**Week 4: ASP .NET WEB API**

**1: First Web API Using .NET Core**

**1.Explain the concept of RESTful web service, Web API & Microservice**

|  |  |
| --- | --- |
| Feature | Explanation |
| REST (Representational State Transfer) | Architectural style for distributed systems using stateless communication |
| Web API | Framework for building HTTP-based services |
| Microservice | Small, independent service performing a specific function |
| WebService vs WebAPI | WebService: XML, SOAP-based; WebAPI: HTTP, JSON preferred |

**2.Explain what is HttpRequest & HttpResponse**

|  |  |
| --- | --- |
| Term | Explanation |
| HttpRequest | Carries client data to the server (headers, body, query, etc.) |
| HttpResponse | Server’s reply to the client’s request (status code, headers, content) |

**3.List the types of Action Verbs**

|  |  |  |
| --- | --- | --- |
| Verb | Description | Usage in Controller |
| HttpGet | Read data | [HttpGet] |
| HttpPost | Create new resource | [HttpPost] |
| HttpPut | Update existing resource | [HttpPut("{id}")] |
| HttpDelete | Delete a resource | [HttpDelete("{id}")] |

**4.List the types of HttpStatusCodes used in WebAPI**

|  |  |
| --- | --- |
| Status Code | Usage |
| 200 - OK | Request succeeded |
| 400 - BadRequest | Invalid input |
| 401 - Unauthorized | Not authenticated |
| 500 - InternalServerError | Unexpected server error |

**5.Demonstrate creation of a simple WebAPI - With Read, Write actions**

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

namespace FirstWebApi.Controllers

{

[ApiController]

[Route("[controller]")]

public class ValuesController : ControllerBase

{

private static List<string> data = new List<string> { "value1", "value2" };

[HttpGet]

public ActionResult<IEnumerable<string>> Get()

{

return Ok(data);

}

[HttpPost]

public ActionResult Post([FromBody] string value)

{

data.Add(value);

return Ok($"Added: {value}");

}

[HttpPut("{id}")]

public ActionResult Put(int id, [FromBody] string value)

{

if (id < 0 || id >= data.Count)

return BadRequest("Invalid ID");

data[id] = value;

return Ok($"Updated ID {id} to {value}");

}

[HttpDelete("{id}")]

public ActionResult Delete(int id)

{

if (id < 0 || id >= data.Count)

return BadRequest("Invalid ID");

var removed = data[id];

data.RemoveAt(id);

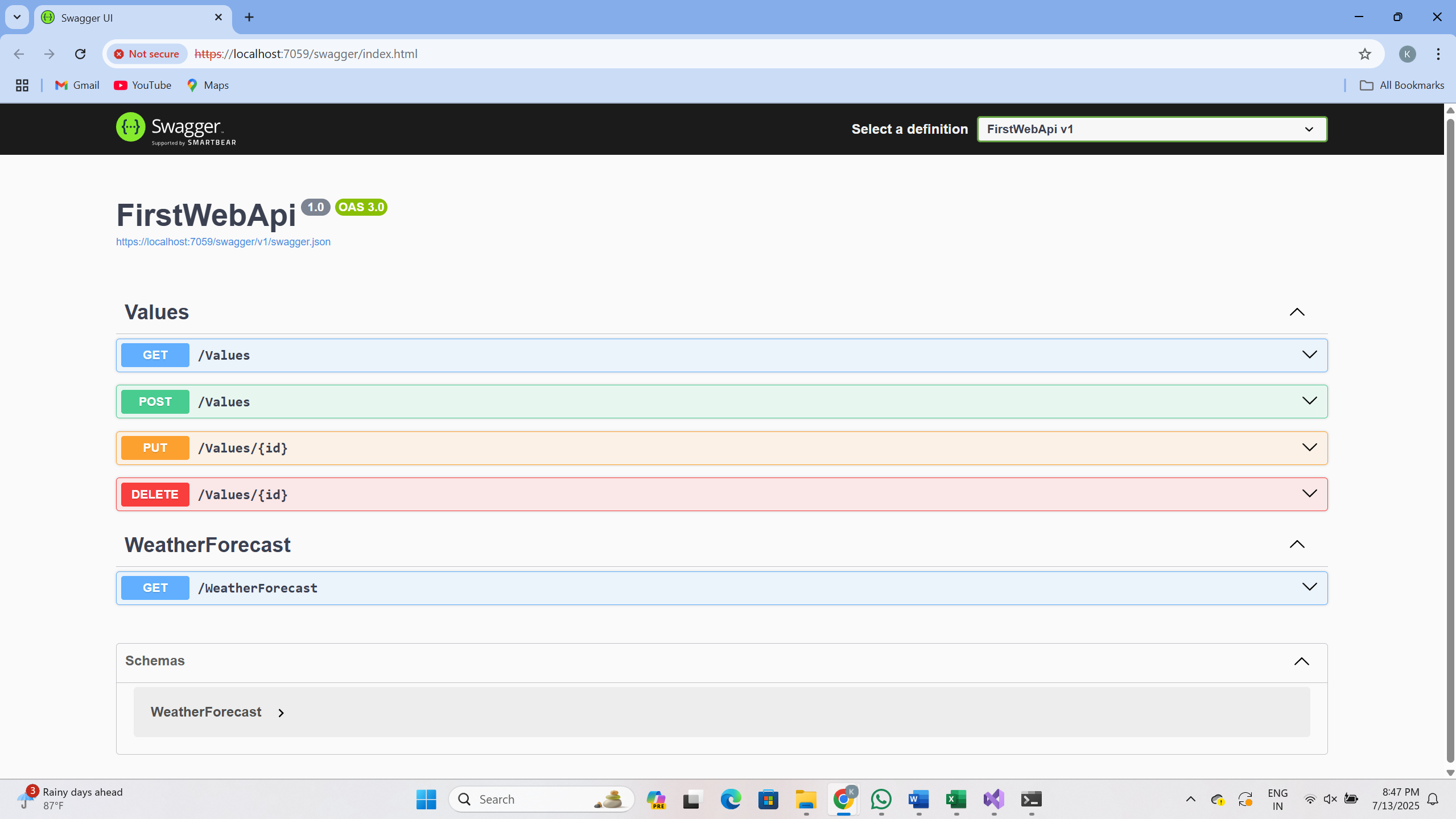
return Ok($"Deleted: {removed}");

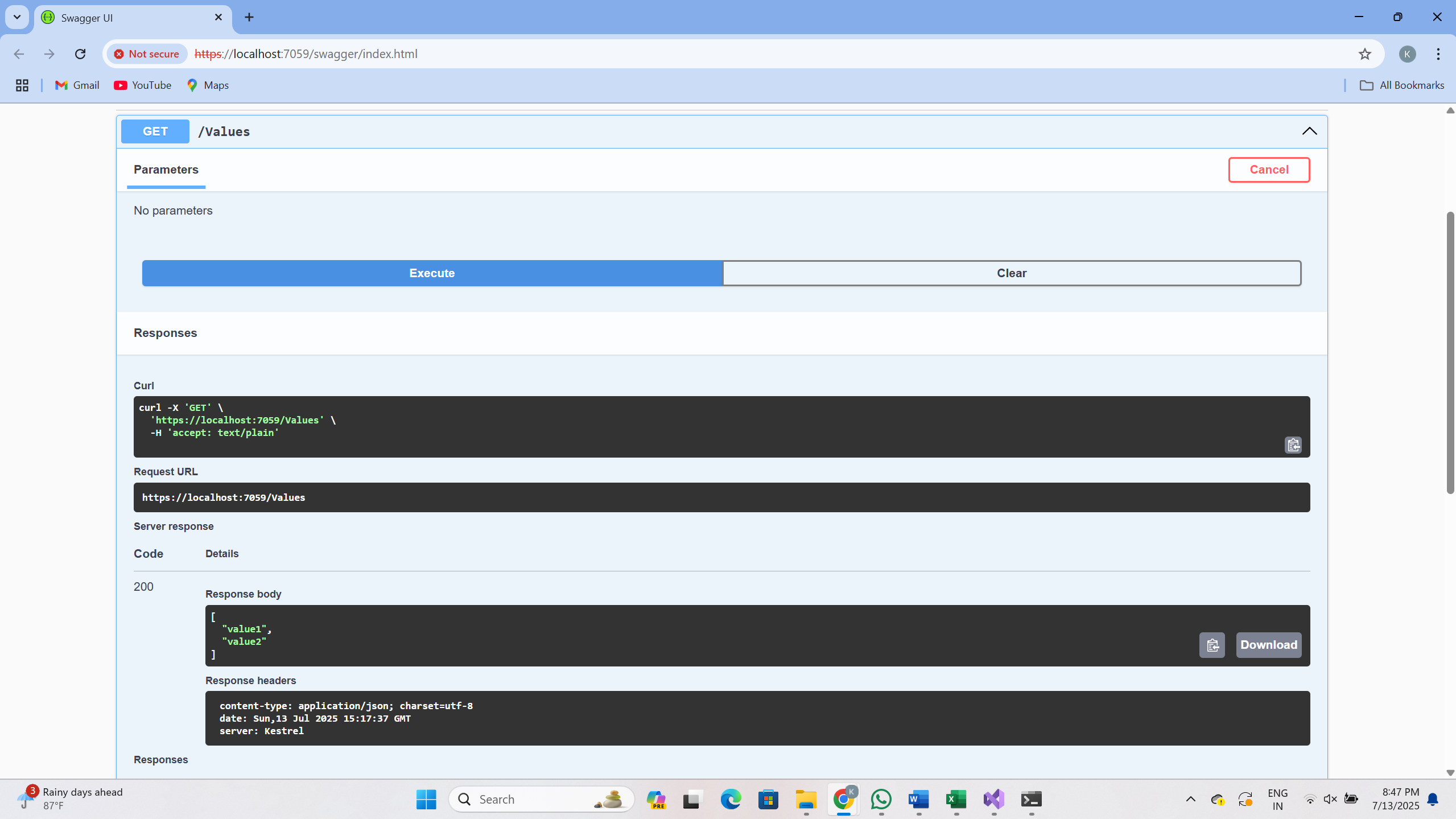
}

}

}

**Output:**

****

****

6.Explain the types of Configuration files of WebAPI

|  |  |
| --- | --- |
| File | Purpose |
| Program.cs / Startup.cs | Configure services and middleware (DI, routing) |
| appsettings.json | Application settings (e.g., connection strings) |
| launchSettings.json | Sets environment and browser launch settings during debug |
| Route.config / WebAPI.config (.NET 4.5) | Used for routing in classic ASP.NET Web API |

**2: Web API using Swagger and Postman**

**1: Install Swagger (Swashbuckle.AspNetCore)**

**NuGet Package Installed:**

Swashbuckle.AspNetCore

**Added to Program.cs:**

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new Microsoft.OpenApi.Models.OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "TBD",

TermsOfService = new Uri("https://example.com/terms"),

Contact = new Microsoft.OpenApi.Models.OpenApiContact

{

Name = "John Doe",

Email = "john@xyzmail.com",

Url = new Uri("https://www.example.com")

},

License = new Microsoft.OpenApi.Models.OpenApiLicense

{

Name = "License Terms",

Url = new Uri("https://www.example.com")

}

});

});

**Added middleware to Program.cs:**

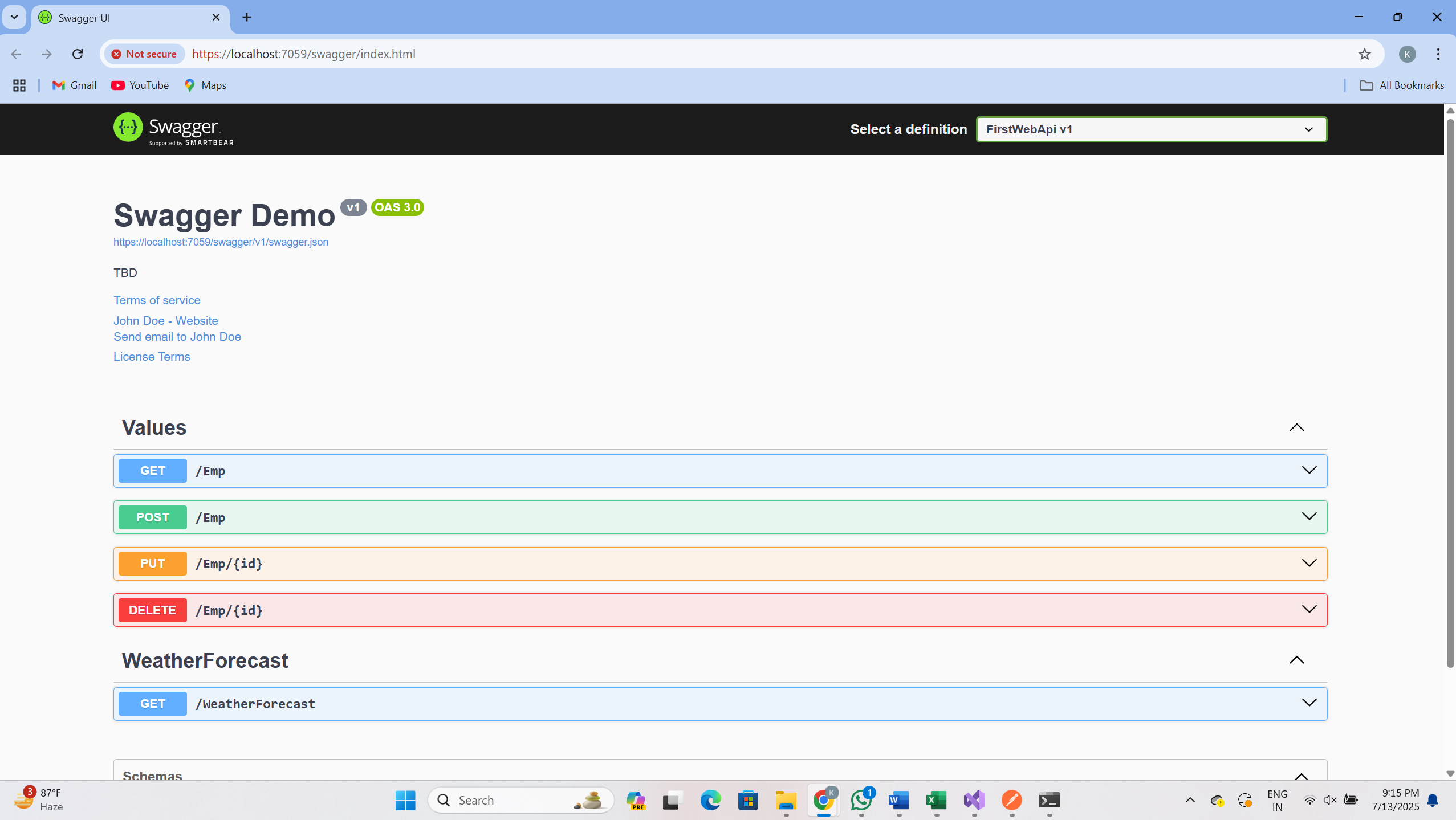
app.UseSwagger();

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "Swagger Demo");

});



**2.Test GET API with Postman**

**URL Used in Postman:**

https://localhost:7059/values

**Method**: GET

**Expected Output** (In Body):

["value1", "value2"]

**Status**: 200 OK

**3: Change Route Name in Controller**

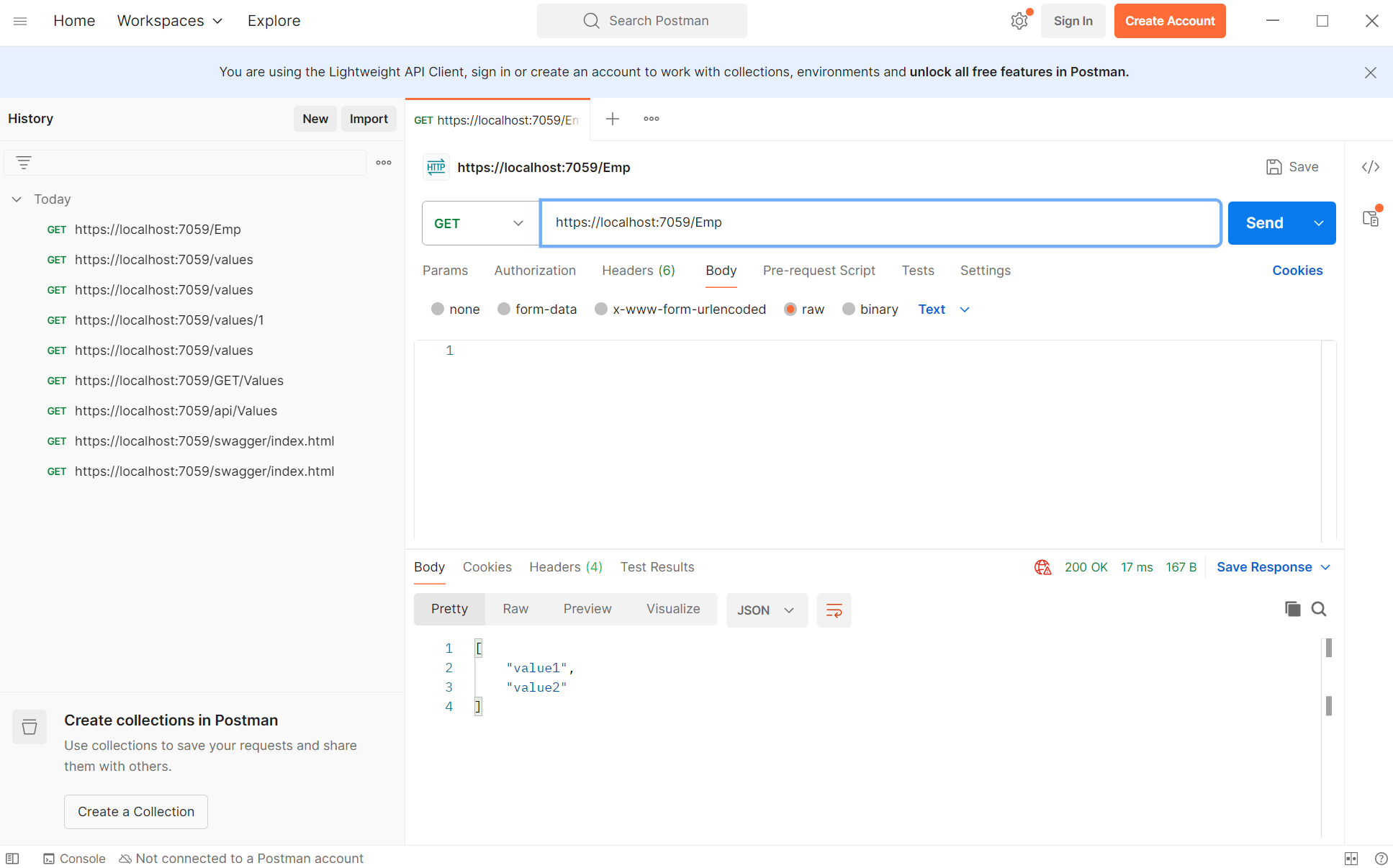
**Modified Route in ValuesController.cs:**

[Route("api/Emp")]

**Now accessible from:**

https://localhost:7059/api/Emp

Verified both in Swagger and Postman after the change



**3.Web Api using custom model class**

**1. Employee.cs Model Code**

public class Employee

{

public int Id { get; set; }

public string Name { get; set; }

public int Salary { get; set; }

public bool Permanent { get; set; }

public Department Department { get; set; }

public List<Skill> Skills { get; set; }

public DateTime DateOfBirth { get; set; }

}

**2. EmployeeController.cs Code**

[HttpGet]

[ProducesResponseType(StatusCodes.Status200OK)]

public ActionResult<List<Employee>> GetStandard()

{

throw new Exception("Simulated Exception");

}

[HttpPost]

public IActionResult PostEmployee([FromBody] Employee emp)

{

return Ok(emp);

}

**3. CustomAuthFilter.cs Code**

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

var hasAuth = context.HttpContext.Request.Headers.TryGetValue("Authorization", out var token);

if (!hasAuth)

{

context.Result = new BadRequestObjectResult("Invalid request - No Auth token");

return;

}

if (!token.ToString().Contains("Bearer"))

{

context.Result = new BadRequestObjectResult("Invalid request - Token present but Bearer unavailable");

return;

}

}

}

**4. CustomExceptionFilter.cs Code**

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

File.WriteAllText("error\_log.txt", context.Exception.ToString());

context.Result = new ObjectResult("An error occurred")

{

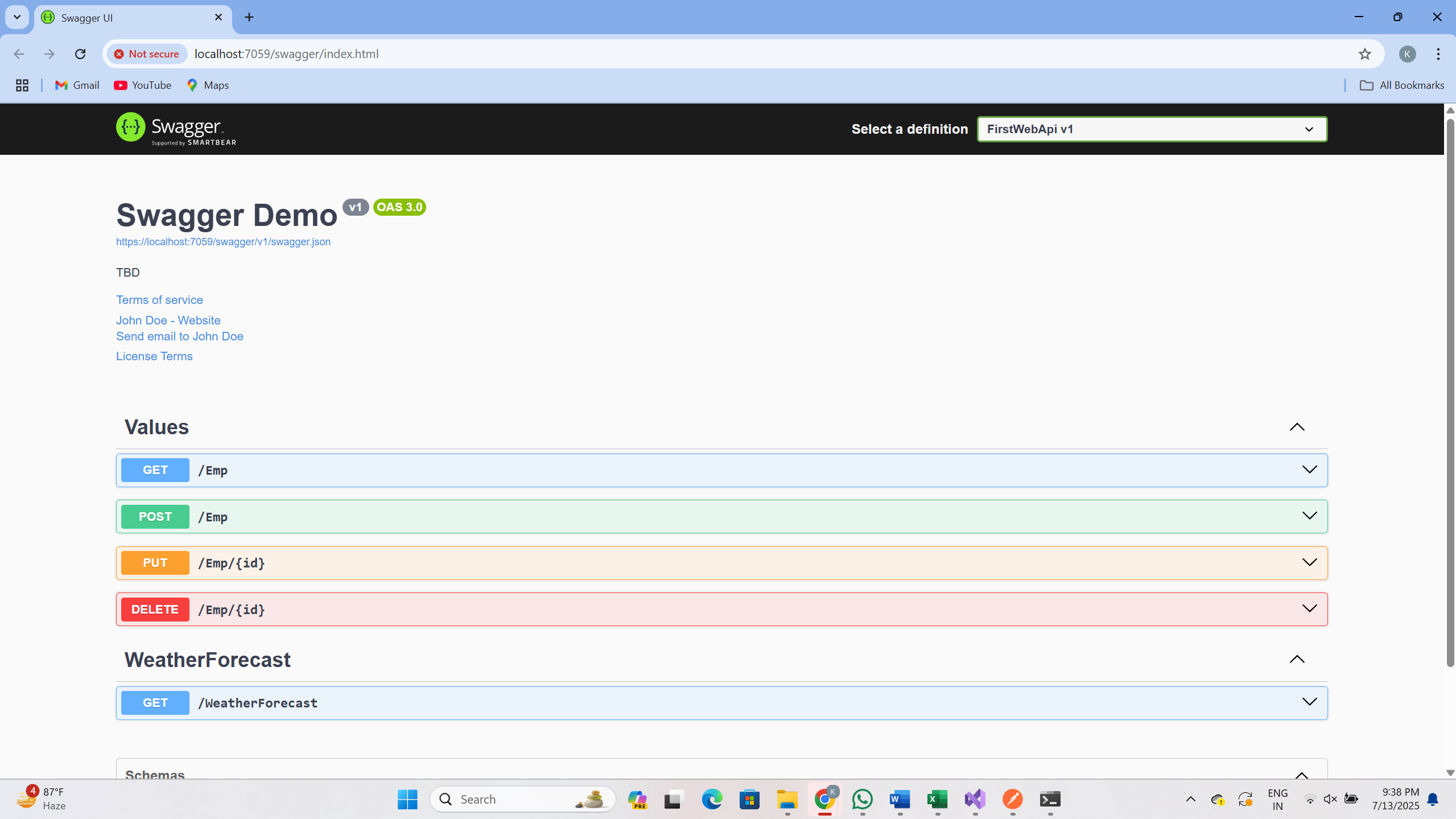
StatusCode = 500

};

}

}

**5. Swagger Output Screenshot**

****

**4.Web Api CRUD operation**

**Modify the Employee and Department Models**

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

}

public class Employee

{

public int Id { get; set; }

public string Name { get; set; }

public int Salary { get; set; }

public bool Permanent { get; set; }

public Department Department { get; set; }

public List<string> Skills { get; set; }

public DateTime DateOfBirth { get; set; }

}

**Modify EmployeeController.cs**

using Microsoft.AspNetCore.Mvc;

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

[ApiController]

public class EmployeeController : ControllerBase

{

private static List<Employee> employees = new List<Employee>

{

new Employee { Id = 1, Name = "John", Salary = 50000, Permanent = true, Department = new Department { Id = 1, Name = "HR" }, Skills = new List<string> { "C#", "SQL" }, DateOfBirth = new DateTime(1995, 5, 21) },

new Employee { Id = 2, Name = "Jane", Salary = 60000, Permanent = false, Department = new Department { Id = 2, Name = "Finance" }, Skills = new List<string> { "Java", "Angular" }, DateOfBirth = new DateTime(1992, 10, 12) }

};

[HttpPut("{id}")]

public ActionResult<Employee> UpdateEmployee(int id, [FromBody] Employee updatedEmployee)

{

if (id <= 0)

{

return BadRequest("Invalid employee id");

}

var emp = employees.FirstOrDefault(e => e.Id == id);

if (emp == null)

{

return BadRequest("Invalid employee id");

}

emp.Name = updatedEmployee.Name;

emp.Salary = updatedEmployee.Salary;

emp.Permanent = updatedEmployee.Permanent;

emp.Department = updatedEmployee.Department;

emp.Skills = updatedEmployee.Skills;

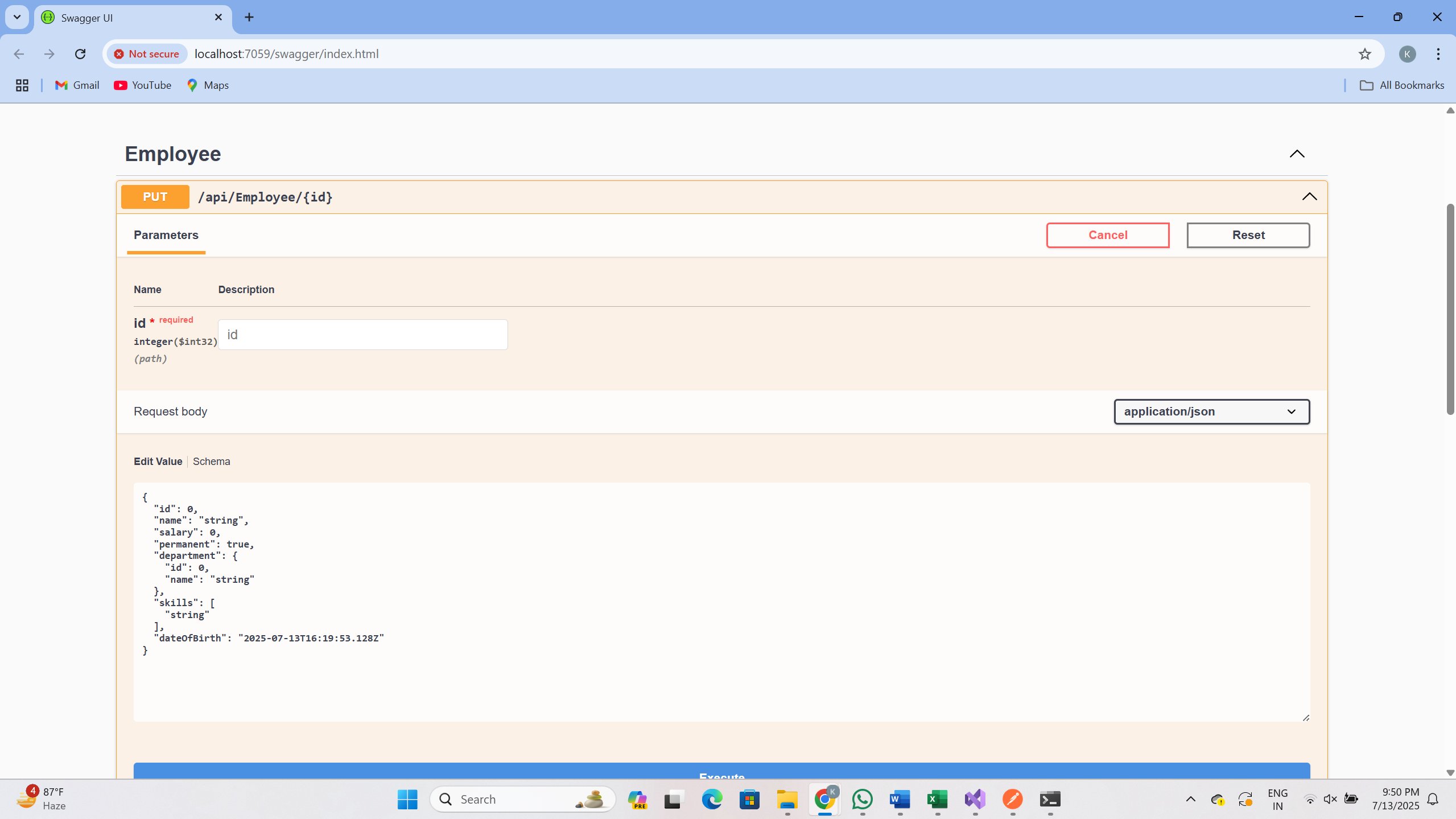
emp.DateOfBirth = updatedEmployee.DateOfBirth;

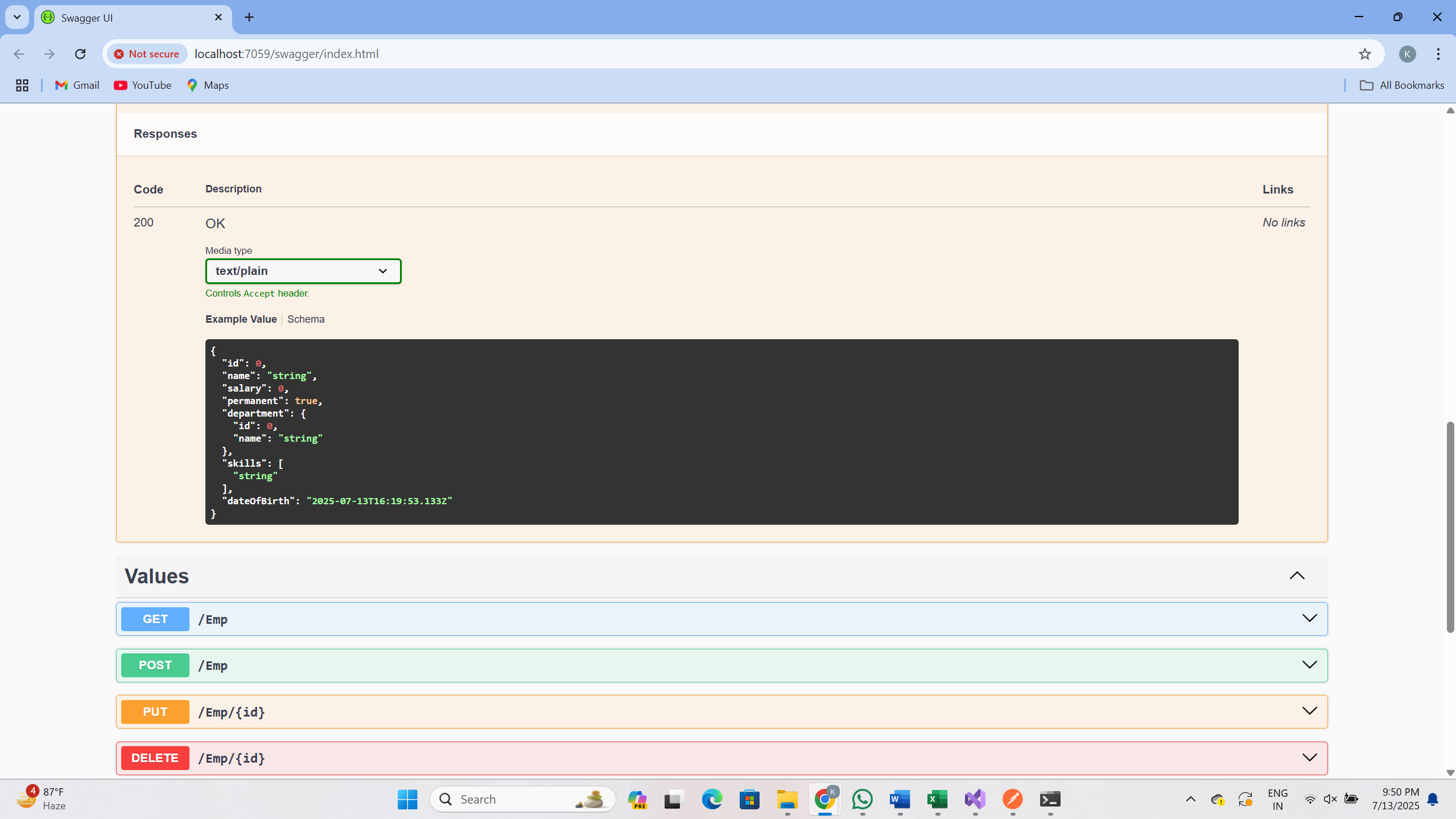
return Ok(emp);

}

}

**Test in Swagger:OUTPUT**





**5. Securing Web API using JWT and Enabling CORS**

**1. Enable JWT Authentication in Program.cs**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

// JWT Key

string securityKey = "mysuperdupersecretkeymysuperdupersecretkey";

var symmetricSecurityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(securityKey));

// Add JWT Authentication

builder.Services.AddAuthentication(x =>

{

x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

}).AddJwtBearer(x =>

{

x.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = "mySystem",

ValidAudience = "myUsers",

IssuerSigningKey = symmetricSecurityKey

};

});

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

app.UseSwagger();

app.UseSwaggerUI();

app.UseHttpsRedirection();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

**2. AuthController.cs to Generate JWT Token**

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

[ApiController]

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

[AllowAnonymous]

public class AuthController : ControllerBase

{

[HttpGet("generate-token")]

public IActionResult GenerateToken()

{

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretkeymysuperdupersecretkey"));

var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);

var claims = new[]

{

new Claim(ClaimTypes.Role, "Admin"),

new Claim("UserId", "101")

};

var token = new JwtSecurityToken(

issuer: "mySystem",

audience: "myUsers",

claims: claims,

expires: DateTime.Now.AddMinutes(10),

signingCredentials: credentials);

return Ok(new { token = new JwtSecurityTokenHandler().WriteToken(token) });

}

}

**3. Secure EmployeeController using JWT Roles**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

[ApiController]

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

[Authorize(Roles = "Admin")]

public class EmployeeController : ControllerBase

{

[HttpGet]

public IActionResult GetEmployee()

{

return Ok(new { Id = 1, Name = "John Doe", Role = "Admin" });

}

}