1. WebApi\_Handson

STEP 1: Understand RESTful Web Services, Web API & Microservices

### What is a RESTful Web Service?

* A **RESTful service** is a way for different software systems to communicate using the **HTTP protocol** (like how you open websites).
* It uses **URLs (web addresses)** to access resources like data (user info, products, etc.).
* It is based on **REST architecture** (more below ).

Features of REST Architecture:

| **Feature** | **Explanation (Simple)** |
| --- | --- |
| **Representational State Transfer (REST)** | The service gives you the data you ask for (like JSON or XML) |
| **Stateless** | The server doesn’t remember anything between requests |
| **Messages** | All data is sent using **HTTP requests and responses** |
| **Microservice Concept** | Break big app into small parts (services) that do one thing (e.g., user service, order service) |
| **WebService vs WebAPI** | WebService uses XML and SOAP. WebAPI is simpler and uses JSON + HTTP |
| **Not limited to XML** | Unlike old services, WebAPI can send **JSON**, **XML**, or **anything** |

|  |  |
| --- | --- |

STEP 2: What is HttpRequest & HttpResponse?

### HttpRequest

* It is like **you sending a letter** (asking for info).
* Example: “Give me all users” → GET /api/users

### HttpResponse

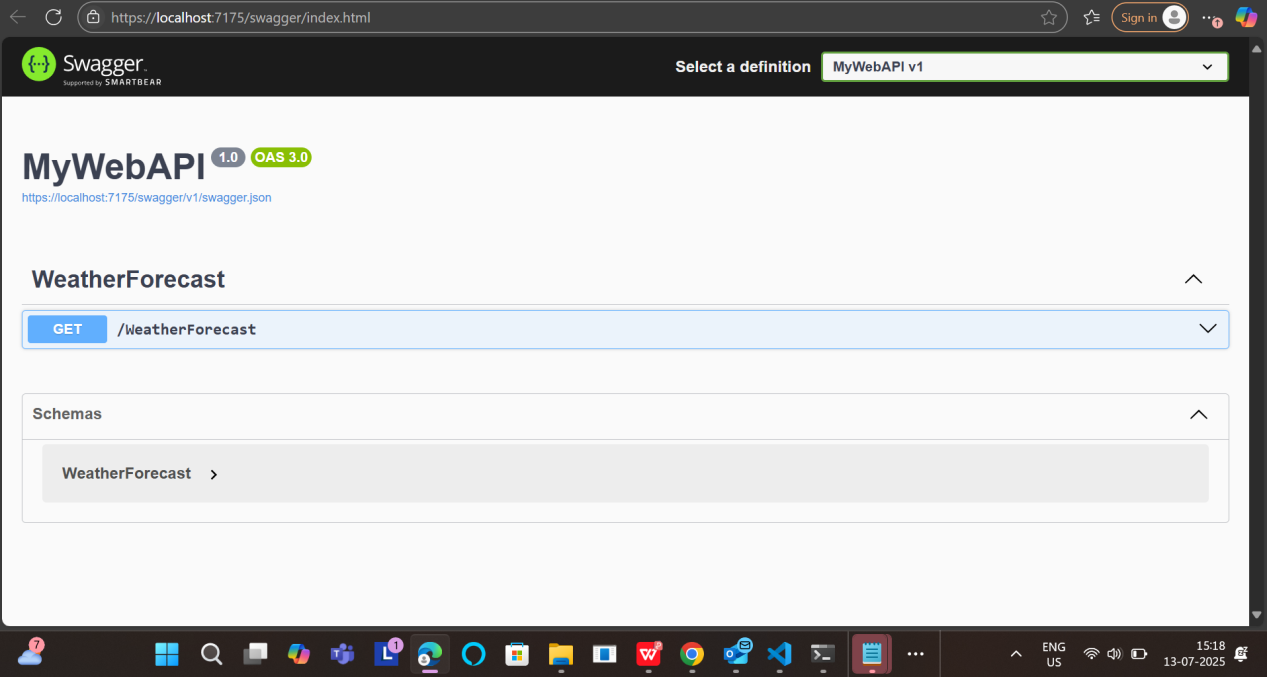
* It is like **getting the reply** (the answer).
* Example: Server says: “Here are 5 users” (in JSON).

STEP 3: What are Action Verbs?

These are the 4 **main verbs** you use when building Web APIs:

| **Verb** | **What it means** | **C# Attribute** |
| --- | --- | --- |
| **GET** | Ask for data | [HttpGet] |
| **POST** | Add new data | [HttpPost] |
| **PUT** | Update data | [HttpPut] |
| **DELETE** | Remove data | [HttpDelete] |

STEP 5: Create First Web API Project in Visual Studio



STEP 6: Modify/Add Write (POST) Method

Replaced the code with this simple version:

using Microsoft.AspNetCore.Mvc;

namespace MyFirstWebAPI.Controllers

{

[ApiController]

[Route("[controller]")]

public class WeatherForecastController : ControllerBase

{

// GET

[HttpGet]

public IActionResult Get()

{

return Ok("This is a GET request");

}

// POST

[HttpPost]

public IActionResult Post()

{

return Ok("This is a POST request");

}

// PUT

[HttpPut]

public IActionResult Put()

{

return Ok("This is a PUT request");

}

// DELETE

[HttpDelete]

public IActionResult Delete()

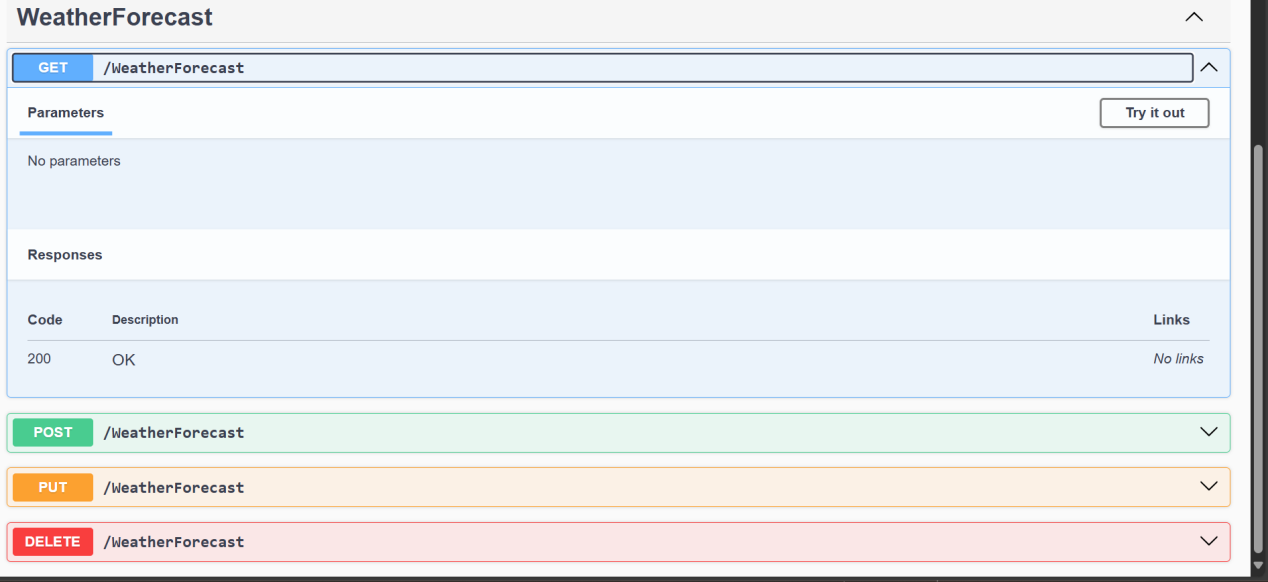
{

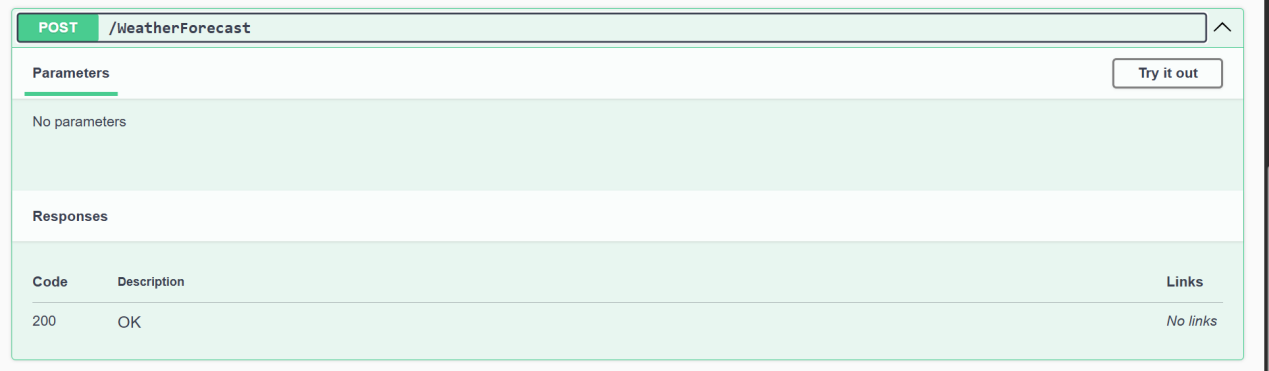
return Ok("This is a DELETE request");

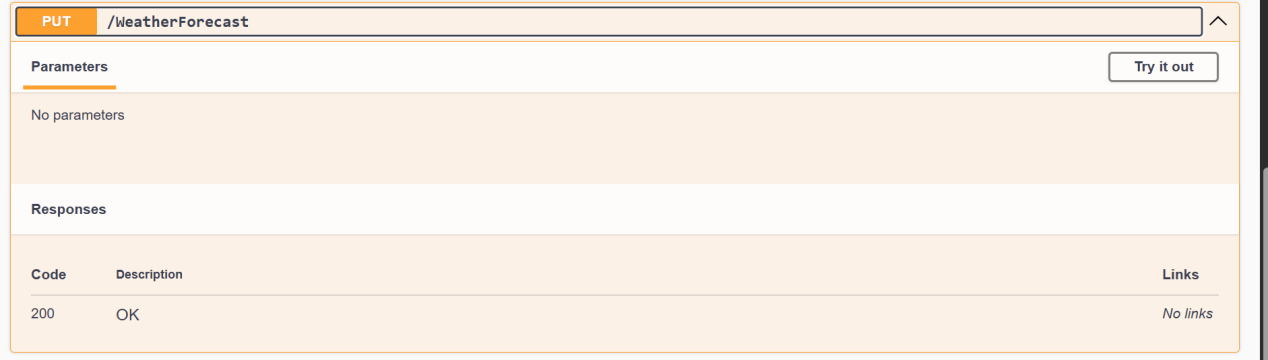
}

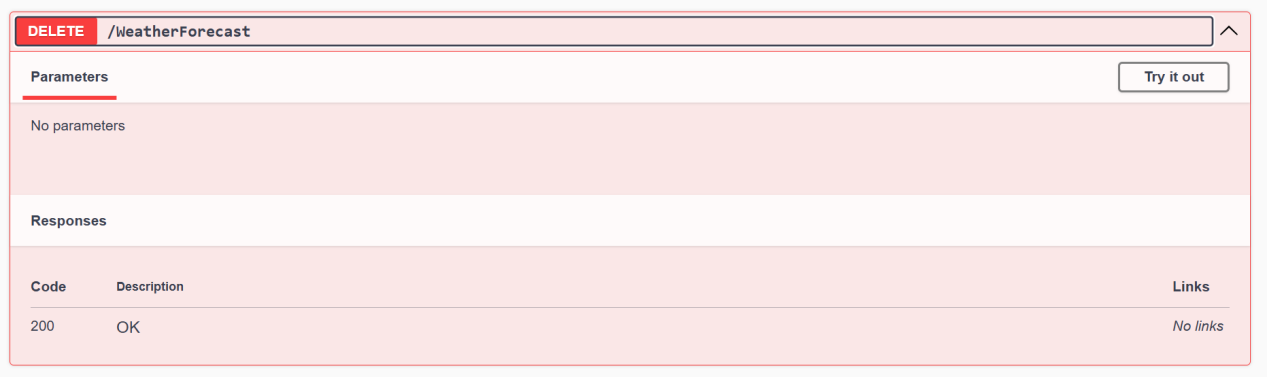
}

}









STEP 7: Learn About Configuration Files

| **File** | **Purpose** |
| --- | --- |
| **Startup.cs** | Registers services, sets middleware (used in .NET 5 and below) |
| **Program.cs** | Used in .NET 6+ (combined with Startup) |
| **appSettings.json** | Like settings.json — stores connection strings, keys |
| **launchSettings.json** | Sets launch URL, profiles when you debug |
|  |  |
|  |  |

1. WebApi\_Handson

PART 1: Add Swagger to our Web API in .NET Core

Program.cs

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "TBD",

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

Contact = new OpenApiContact

{

Name = "John Doe",

Email = "john@xyzmail.com",

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

},

License = new OpenApiLicense

{

Name = "License Terms",

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

}

});

});

var app = builder.Build();

// Configure the HTTP request pipeline.

app.UseSwagger();

app.UseSwaggerUI(c =>

{

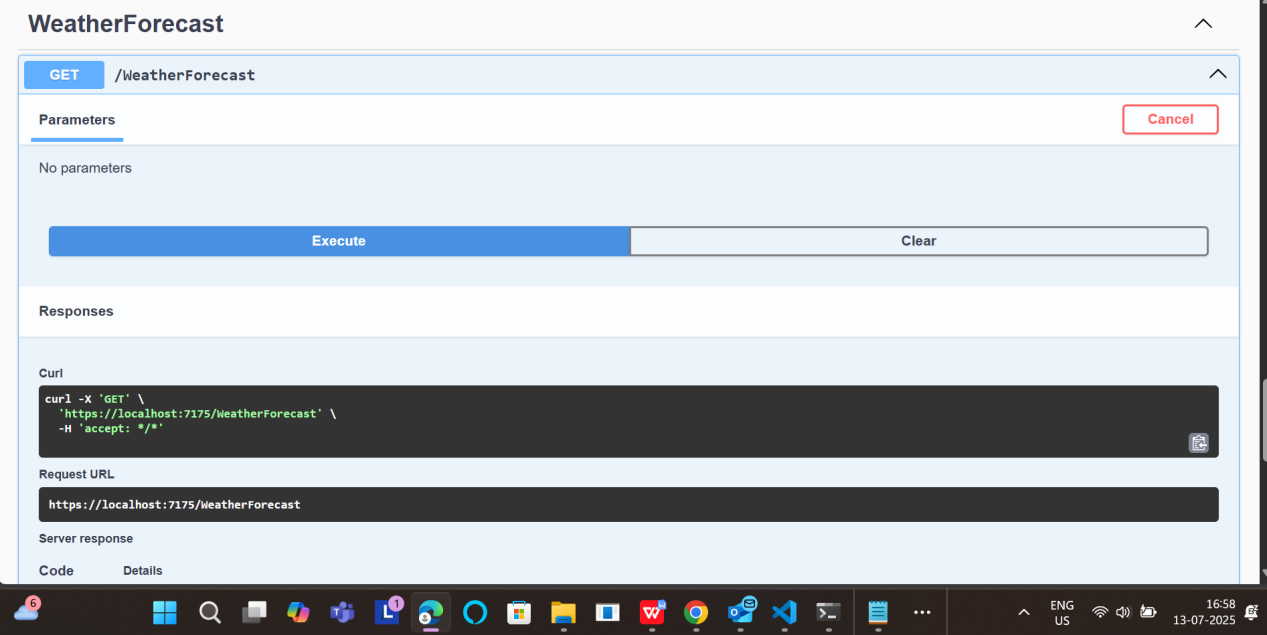
c.SwaggerEndpoint("/swagger/v1/swagger.json", "Swagger Demo");

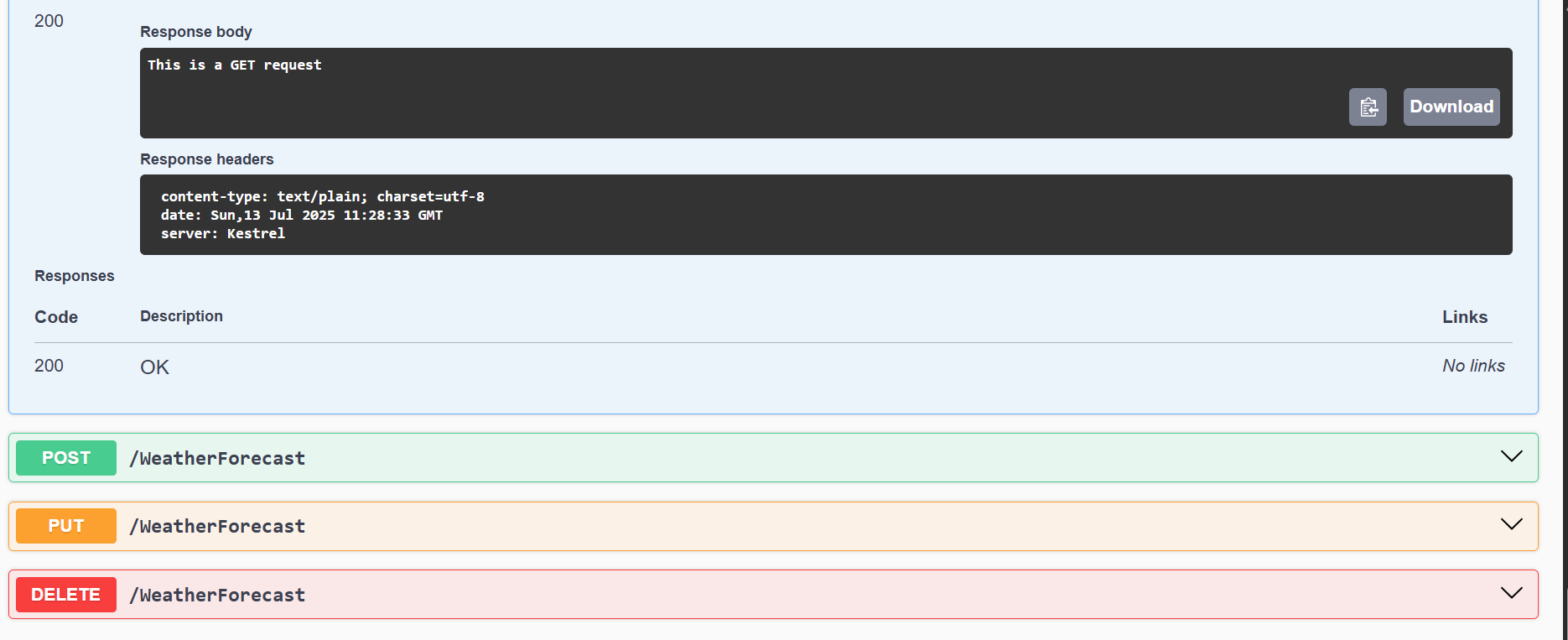
});

app.UseAuthorization();

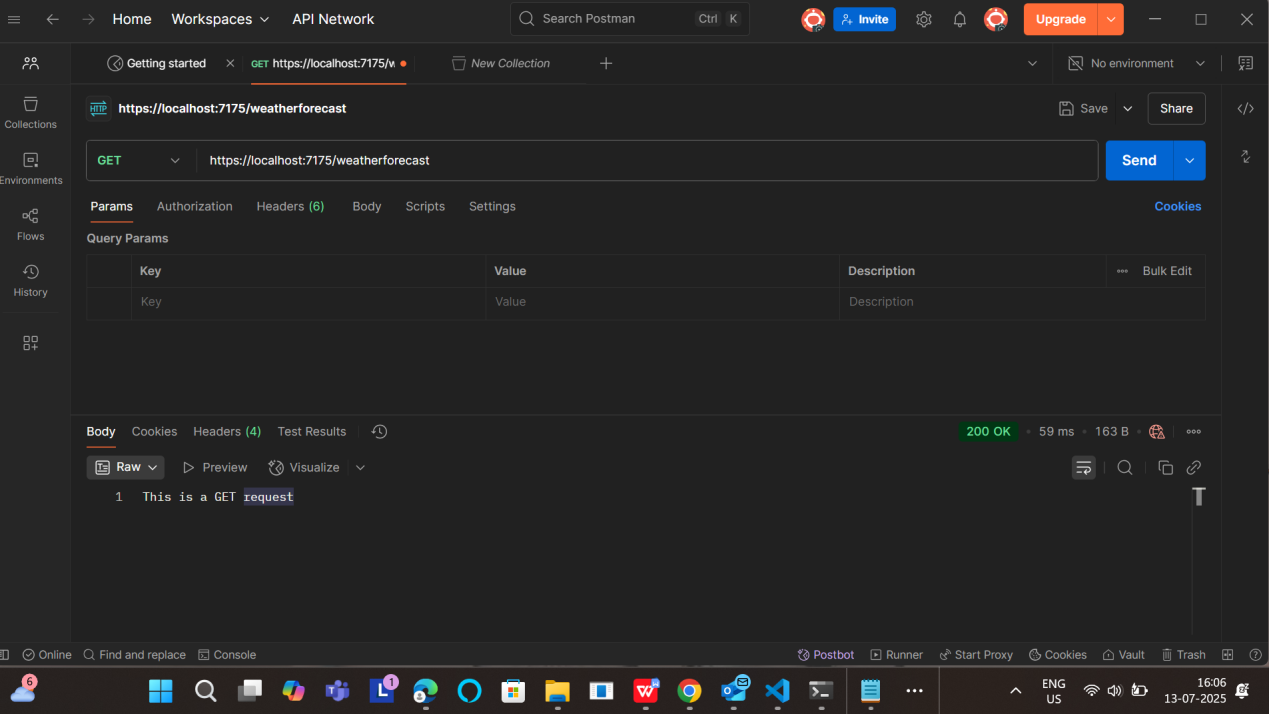
app.MapControllers();

app.Run();





Test our API Using POSTMAN



Modify Route and Use ActionName

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

namespace YourProjectName.Controllers

{

[Route("Emp/[action]")]

[ApiController]

public class EmployeeController : ControllerBase

{

// GET: Emp/GetAll

[HttpGet]

[ActionName("GetAll")]

[ProducesResponseType(StatusCodes.Status200OK)]

public IActionResult GetAll()

{

// Simulated employee list

var employees = new[] { "John", "Jane", "Alice" };

return Ok(employees);

}

// GET: Emp/GetSingle

[HttpGet]

[ActionName("GetSingle")]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status404NotFound)]

public IActionResult GetSingle()

{

var employee = "John";

return Ok(employee);

}

// POST: Emp/Add

[HttpPost]

[ActionName("Add")]

[ProducesResponseType(StatusCodes.Status201Created)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

public IActionResult AddEmployee([FromBody] string newEmployee)

{

if (string.IsNullOrEmpty(newEmployee))

{

return BadRequest("Employee name is required");

}

return Created("", $"Employee '{newEmployee}' added successfully!");

}

// PUT: Emp/Update

[HttpPut]

[ActionName("Update")]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status404NotFound)]

public IActionResult UpdateEmployee([FromBody] string updatedEmployee)

{

return Ok($"Employee '{updatedEmployee}' updated successfully!");

}

// DELETE: Emp/Delete

[HttpDelete]

[ActionName("Delete")]

[ProducesResponseType(StatusCodes.Status200OK)]

public IActionResult DeleteEmployee([FromQuery] string name)

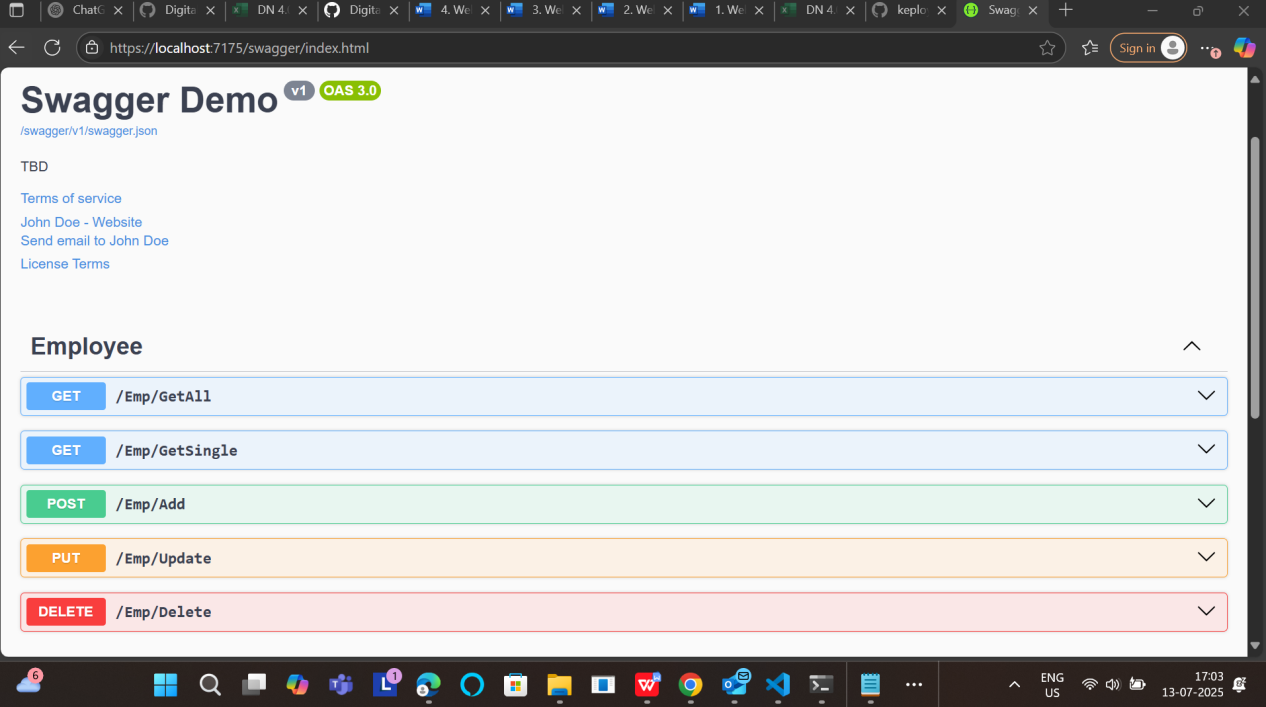
{

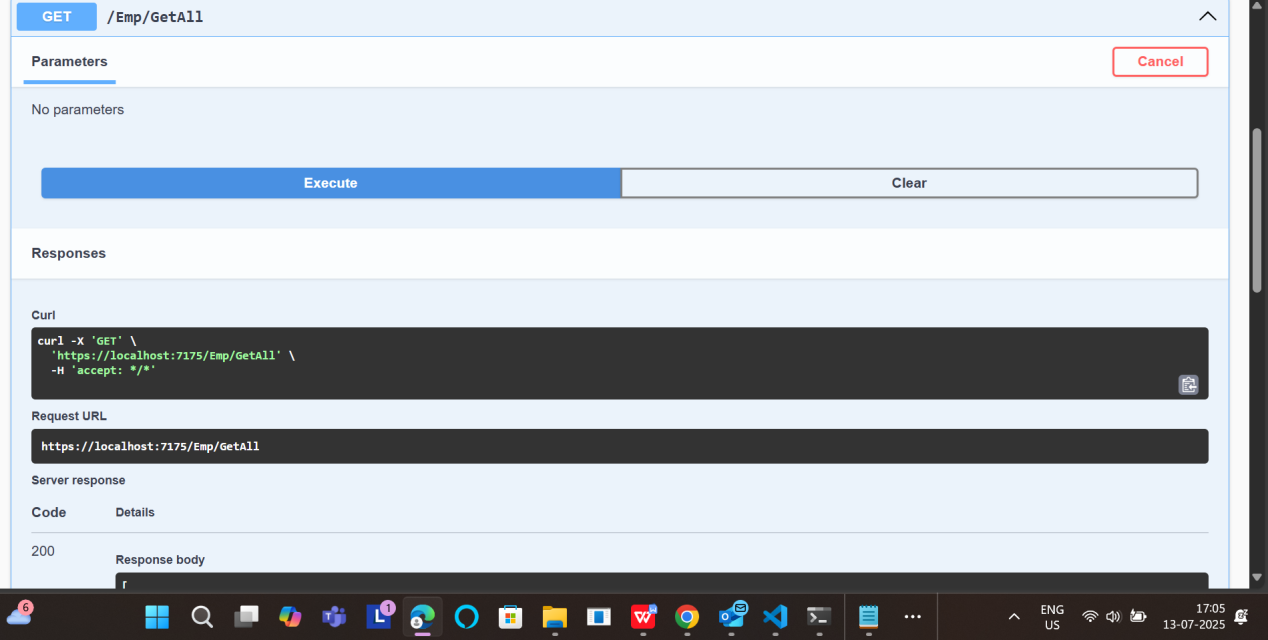
return Ok($"Employee '{name}' deleted successfully!");

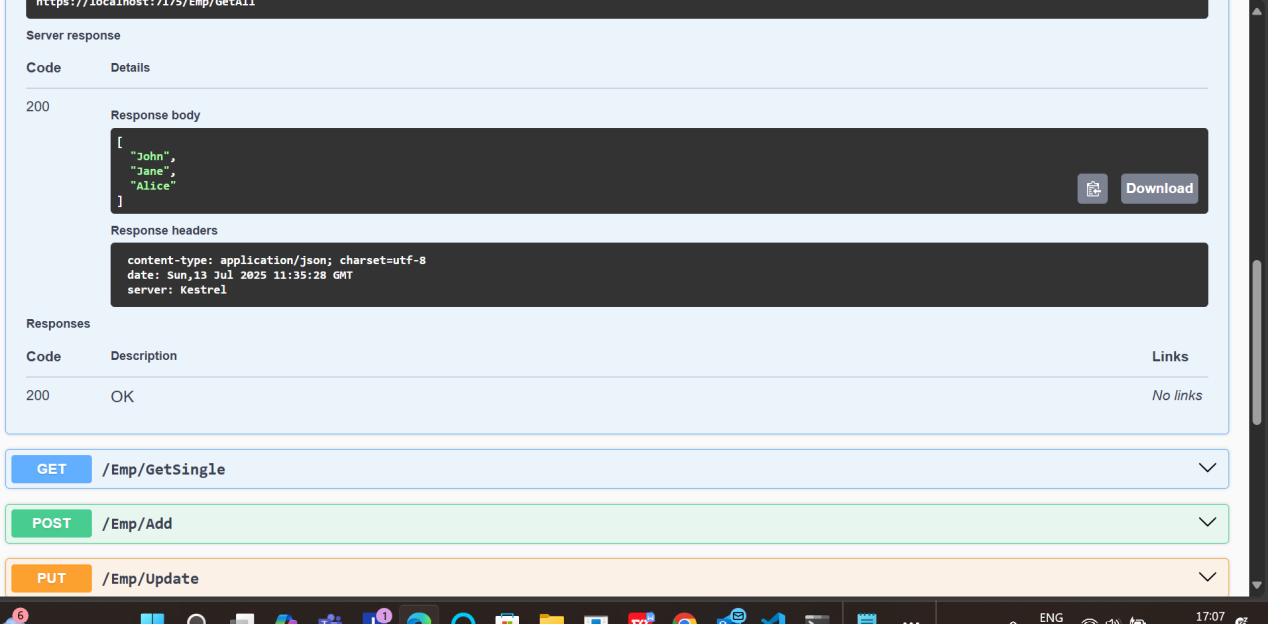
}

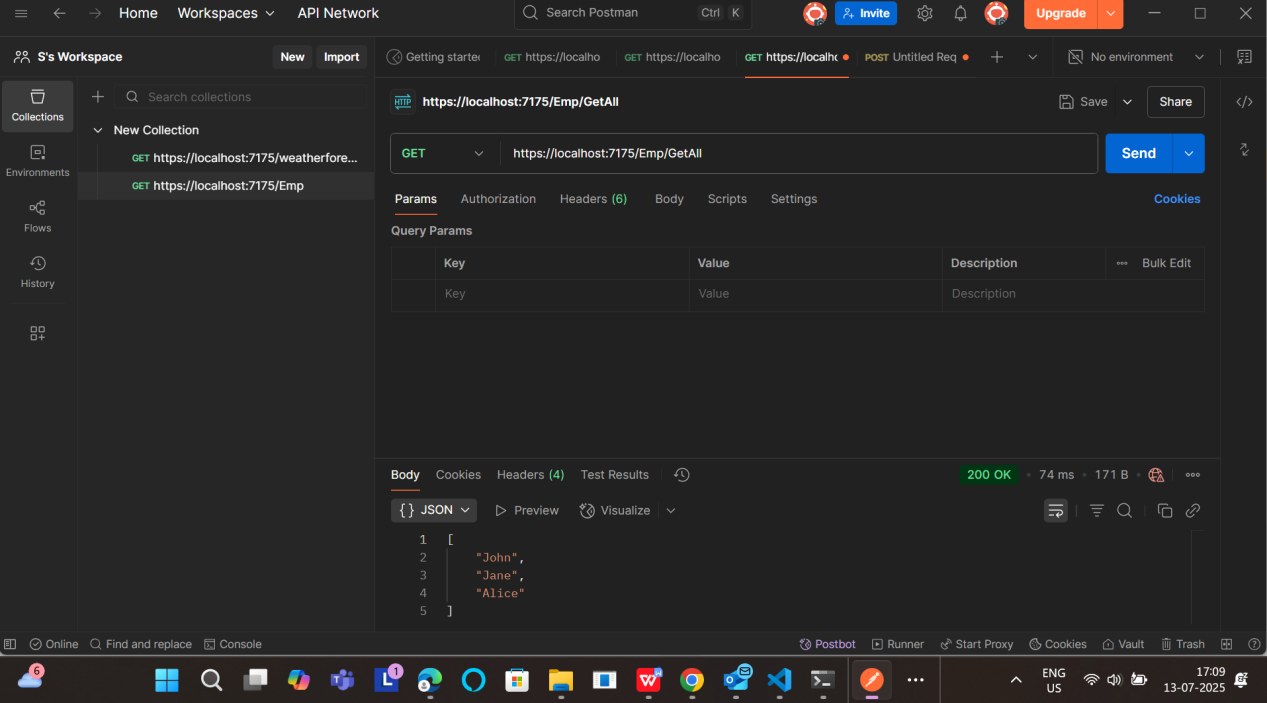
}

}









1. WebApi\_Handson

Add Model Classes

Create a folder Models, then add a file Employee.cs

using System;

using System.Collections.Generic;

namespace YourProjectName.Models

{

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; }

}

public class Department

{

public int Id { get; set; }

public string? Name { get; set; }

}

public class Skill

{

public int Id { get; set; }

public string? Name { get; set; }

}

}

EmployeeController.cs

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

using System;

using EmployeeApiDemo.Models;

using EmployeeApiDemo.Filters;

using Microsoft.AspNetCore.Authorization;

namespace EmployeeApiDemo.Controllers

{

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

[ApiController]

[ServiceFilter(typeof(CustomAuthFilter))] // Authorization filter applied

public class EmployeeController : ControllerBase

{

private static List<Employee>? employees;

public EmployeeController()

{

if (employees == null)

{

employees = GetStandardEmployeeList();

}

}

[HttpGet]

[AllowAnonymous]

[ProducesResponseType(200)]

public ActionResult<List<Employee>> GetStandard()

{

return Ok(employees);

}

[HttpPost]

public IActionResult AddEmployee([FromBody] Employee emp)

{

employees!.Add(emp);

return Ok("Employee added!");

}

[HttpGet("cause-exception")]

[ProducesResponseType(500)]

public IActionResult CauseException()

{

throw new Exception("Test exception");

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John",

Salary = 10000,

Permanent = true,

DateOfBirth = new DateTime(1990, 5, 23),

Department = new Department { Id = 1, Name = "HR" },

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "C#" },

new Skill { Id = 2, Name = "ASP.NET" }

}

}

};

}

}

}

Create a folder Filters, then add CustomAuthFilter.cs:

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace EmployeeApiDemo.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

if (!context.HttpContext.Request.Headers.ContainsKey("Authorization"))

{

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

return;

}

var token = context.HttpContext.Request.Headers["Authorization"].ToString();

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

{

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

return;

}

base.OnActionExecuting(context);

}

}

}

Add Custom Exception Filter

Add another file in Filters: CustomExceptionFilter.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace EmployeeApiDemo.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

string logPath = Path.Combine(Directory.GetCurrentDirectory(), "logs.txt");

File.AppendAllText(logPath, $"Exception: {context.Exception.Message}\n");

context.Result = new ObjectResult("Internal Server Error. Please try again later.")

{

StatusCode = 500

};

}

}

}

Update Program.cs

using Microsoft.OpenApi.Models;

using EmployeeApiDemo.Filters;

var builder = WebApplication.CreateBuilder(args);

// Add services

builder.Services.AddControllers(options =>

{

options.Filters.Add(typeof(CustomExceptionFilter)); // Global exception handler

});

builder.Services.AddScoped<CustomAuthFilter>();

builder.Services.AddScoped<CustomExceptionFilter>();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "Employee API with Custom Filters",

Contact = new OpenApiContact

{

Name = "John Doe",

Email = "john@example.com",

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

}

});

});

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

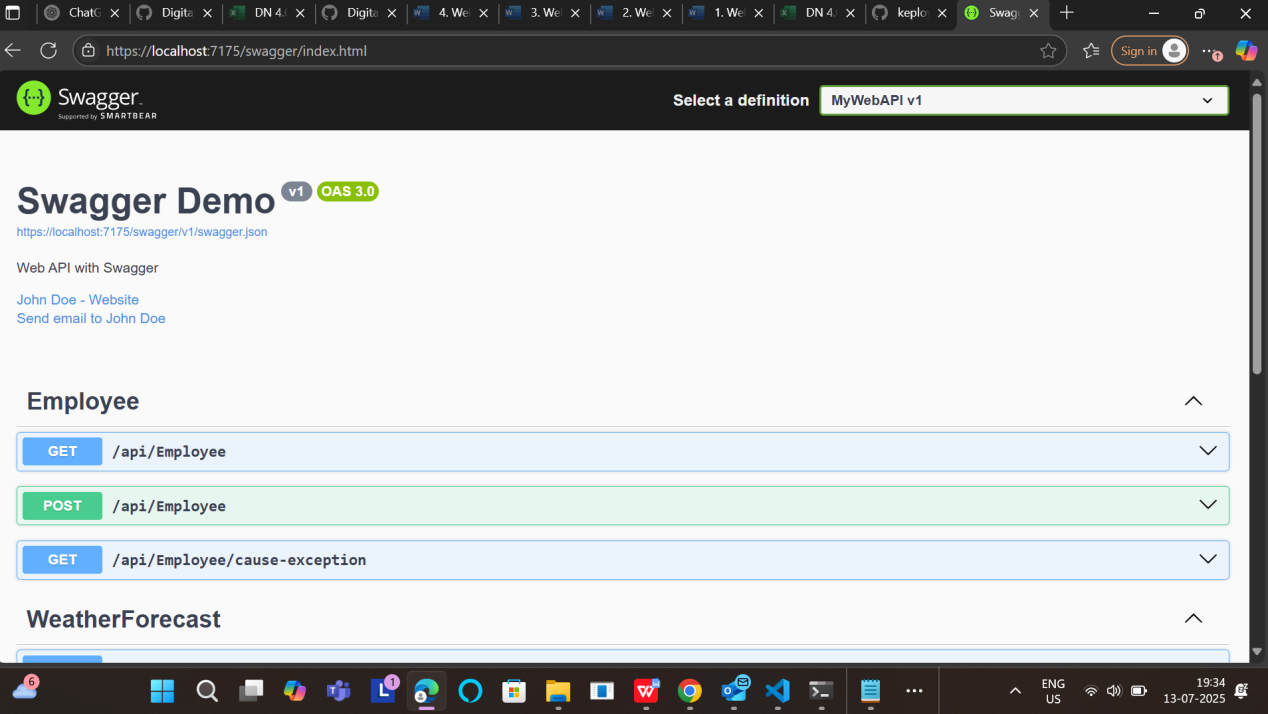
}

app.UseAuthorization();

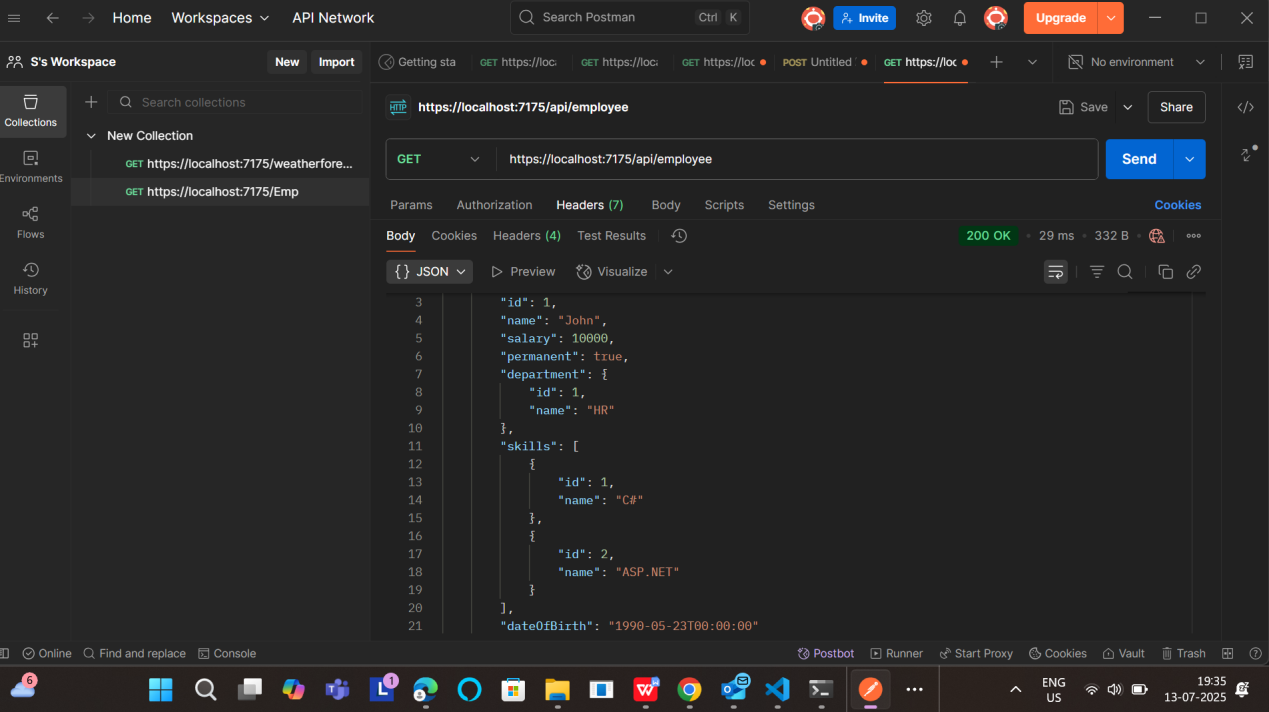
app.MapControllers();

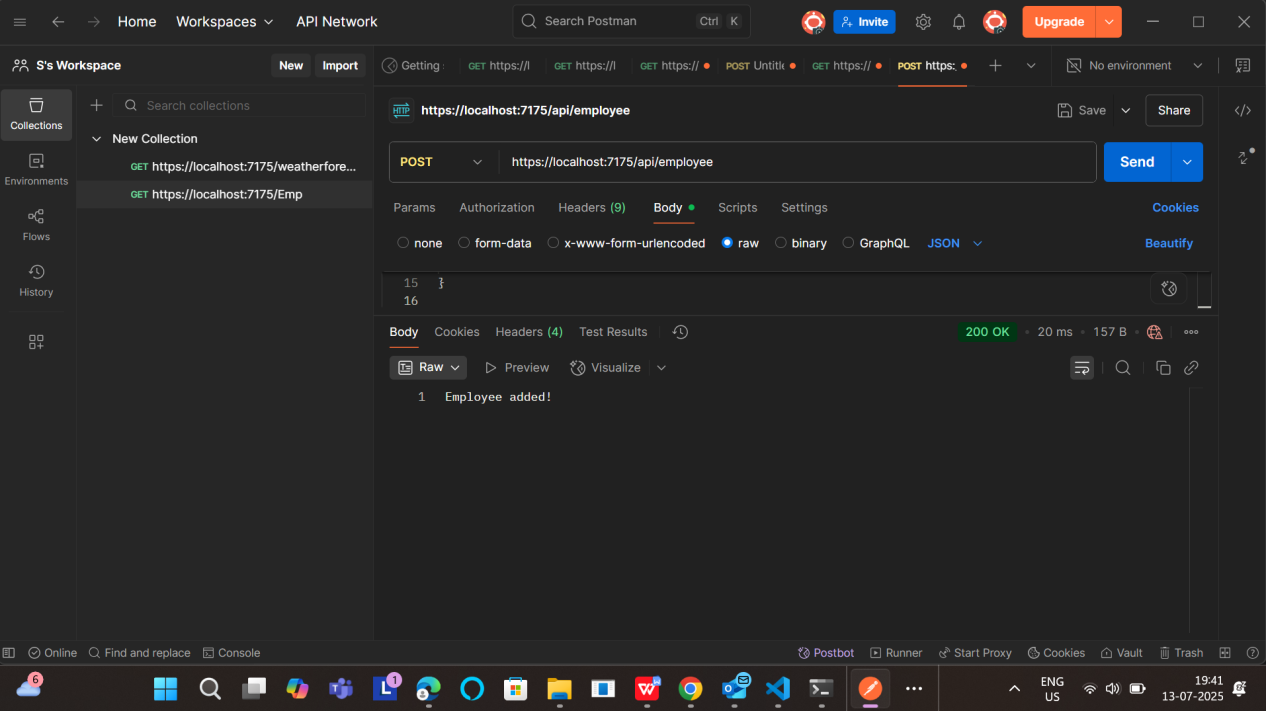
app.Run();

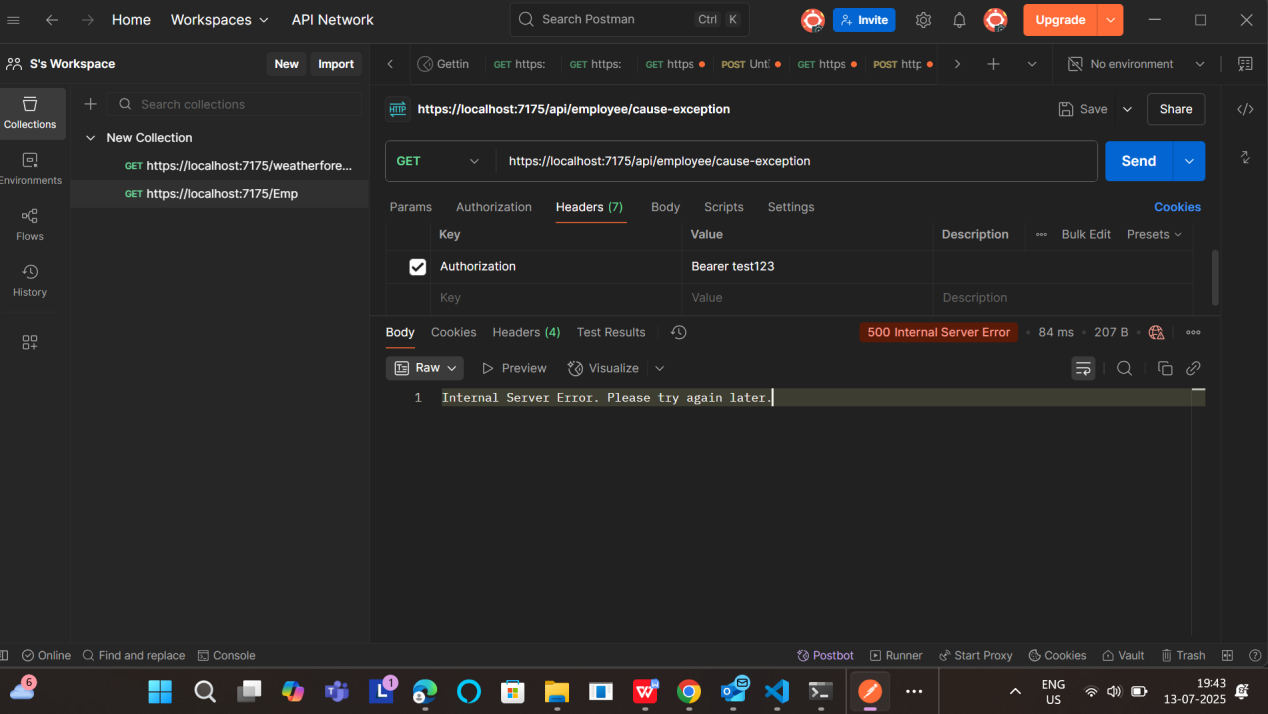
Test in Swagger



Test in Postman







1. WebApi\_Handson

EmployeeController.cs

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

using System;

using System.Linq;

using EmployeeApiDemo.Models;

using EmployeeApiDemo.Filters;

using Microsoft.AspNetCore.Authorization;

namespace MyWebAPI.Controllers

{

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

[ApiController]

[ServiceFilter(typeof(CustomAuthFilter))] // Authorization filter

public class EmployeeController : ControllerBase

{

private static List<Employee>? employees;

public EmployeeController()

{

if (employees == null)

{

employees = GetStandardEmployeeList();

}

}

[HttpGet]

[AllowAnonymous]

[ProducesResponseType(200)]

public ActionResult<List<Employee>> GetStandard()

{

return Ok(employees);

}

[HttpPost]

public IActionResult AddEmployee([FromBody] Employee emp)

{

employees!.Add(emp);

return Ok("Employee added!");

}

[HttpPut]

public ActionResult<Employee> UpdateEmployee([FromBody] Employee emp)

{

if (emp.Id <= 0)

{

return BadRequest("Invalid employee id");

}

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

if (existingEmp == null)

{

return BadRequest("Invalid employee id");

}

// Update the employee fields

existingEmp.Name = emp.Name;

existingEmp.Salary = emp.Salary;

existingEmp.Permanent = emp.Permanent;

existingEmp.Department = emp.Department;

existingEmp.Skills = emp.Skills;

existingEmp.DateOfBirth = emp.DateOfBirth;

return Ok(existingEmp);

}

[HttpGet("cause-exception")]

[ProducesResponseType(500)]

public IActionResult CauseException()

{

throw new Exception("Test exception");

}

// Private method to return hardcoded employees

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John",

Salary = 10000,

Permanent = true,

DateOfBirth = new DateTime(1990, 5, 23),

Department = new Department { Id = 1, Name = "HR" },

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "C#" },

new Skill { Id = 2, Name = "ASP.NET" }

}

}

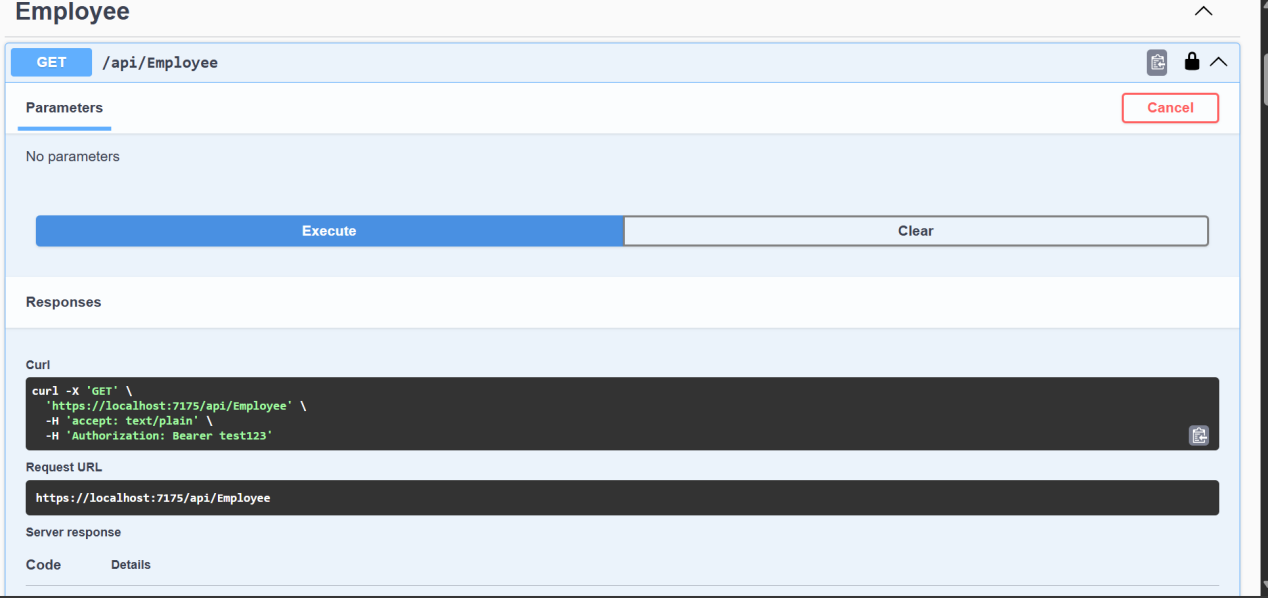
};

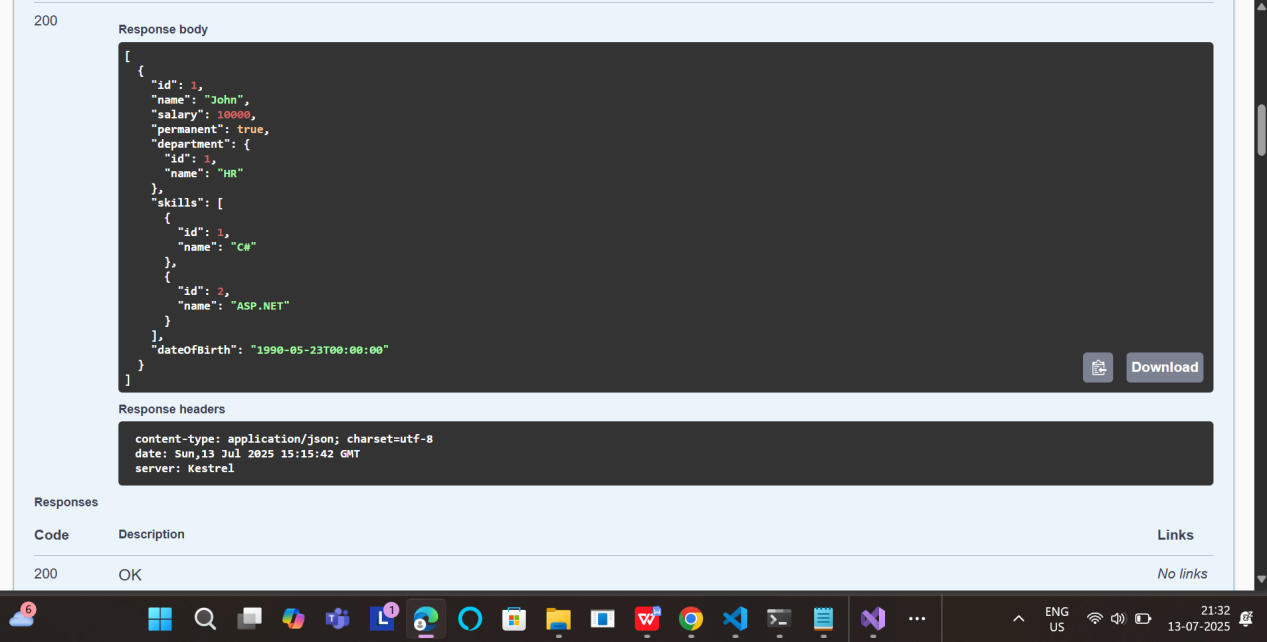
}

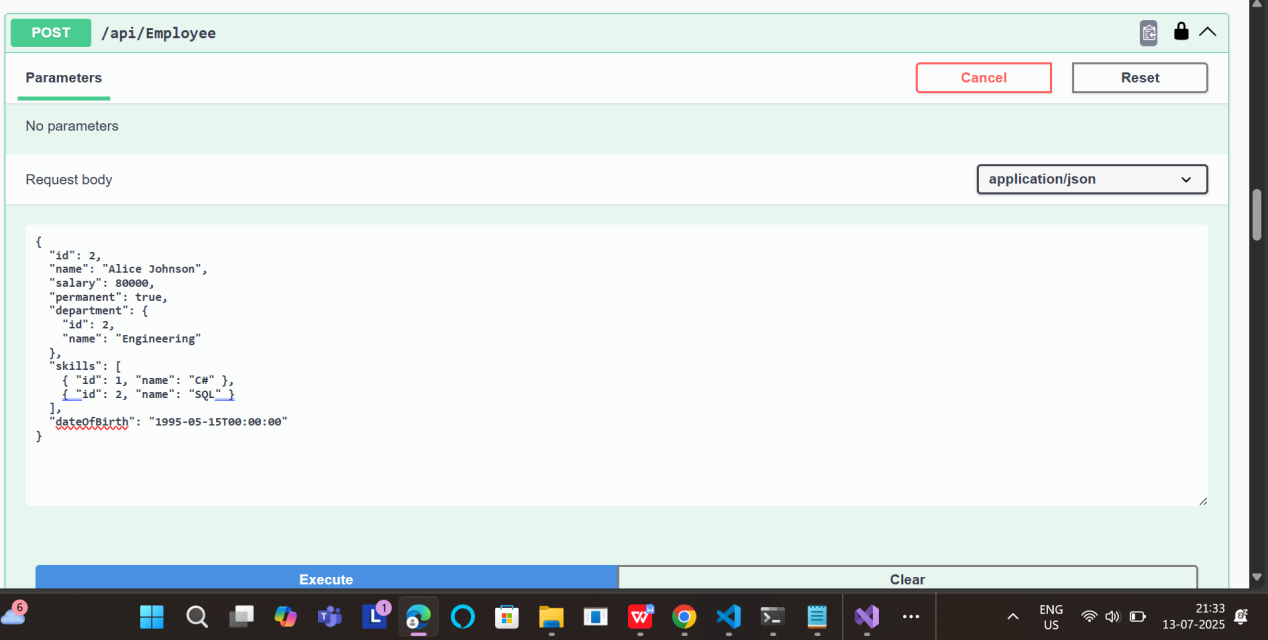
}

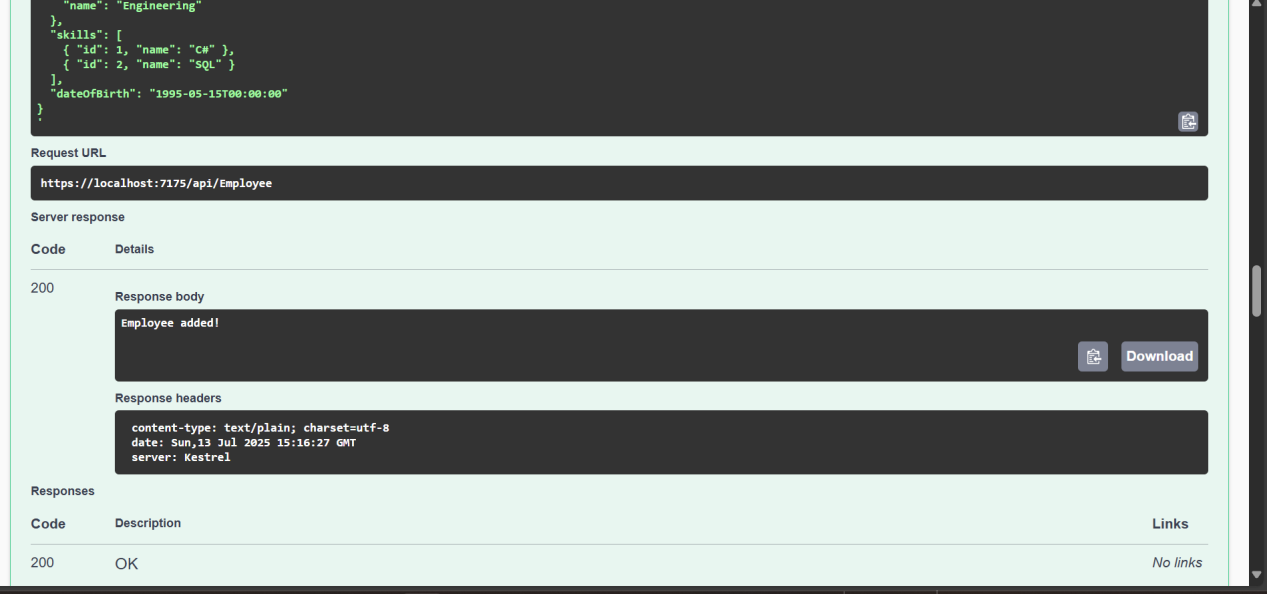
}

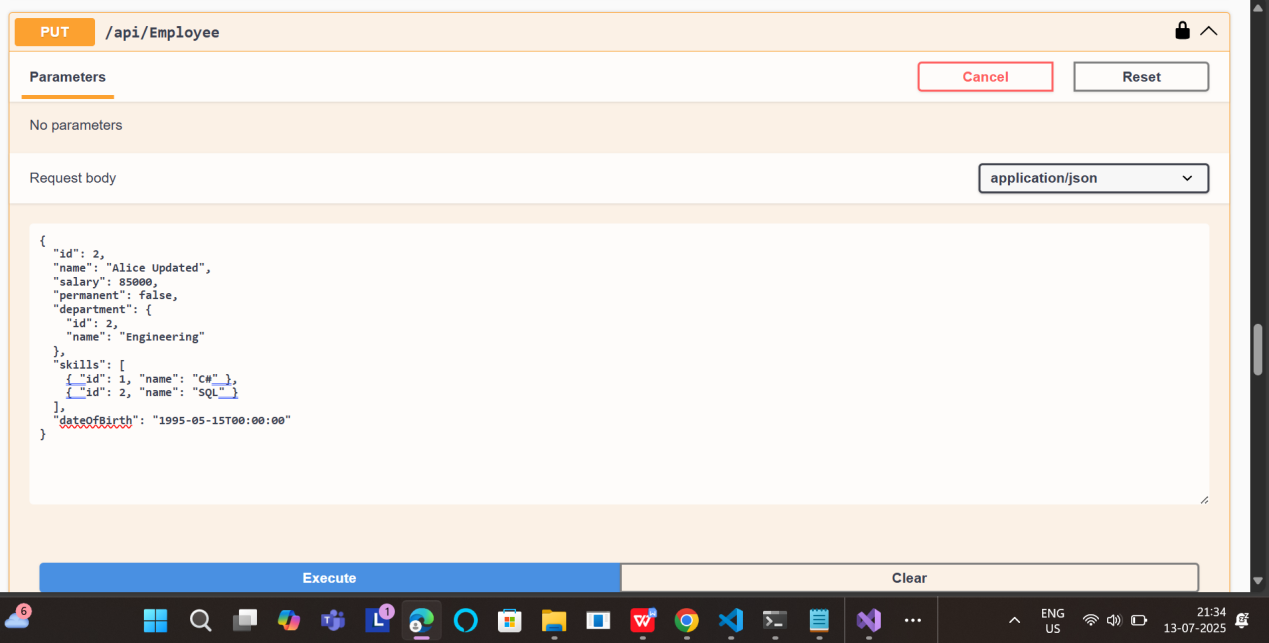
Test in Swagger

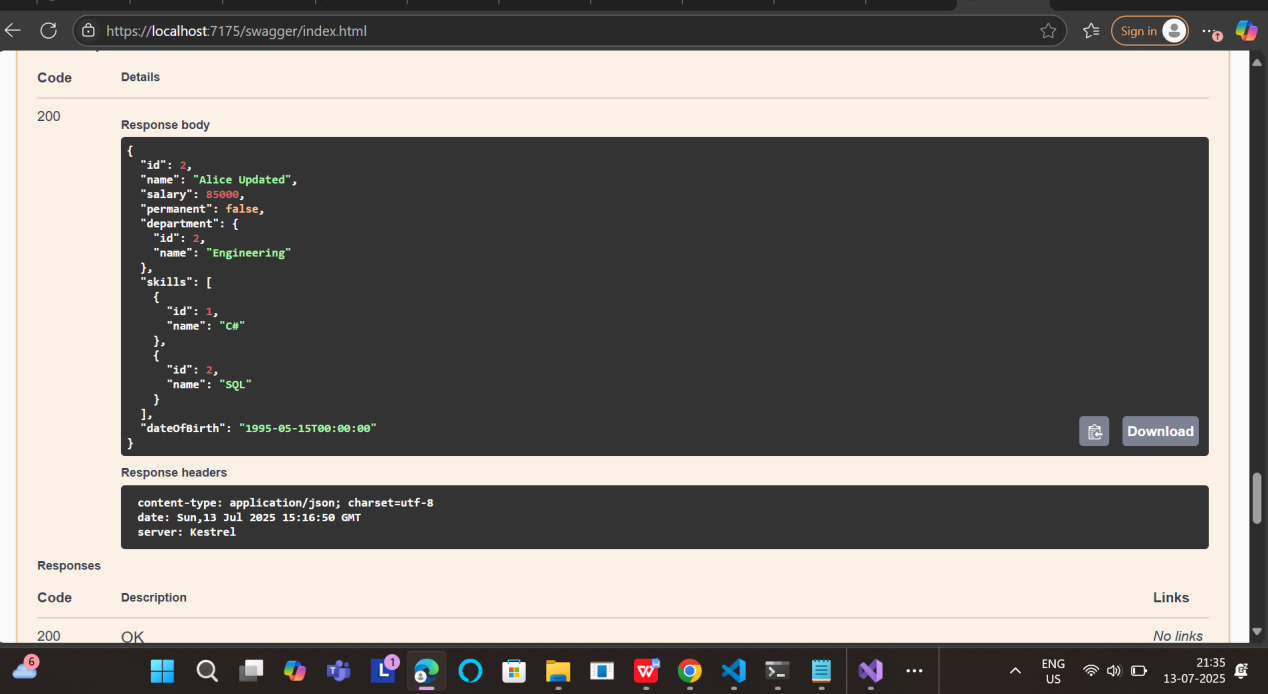




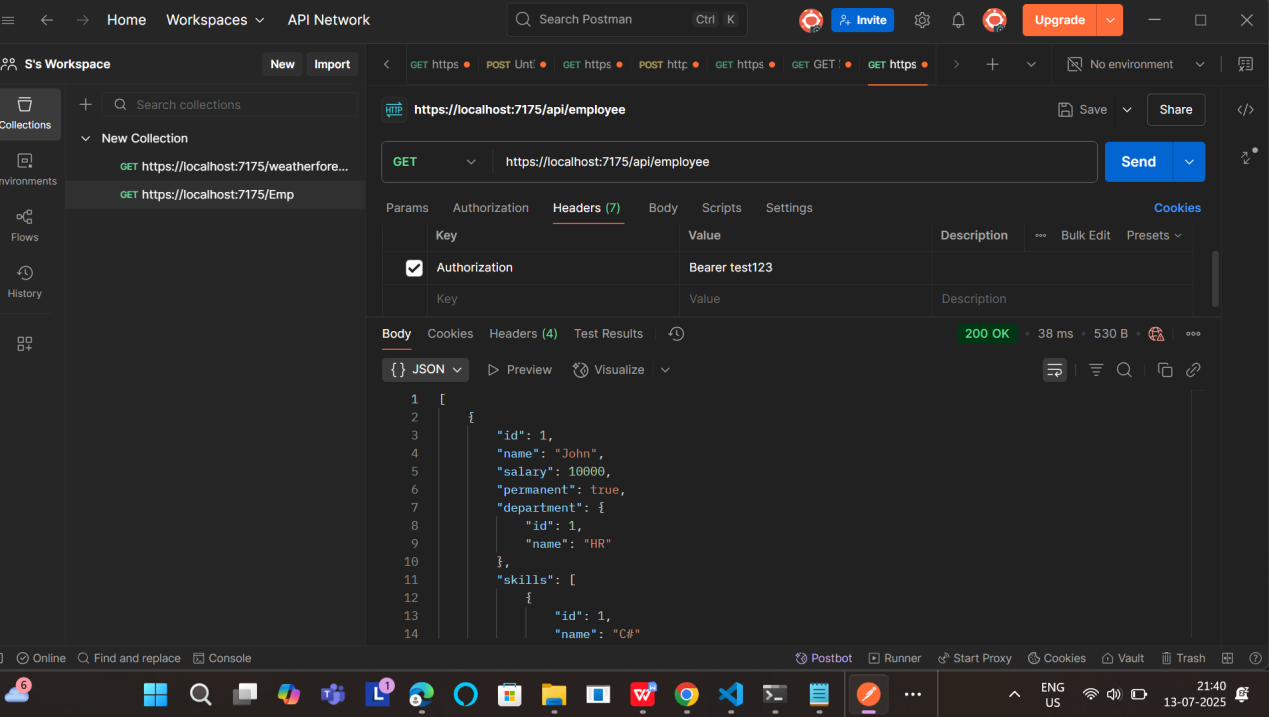


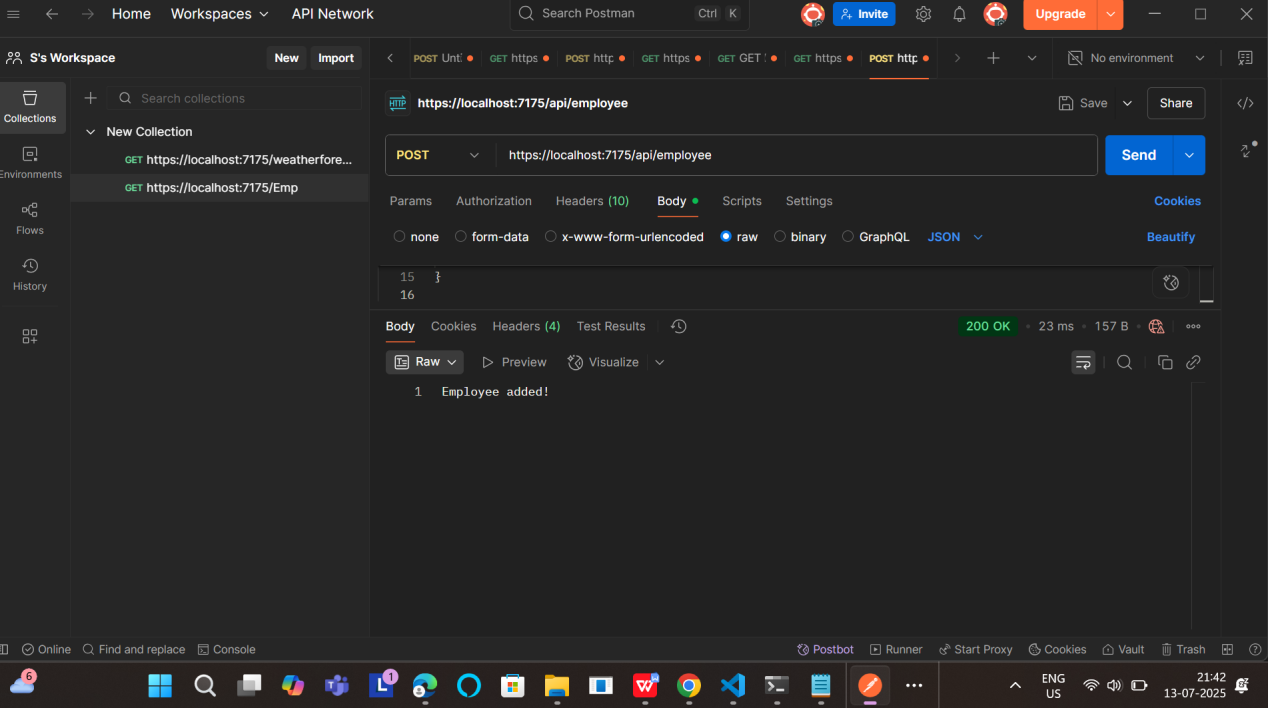


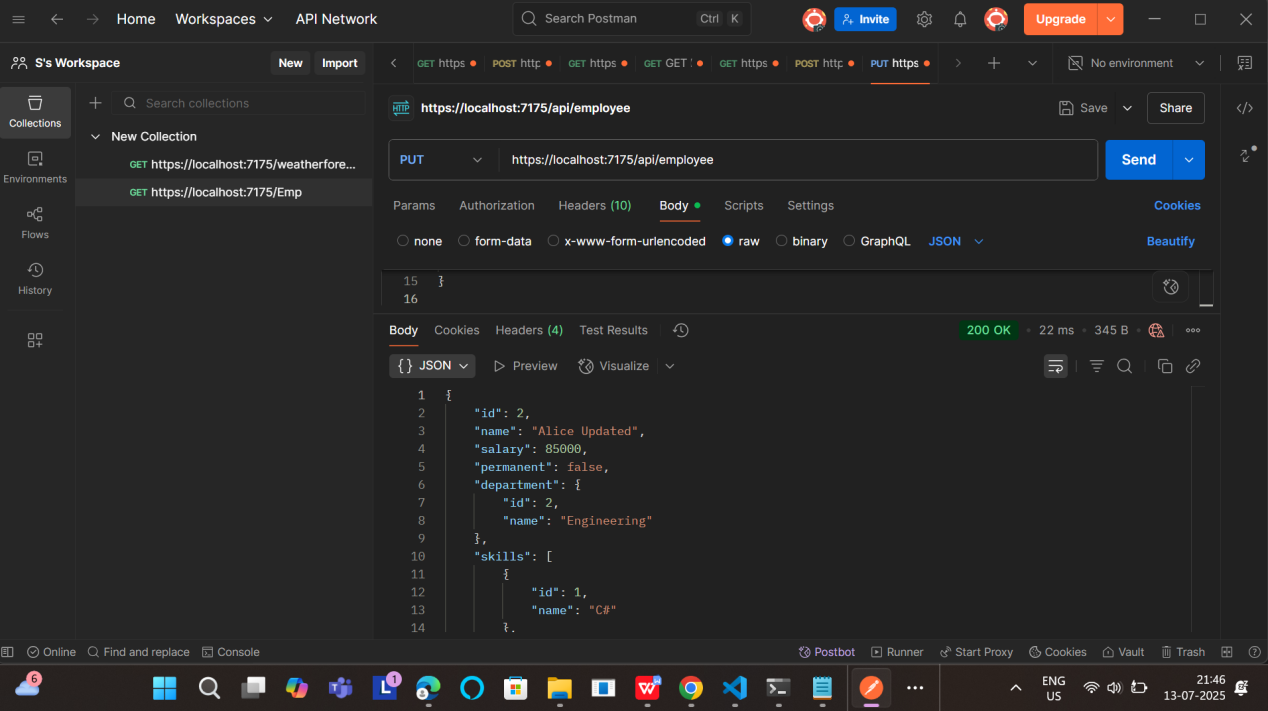




Testing using PostMan







Add DELETE /api/employee/{id}

EmployeeController.cs

using Microsoft.AspNetCore.Mvc;

using MyWebApi.Models;

using MyWebApi.Filters;

using Microsoft.AspNetCore.Authorization;

namespace MyWebApi.Controllers

{

[ApiController]

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

[ServiceFilter(typeof(CustomAuthFilter))] // Apply custom auth filter

public class EmployeeController : ControllerBase

{

// In-memory employee list

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

{

new Employee

{

Id = 1,

Name = "John Doe",

Salary = 50000,

Permanent = true,

Department = new Department { Id = 1, Name = "HR" },

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "Communication" },

new Skill { Id = 2, Name = "Management" }

},

DateOfBirth = new DateTime(1990, 5, 10)

}

};

[HttpGet]

[AllowAnonymous]

[ProducesResponseType(200)]

public ActionResult<List<Employee>> GetStandard()

{

return Ok(employees);

}

[HttpPost]

[ProducesResponseType(200)]

[ProducesResponseType(400)]

public IActionResult AddEmployee([FromBody] Employee emp)

{

if (emp == null || emp.Id <= 0)

return BadRequest("Invalid employee data");

employees.Add(emp);

return Ok("Employee added!");

}

[HttpPut]

[ProducesResponseType(200)]

[ProducesResponseType(400)]

public IActionResult UpdateEmployee([FromBody] Employee emp)

{

if (emp == null || emp.Id <= 0)

return BadRequest("Invalid employee id");

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

if (existing == null)

return BadRequest("Invalid employee id");

existing.Name = emp.Name;

existing.Salary = emp.Salary;

existing.Permanent = emp.Permanent;

existing.Department = emp.Department;

existing.Skills = emp.Skills;

existing.DateOfBirth = emp.DateOfBirth;

return Ok(existing);

}

[HttpDelete("{id}")]

[ProducesResponseType(200)]

[ProducesResponseType(400)]

public IActionResult Delete(int id)

{

if (id <= 0)

return BadRequest("Invalid employee id");

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

if (employee == null)

return BadRequest("Employee not found");

employees.Remove(employee);

return Ok($"Employee with ID {id} deleted");

}

[HttpGet("cause-exception")]

[ProducesResponseType(500)]

public IActionResult CauseException()

{

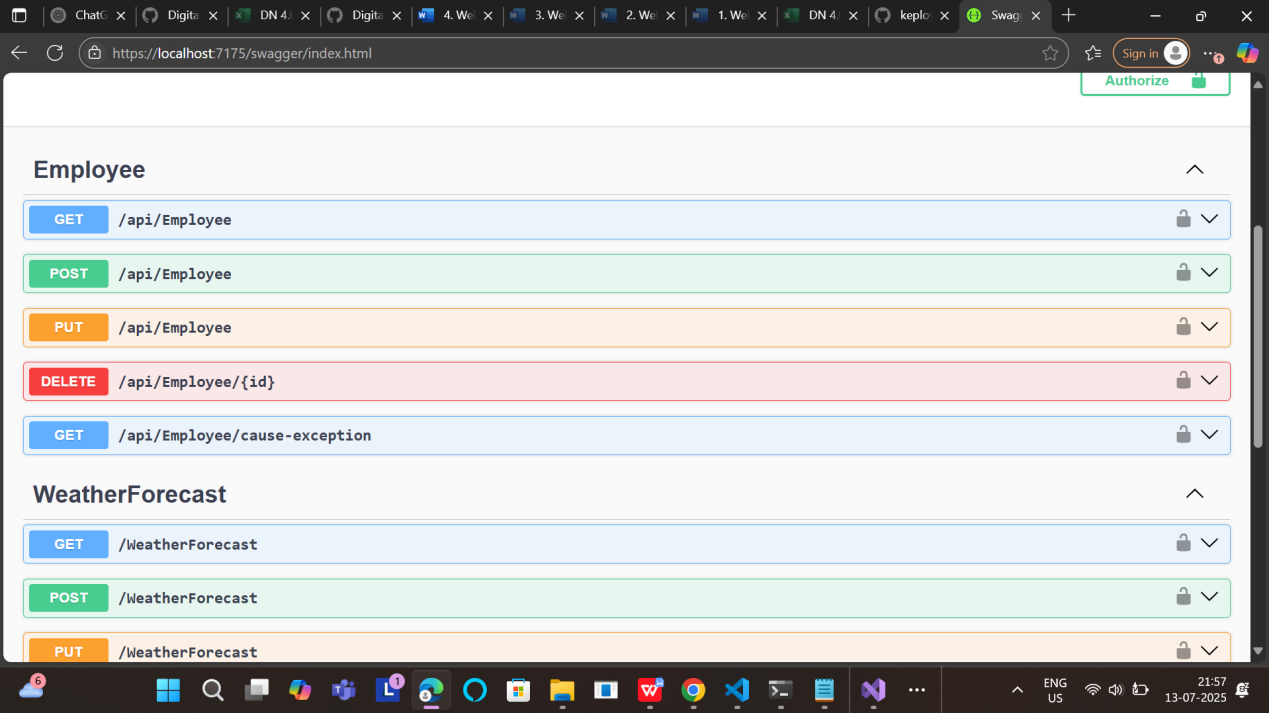
throw new Exception("Test exception");

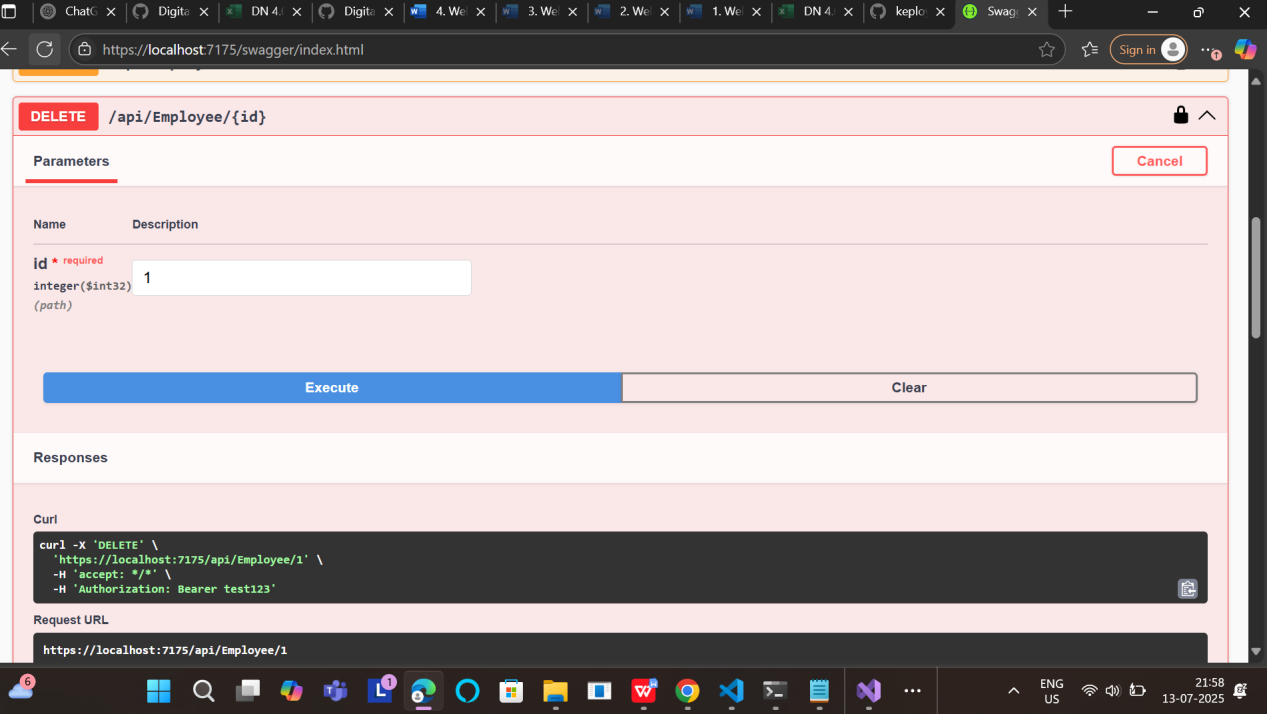
}

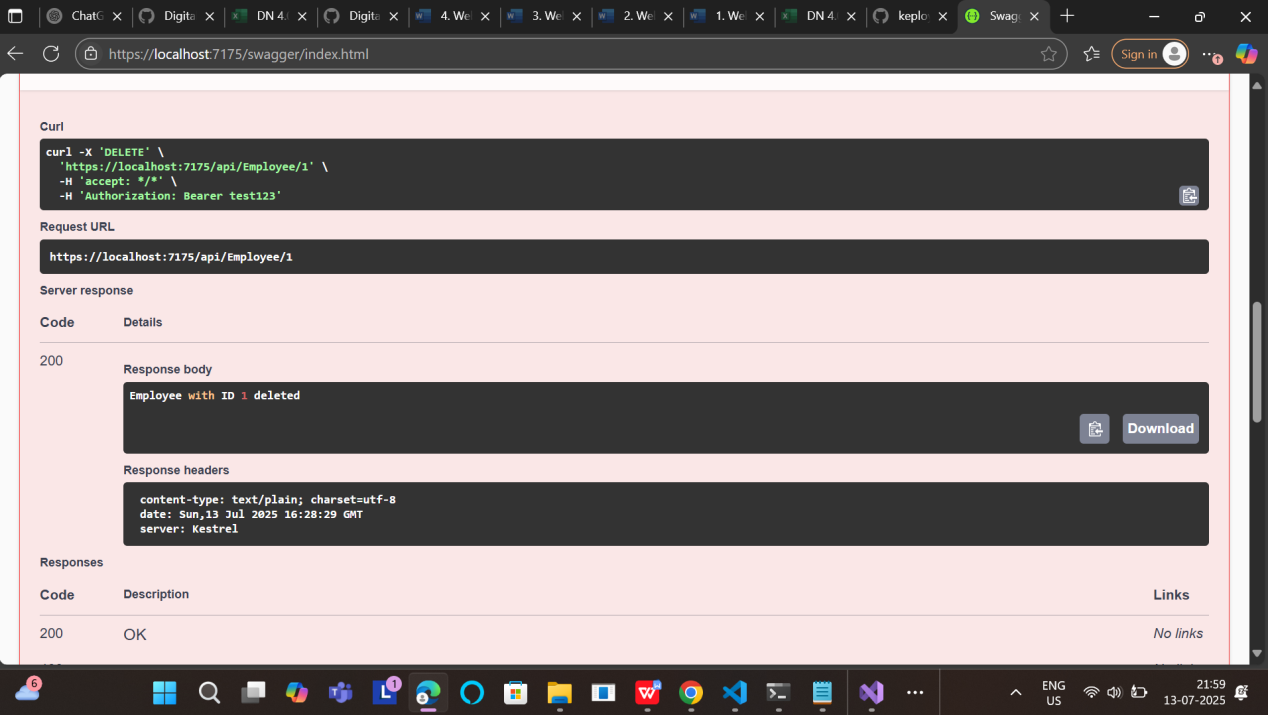
}

}

Test DELETE in **Swagger**

****

****

****

1. **WebApi\_Handson**

## 1. What is CORS?

**CORS** stands for Cross-Origin Resource Sharing.

It is a security feature enforced by web browsers that allows or blocks requests made from one origin (domain) to another

How to Enable CORS in ASP.NET Core:

In Program.cs:

builder.Services.AddCors(options =>

{

options.AddPolicy("MyPolicy", policy =>

{

policy.AllowAnyOrigin()

.AllowAnyMethod()

.AllowAnyHeader();

});

});

...

app.UseCors("MyPolicy"); // Add this before UseAuthentication()

1. JWT Authentication Setup

Install the Package

dotnet add package Microsoft.AspNetCore.Authentication.JwtBearer

Add JWT Config in Program.cs

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

string securityKey = "mysuperdupersecretmysuperdupersecret"; // at least 256-bit key

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

builder.Services.AddAuthentication(x =>

{

x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

x.DefaultSignInScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(JwtBearerDefaults.AuthenticationScheme, x =>

{

x.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = "mySystem",

ValidAudience = "myUsers",

IssuerSigningKey = symmetricSecurityKey

};

});

app.UseAuthentication();

app.UseAuthorization();

1. AuthController – Generate JWT Token

Create AuthController.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace MyWebAPI.Controllers

{

[ApiController]

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

[AllowAnonymous]

public class AuthController : ControllerBase

{

[HttpGet("get-token")]

public IActionResult GetToken()

{

var token = GenerateJSONWebToken(1, "Admin"); // UserId = 1, Role = Admin

return Ok(new { token });

}

private string GenerateJSONWebToken(int userId, string userRole)

{

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

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

var claims = new List<Claim>

{

new Claim("role", userRole), // 🛑 Use "role" instead of ClaimTypes.Role

new Claim("UserId", userId.ToString())

};

var token = new JwtSecurityToken(

issuer: "mySystem",

audience: "myUsers",

claims: claims,

expires: DateTime.Now.AddMinutes(10),

signingCredentials: credentials);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

}

1. Secure EmployeeController Using JWT

Add Authorization and Role Check

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using MyWebAPI.Models;

namespace MyWebAPI.Controllers

{

[Authorize(Roles = "Admin,POC")] // Only Admin and POC can access

[ApiController]

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

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<Skill>

{

new Skill { Id = 1, Name = "C#" },

new Skill { Id = 2, Name = "SQL" }

},

DateOfBirth = new DateTime(1990, 5, 20)

},

new Employee

{

Id = 2,

Name = "Jane",

Salary = 60000,

Permanent = false,

Department = new Department { Id = 2, Name = "IT" },

Skills = new List<Skill>

{

new Skill { Id = 3, Name = "Java" },

new Skill { Id = 4, Name = "Python" }

},

DateOfBirth = new DateTime(1992, 10, 15)

}

};

[HttpGet]

public ActionResult<List<Employee>> GetAll()

{

return Ok(employees);

}

[HttpGet("{id}")]

public ActionResult<Employee> GetById(int id)

{

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

if (emp == null)

return NotFound("Employee not found");

return Ok(emp);

}

[HttpPost]

public IActionResult AddEmployee([FromBody] Employee emp)

{

emp.Id = employees.Max(e => e.Id) + 1; // Auto-increment Id

employees.Add(emp);

return Ok(emp);

}

[HttpPut("{id}")]

public IActionResult UpdateEmployee(int id, [FromBody] Employee updatedEmp)

{

if (id <= 0)

return BadRequest("Invalid employee id");

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

if (emp == null)

return BadRequest("Invalid employee id");

// Update fields

emp.Name = updatedEmp.Name;

emp.Salary = updatedEmp.Salary;

emp.Permanent = updatedEmp.Permanent;

emp.Department = updatedEmp.Department;

emp.Skills = updatedEmp.Skills;

emp.DateOfBirth = updatedEmp.DateOfBirth;

return Ok(emp);

}

[HttpDelete("{id}")]

public IActionResult DeleteEmployee(int id)

{

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

if (emp == null)

return NotFound("Employee not found");

employees.Remove(emp);

return Ok($"Employee with Id {id} deleted");

}

[HttpGet("cause-exception")]

[ProducesResponseType(500)]

public IActionResult CauseException()

{

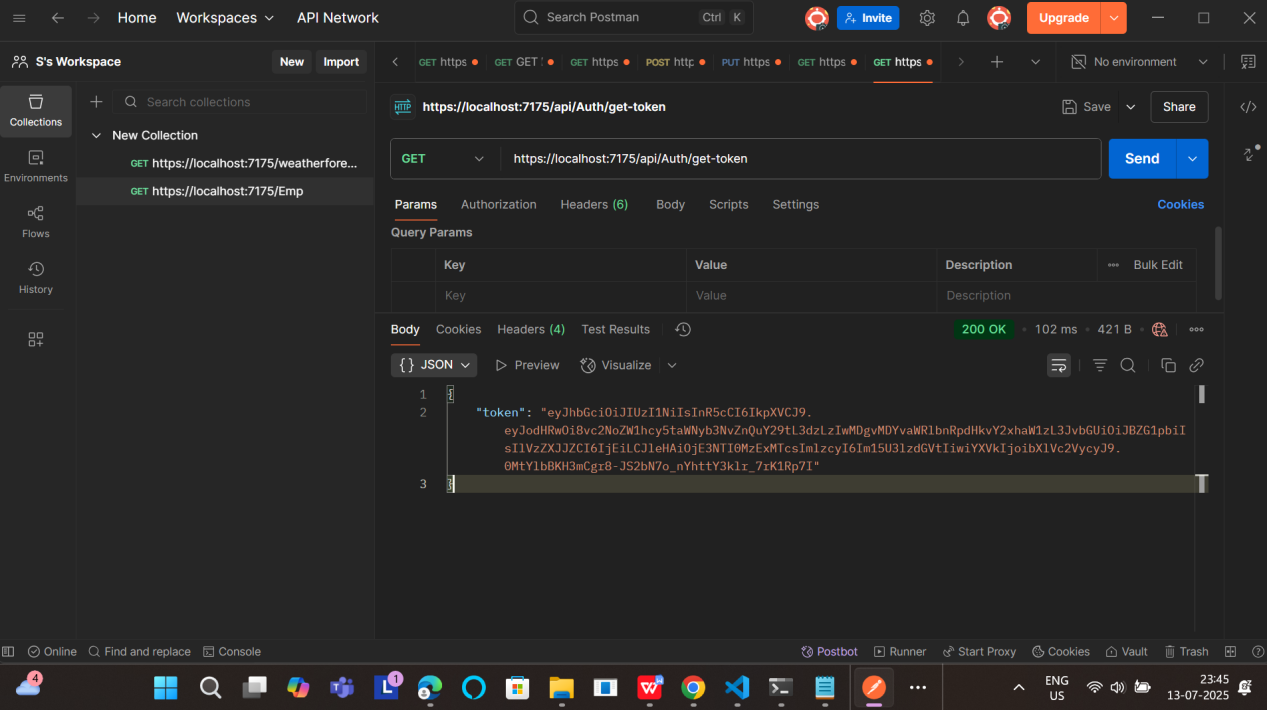
throw new Exception("Test exception");

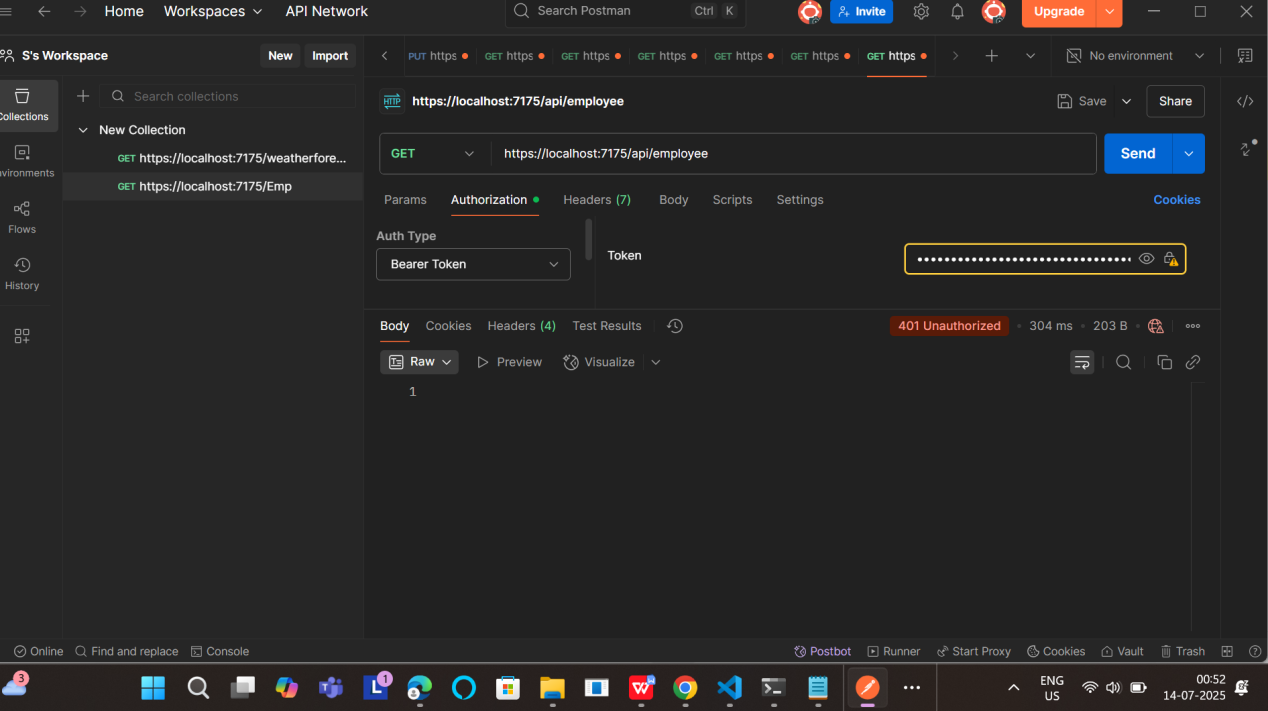
}

}

}

1. Postman Testing





.