****

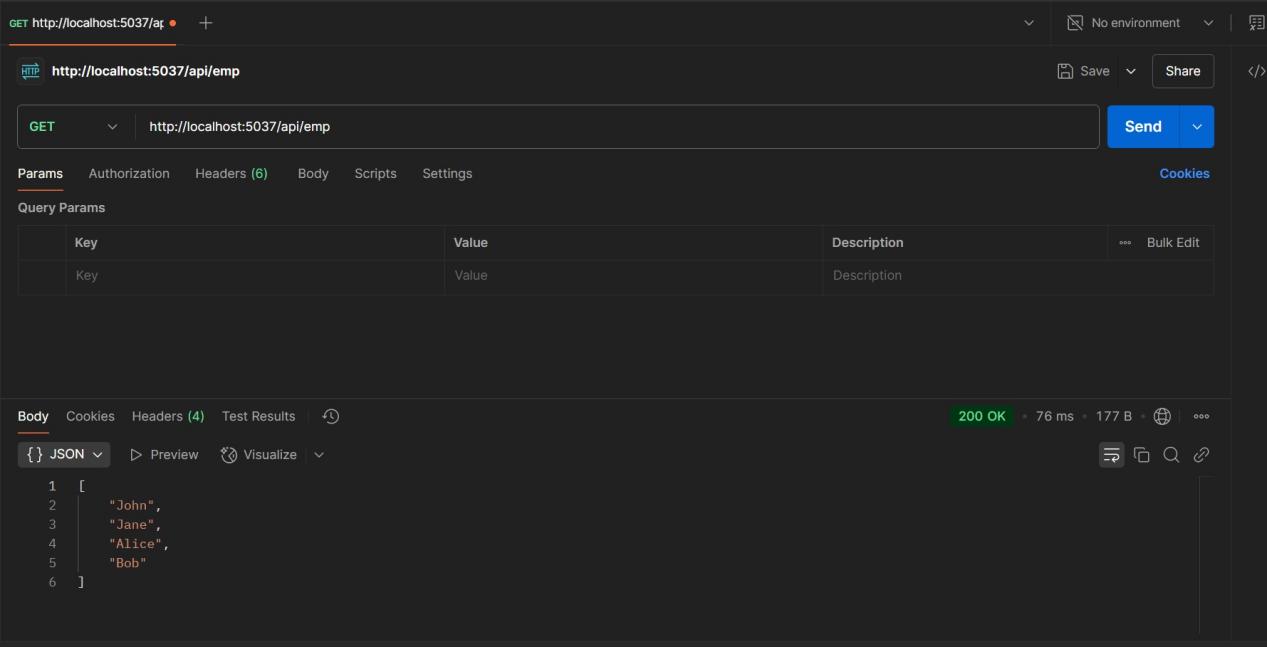
**2)**

****

****

**3)**

****

****