**1. WebApi\_Handson**

**Code:**

using Microsoft.AspNetCore.Mvc;

namespace WebApplication1.Controllers

{

[ApiController]

[Route("[controller]")]

public class ValuesController : ControllerBase

{

[HttpGet]

public IActionResult Get()

{

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

}

[HttpPost]

public IActionResult Post([FromBody] string value)

{

return Ok($"You posted: {value}");

}

[HttpPut("{id}")]

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

{

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

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

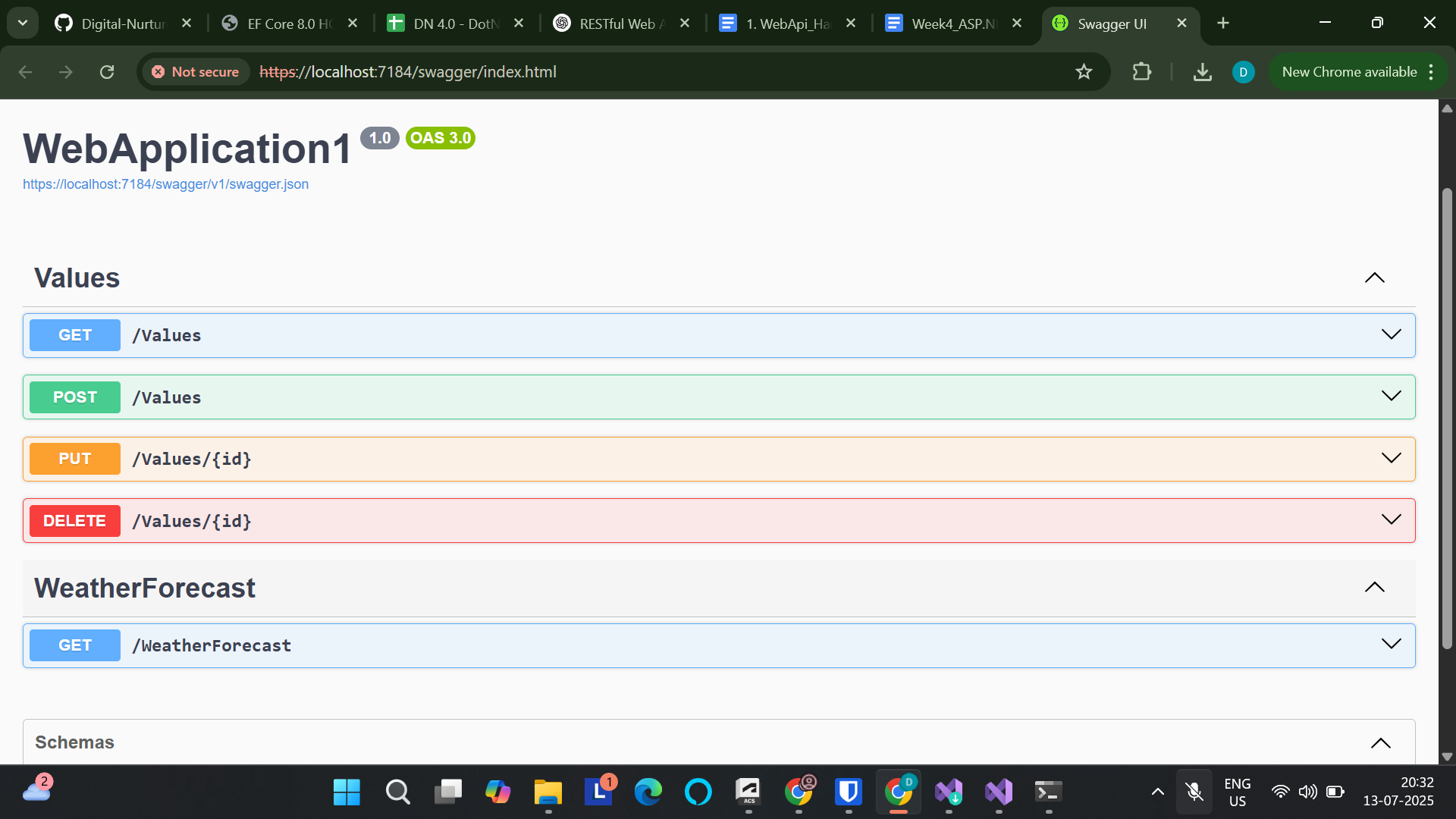
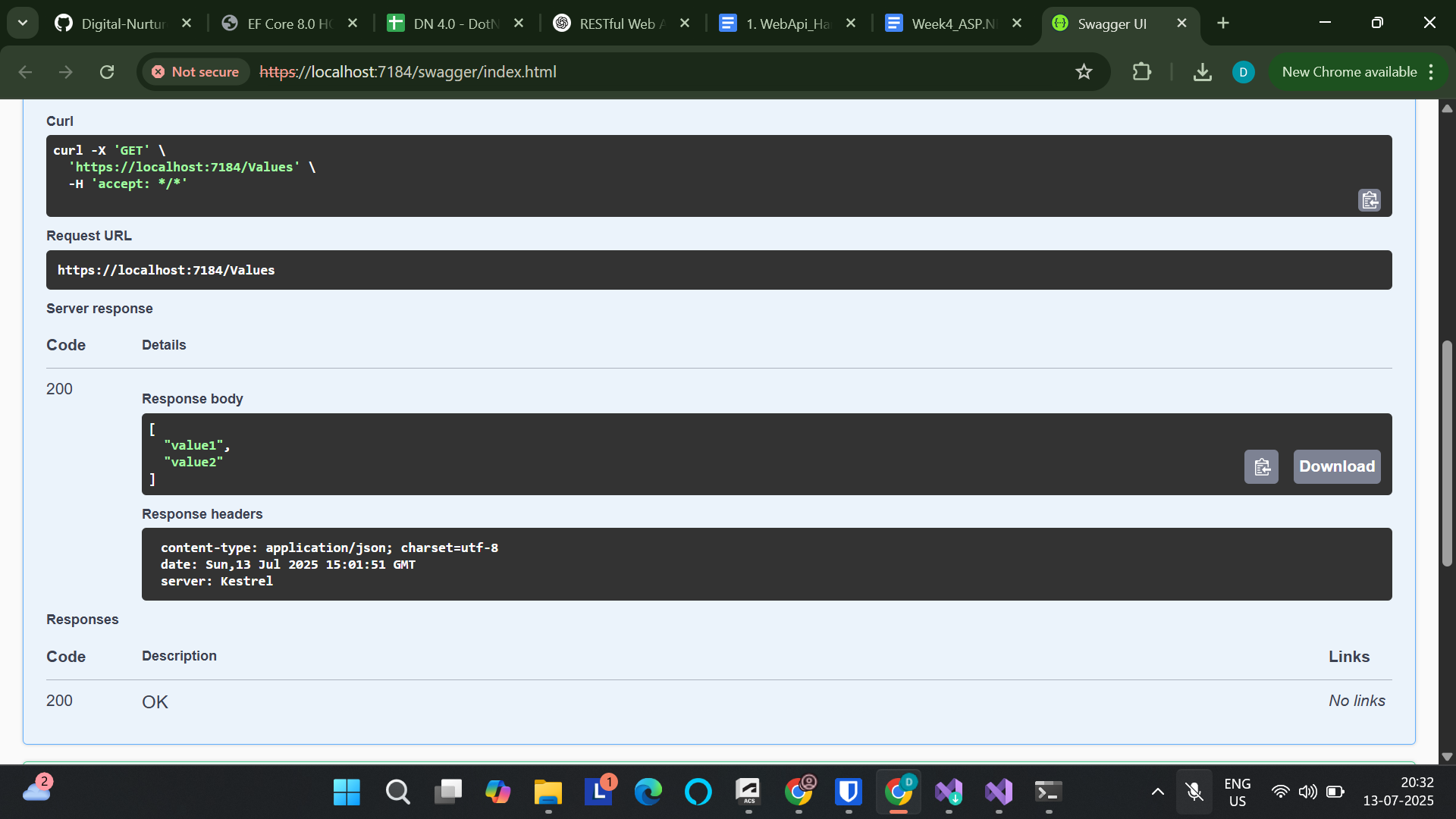
return Ok($"Deleted id {id}");

}

}

}

**Output:**

****

**2. WebApi\_Handson**

**Code:**

[Program.cs](http://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"),

Contact = new OpenApiContact

{

Name = "John Doe",

Email = "john@xyzmail.com",

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

},

License = new OpenApiLicense

{

Name = "License Terms",

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

}

});

});

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

[ValuesController.cs](http://valuescontroller.cs)

using Microsoft.AspNetCore.Mvc;

namespace WebApplication1.Controllers

{

[ApiController]

[Route("emp")]

public class EmployeeController : ControllerBase

{

[HttpGet]

public IActionResult Get()

{

var employees = new[] { "Alice", "Bob", "Charlie" };

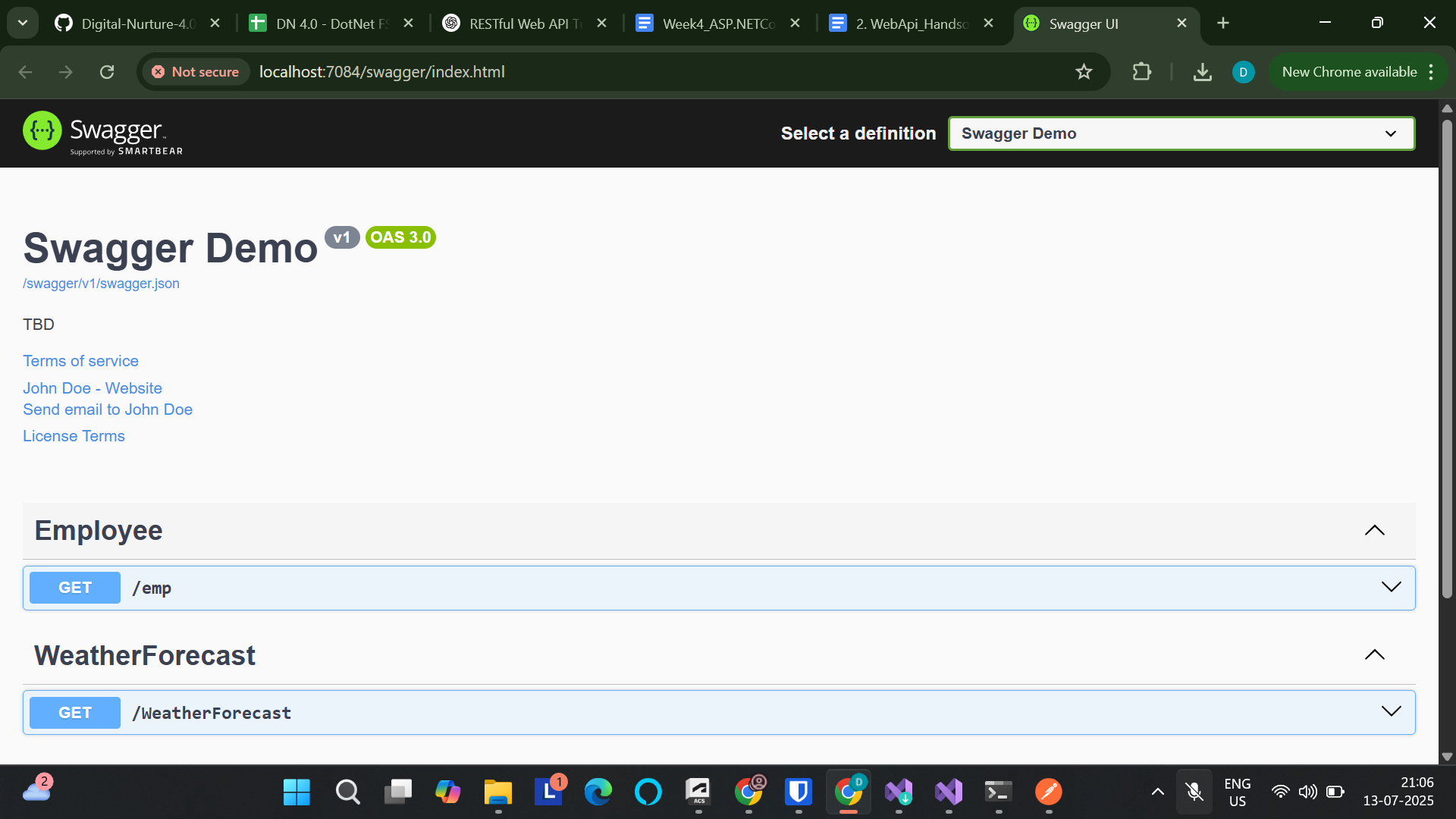
return Ok(employees);

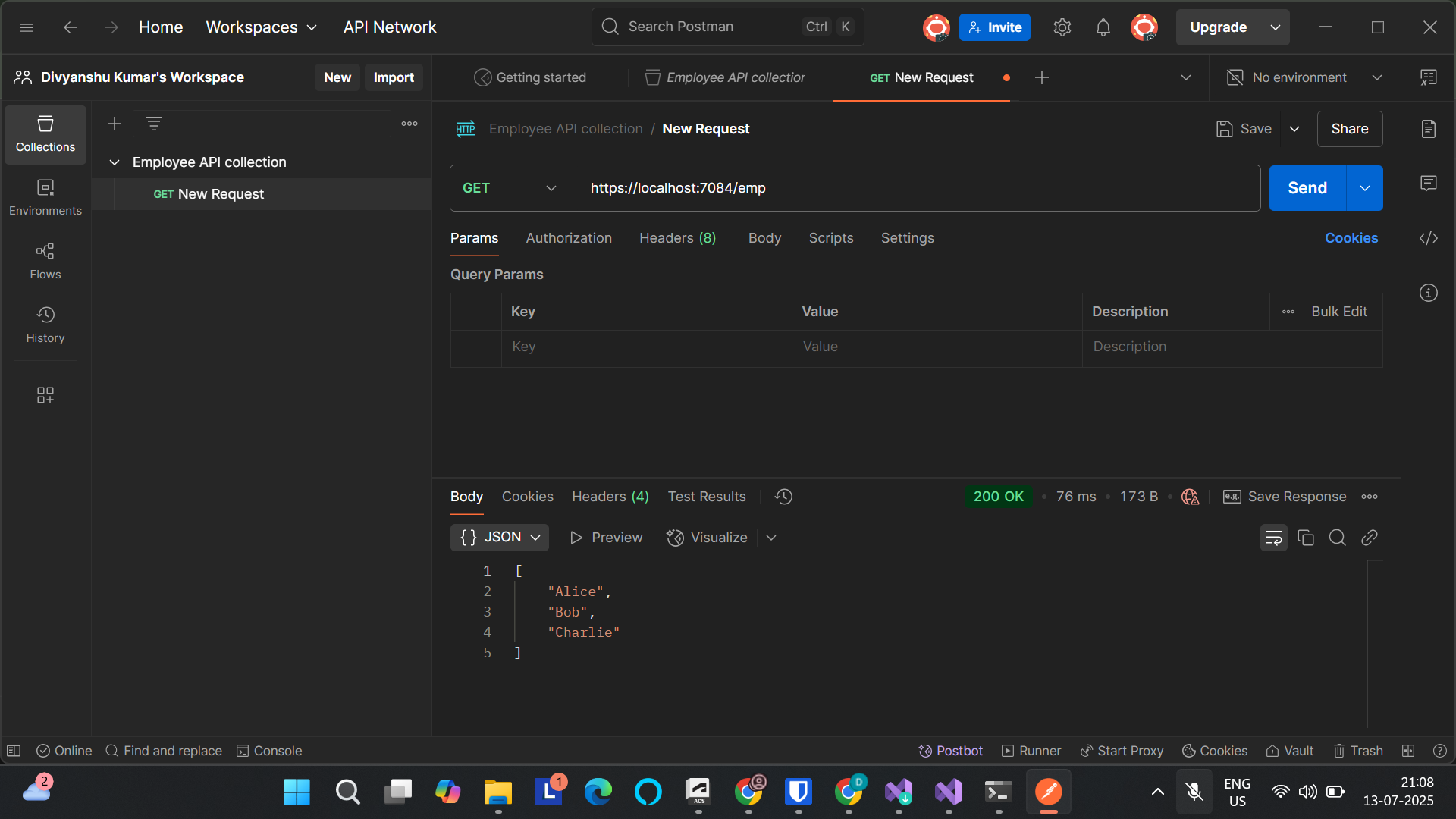
}

}

}

**Output:**

****

****

**3. WebApi\_Handson**

**Code:**

[Program.cs](http://program.cs)

using Microsoft.OpenApi.Models;

using WebApplication1.Filters;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

builder.Services.AddControllers(options =>

{

options.Filters.Add<CustomExceptionFilter>();

});

builder.Services.AddScoped<CustomAuthFilter>();

builder.Services.AddScoped<CustomExceptionFilter>();

builder.Services.AddEndpointsApiExplorer();

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"),

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")

}

});

c.AddSecurityDefinition("Bearer", new Microsoft.OpenApi.Models.OpenApiSecurityScheme

{

Name = "Authorization",

Type = Microsoft.OpenApi.Models.SecuritySchemeType.ApiKey,

Scheme = "Bearer",

BearerFormat = "JWT",

In = Microsoft.OpenApi.Models.ParameterLocation.Header,

Description = "Enter 'Bearer' [space] and then your token."

});

c.AddSecurityRequirement(new Microsoft.OpenApi.Models.OpenApiSecurityRequirement

{

{

new Microsoft.OpenApi.Models.OpenApiSecurityScheme

{

Reference = new Microsoft.OpenApi.Models.OpenApiReference

{

Type = Microsoft.OpenApi.Models.ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

new string[] {}

}

});

});

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

[ValuesController.cs](http://valuescontroller.cs)

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using WebApplication1.Models;

using WebApplication1.Filters;

namespace WebApplication1.Controllers

{

[ApiController]

[Route("emp")]

[ServiceFilter(typeof(CustomAuthFilter))] // Apply your auth filter here

public class EmployeeController : ControllerBase

{

private readonly List<Employee> \_employees;

public EmployeeController()

{

\_employees = GetStandardEmployeeList();

}

[HttpGet]

[AllowAnonymous] // Allows access without auth

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public ActionResult<List<Employee>> Get()

{

// throw new Exception("Simulating server error"); // Optional for testing

return Ok(\_employees);

}

[HttpPost]

public IActionResult Post([FromBody] Employee emp)

{

if (emp == null) return BadRequest("Invalid data");

\_employees.Add(emp);

return Ok(emp);

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John",

Salary = 50000,

Permanent = true,

Department = new Department { Id = 101, Name = "HR" },

Skills = new List<Skill> {

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

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

},

DateOfBirth = new DateTime(1990, 5, 23)

}

};

}

}

}

[Employee.cs](http://employee.cs)

using System;

using System.Collections.Generic;

namespace WebApplication1.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; }

}

}

[Skill.cs](http://skills.cs)

namespace WebApplication1.Models

{

public class Skill

{

public int Id { get; set; }

public string Name { get; set; }

}

}

[Department.cs](http://department.cs)

namespace WebApplication1.Models

{

public class Department

{

public int Id { get; set; }

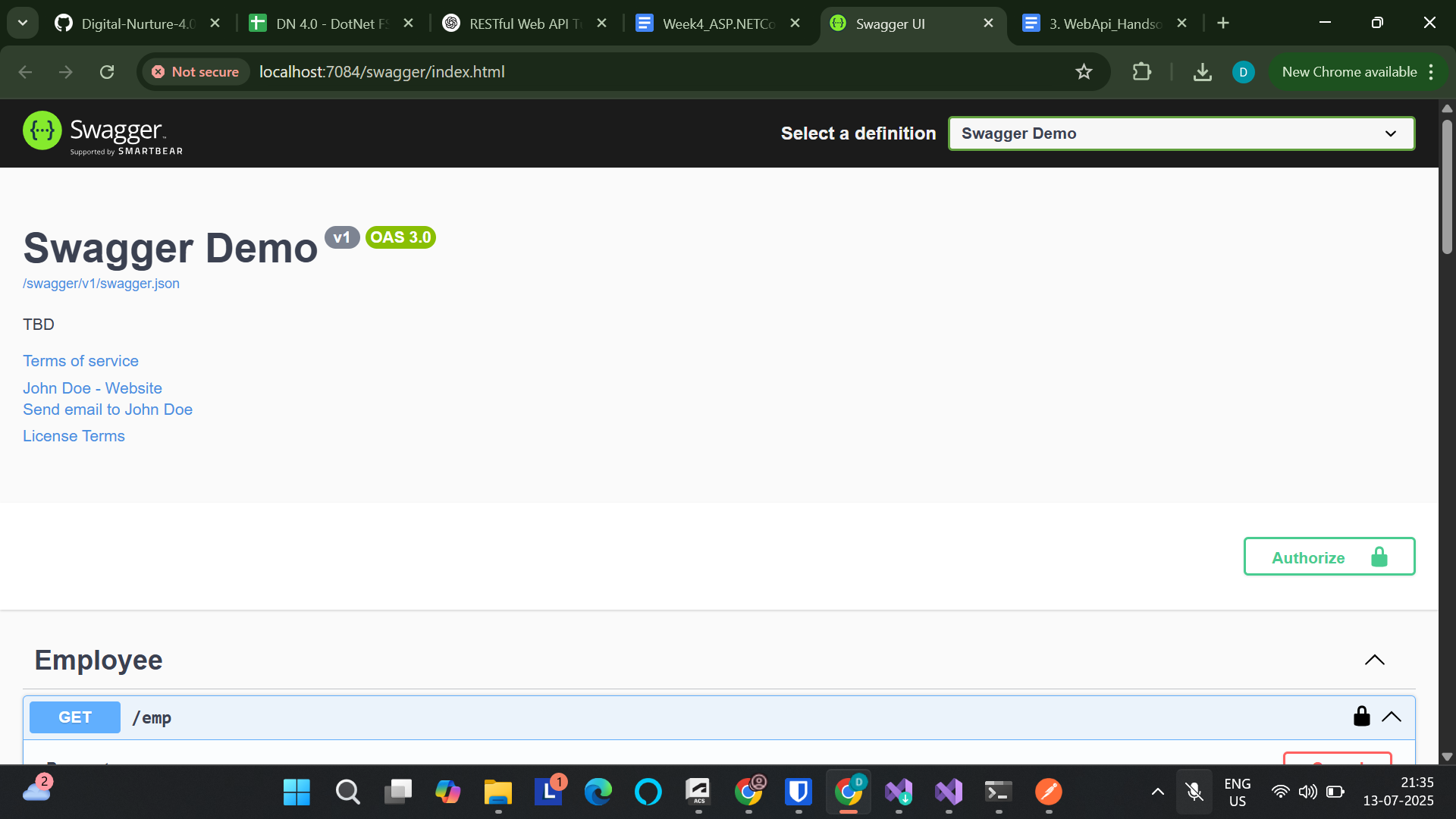
public string Name { get; set; }

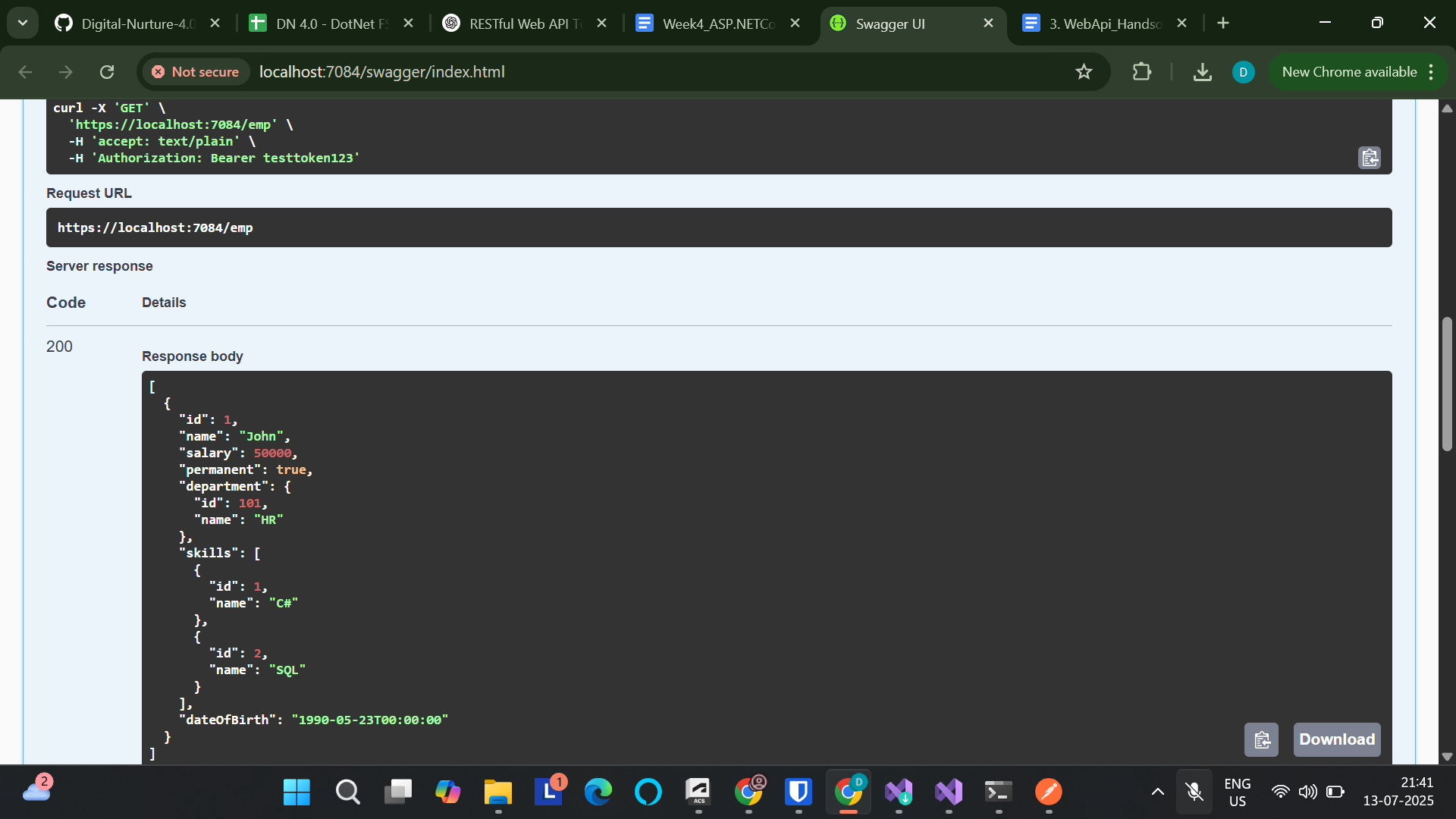
}

}

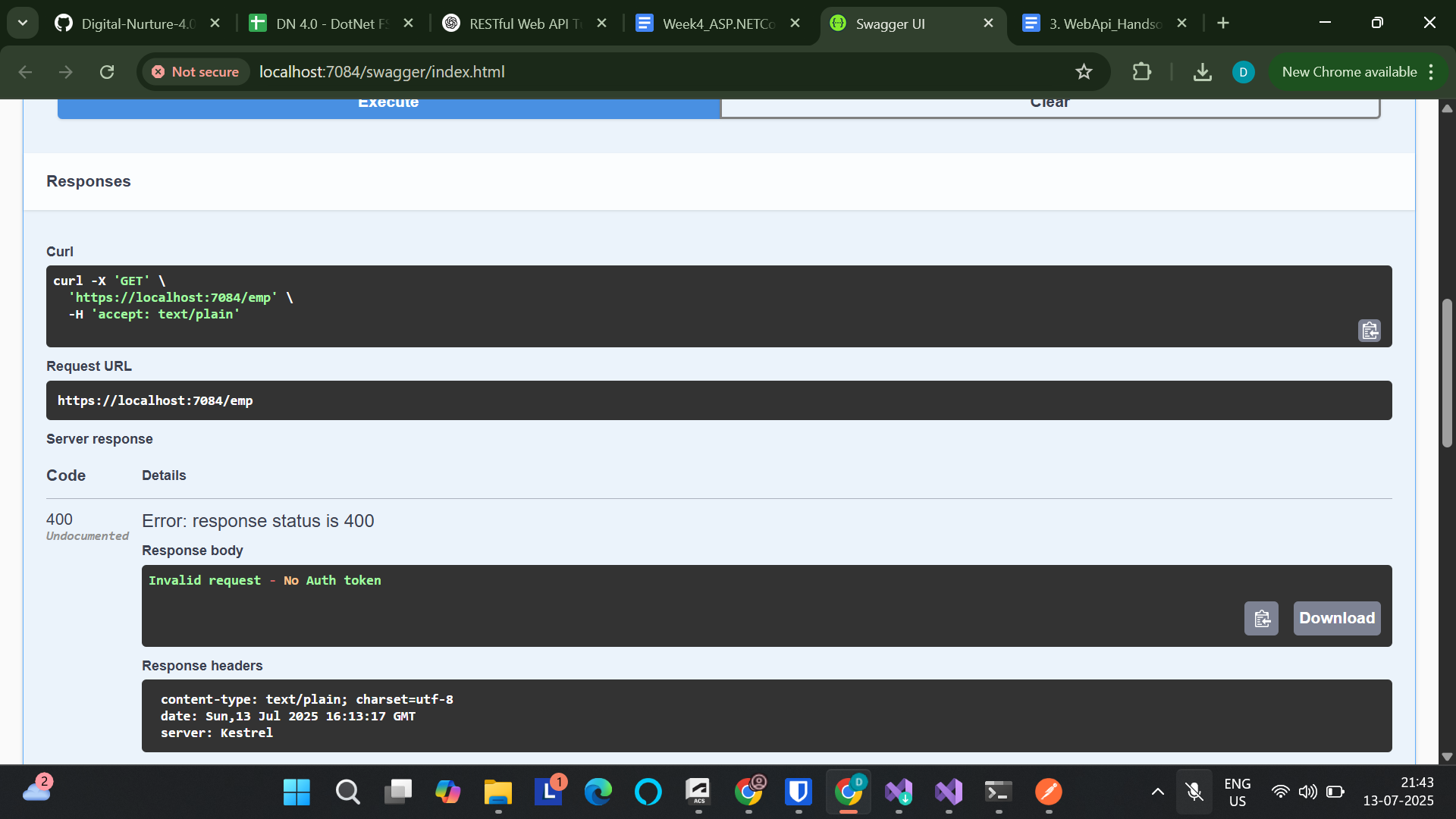
**Output:**

**With Authorization:**

****

****

**Without Authorization:**

****

**4. WebApi\_Handson**

**Code:**

[**Program.cs**](http://program.cs)

using Microsoft.OpenApi.Models;

using WebApplication1.Filters;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

builder.Services.AddControllers(options =>

{

options.Filters.Add<CustomExceptionFilter>();

});

builder.Services.AddScoped<CustomAuthFilter>();

builder.Services.AddScoped<CustomExceptionFilter>();

builder.Services.AddEndpointsApiExplorer();

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"),

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")

}

});

c.AddSecurityDefinition("Bearer", new Microsoft.OpenApi.Models.OpenApiSecurityScheme

{

Name = "Authorization",

Type = Microsoft.OpenApi.Models.SecuritySchemeType.ApiKey,

Scheme = "Bearer",

BearerFormat = "JWT",

In = Microsoft.OpenApi.Models.ParameterLocation.Header,

Description = "Enter 'Bearer' [space] and then your token."

});

c.AddSecurityRequirement(new Microsoft.OpenApi.Models.OpenApiSecurityRequirement

{

{

new Microsoft.OpenApi.Models.OpenApiSecurityScheme

{

Reference = new Microsoft.OpenApi.Models.OpenApiReference

{

Type = Microsoft.OpenApi.Models.ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

new string[] {}

}

});

});

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

[ValueController.cs](http://valuecontroller.cs)

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using WebApplication1.Models;

using WebApplication1.Filters;

namespace WebApplication1.Controllers

{

[ApiController]

[Route("emp")]

[ServiceFilter(typeof(CustomAuthFilter))] // Apply your auth filter here

public class EmployeeController : ControllerBase

{

private static List<Employee> \_employees = new List<Employee>();

public EmployeeController()

{

if (\_employees.Count == 0)

{

\_employees = GetStandardEmployeeList();

}

}

[HttpGet]

[AllowAnonymous] // Allows access without auth

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public ActionResult<List<Employee>> Get()

{

// throw new Exception("Simulating server error"); // Optional for testing

return Ok(\_employees);

}

[HttpPost]

public IActionResult Post([FromBody] Employee emp)

{

if (emp == null) return BadRequest("Invalid data");

\_employees.Add(emp);

return Ok(emp);

}

[HttpPut]

public ActionResult<Employee> Put([FromBody] Employee updatedEmployee)

{

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

{

return BadRequest("Invalid employee id");

}

// Try to find the employee in the list

var employee = \_employees.FirstOrDefault(e => e.Id == updatedEmployee.Id);

if (employee == null)

{

return BadRequest("Invalid employee id");

}

// Update properties

employee.Name = updatedEmployee.Name;

employee.Salary = updatedEmployee.Salary;

employee.Permanent = updatedEmployee.Permanent;

employee.Department = updatedEmployee.Department;

employee.Skills = updatedEmployee.Skills;

employee.DateOfBirth = updatedEmployee.DateOfBirth;

return Ok(employee); // Return updated employee

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John",

Salary = 50000,

Permanent = true,

Department = new Department { Id = 101, Name = "HR" },

Skills = new List<Skill> {

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

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

},

DateOfBirth = new DateTime(1990, 5, 23)

}

};

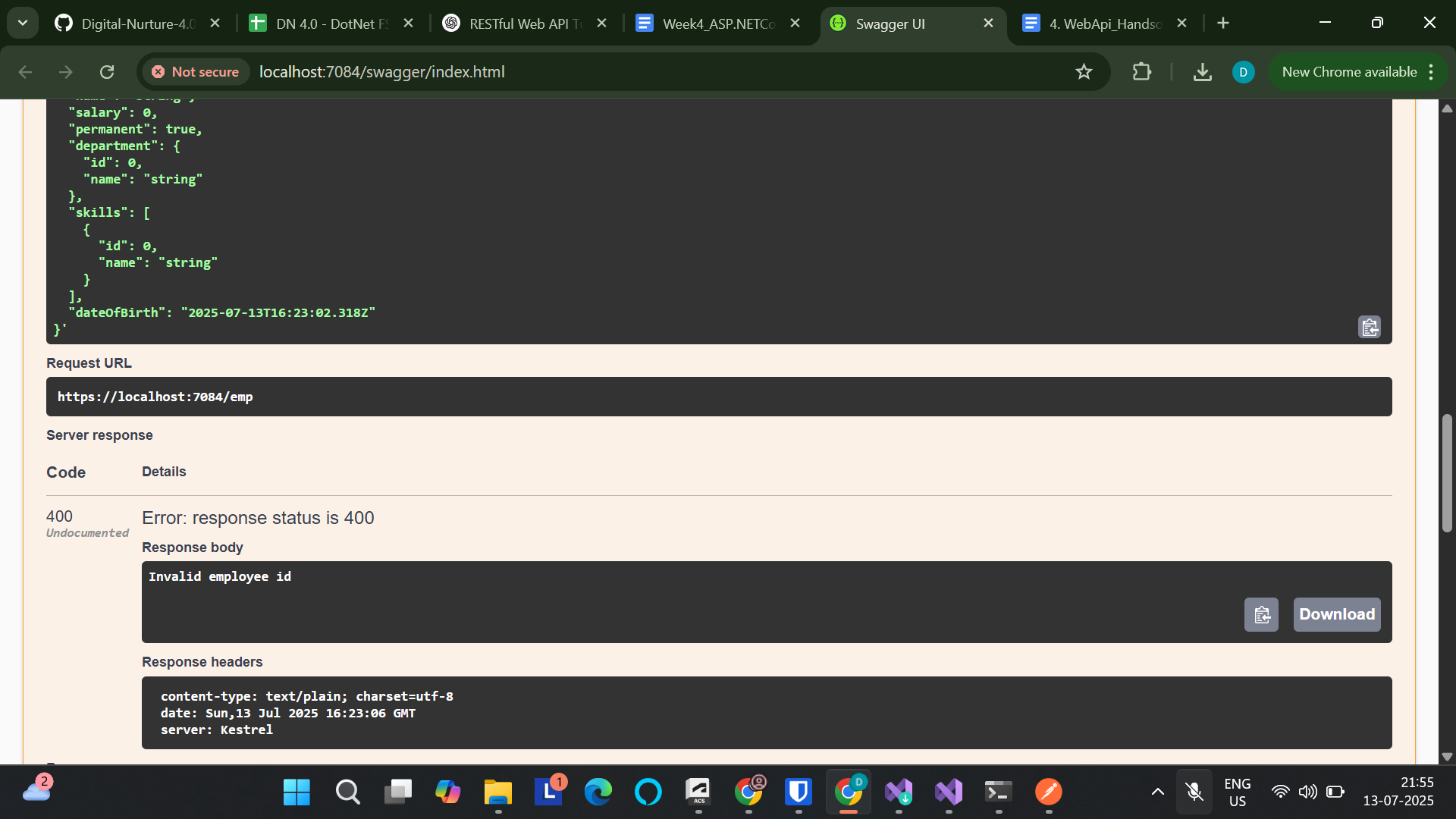
}

}

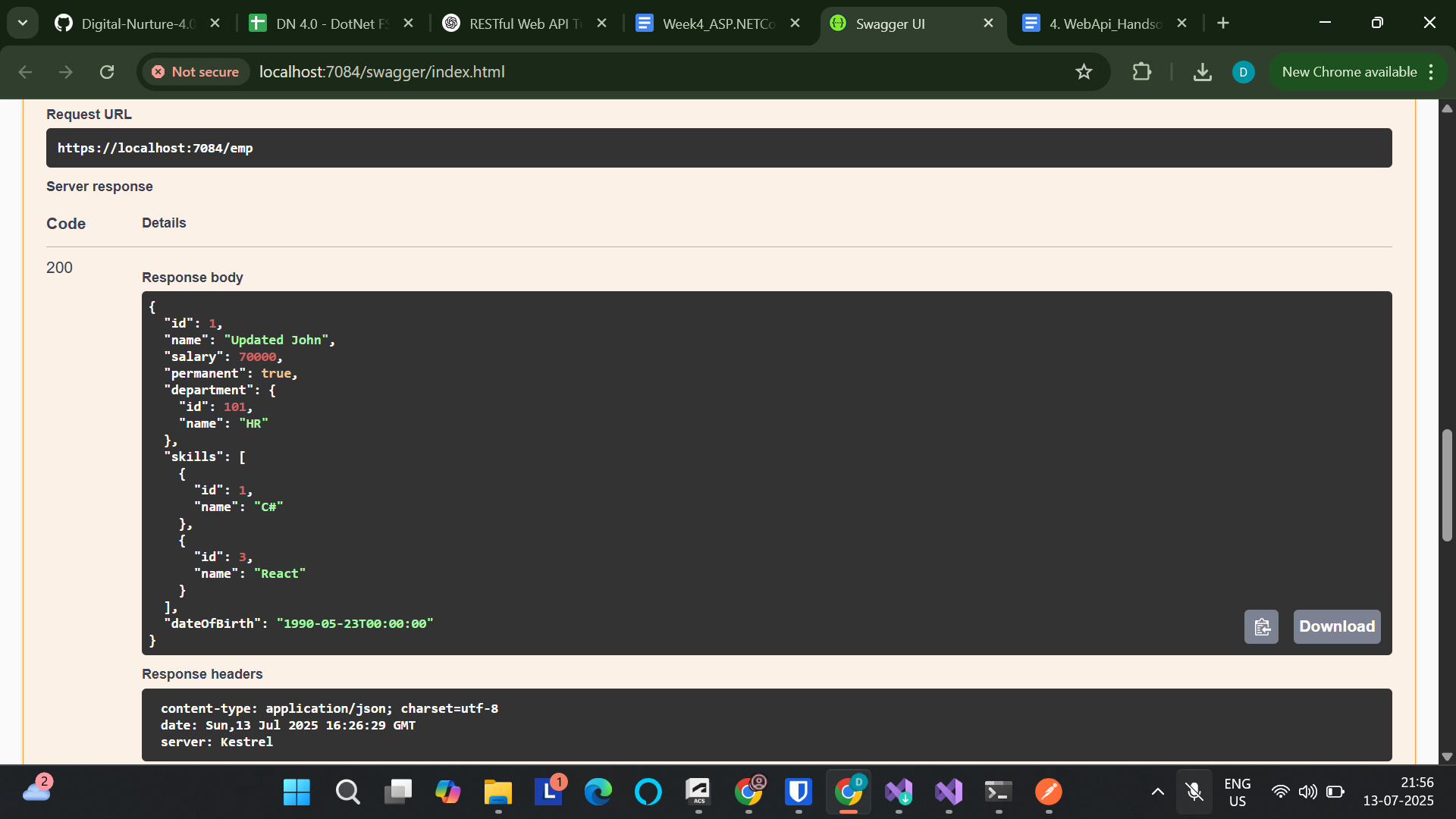
}

**Output:**

**ID is 0 - meaning invalid:**

****

**ID is valid:**

****

**5. WebApi\_Handson**

**Code:**

[**Program.cs**](http://program.cs)

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using System.Text;

using WebApplication1.Filters;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

var securityKey = "mysuperdupersecretkeythatismorethan32chars";

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

builder.Services

.AddAuthentication(options =>

{

options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = "mySystem",

ValidAudience = "myUsers",

IssuerSigningKey = keyBytes

};

});

builder.Services.AddAuthorization();

builder.Services.AddCors(opts =>

{

opts.AddPolicy("DevPolicy", p =>

p.AllowAnyOrigin()

.AllowAnyHeader()

.AllowAnyMethod());

});

builder.Services.AddControllers();

builder.Services.AddControllers(options =>

{

options.Filters.Add<CustomExceptionFilter>();

});

builder.Services.AddScoped<CustomAuthFilter>();

builder.Services.AddScoped<CustomExceptionFilter>();

builder.Services.AddEndpointsApiExplorer();

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"),

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")

}

});

c.AddSecurityDefinition("Bearer", new Microsoft.OpenApi.Models.OpenApiSecurityScheme

{

Name = "Authorization",

Type = Microsoft.OpenApi.Models.SecuritySchemeType.ApiKey,

Scheme = "Bearer",

BearerFormat = "JWT",

In = Microsoft.OpenApi.Models.ParameterLocation.Header,

Description = "Enter 'Bearer' [space] and then your token."

});

c.AddSecurityRequirement(new Microsoft.OpenApi.Models.OpenApiSecurityRequirement

{

{

new Microsoft.OpenApi.Models.OpenApiSecurityScheme

{

Reference = new Microsoft.OpenApi.Models.OpenApiReference

{

Type = Microsoft.OpenApi.Models.ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

new string[] {}

}

});

});

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

}

app.UseCors("DevPolicy");

app.UseAuthentication();

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

[AuthController.cs](http://authcontroller.cs)

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace WebApplication1.Controllers

{

[ApiController]

[Route("auth")]

[AllowAnonymous] // No token required for this controller

public class AuthController : ControllerBase

{

[HttpGet("token")]

public IActionResult GetToken()

{

var token = GenerateJsonWebToken(1, "Admin");

return Ok(token);

}

private string GenerateJsonWebToken(int userId, string userRole)

{

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretkeythatismorethan32chars"));

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

var claims = new List<Claim>

{

new Claim(ClaimTypes.Role, userRole),

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

};

var token = new JwtSecurityToken(

issuer: "mySystem",

audience: "myUsers",

claims: claims,

expires: DateTime.Now.AddMinutes(10), // change to 2 to test expiry

signingCredentials: credentials);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

}

[ValueController.cs](http://valuecontroller.cs)

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using WebApplication1.Models;

namespace WebApplication1.Controllers

{

[ApiController]

[Route("emp")]

[Authorize(Roles = "Admin,POC")] // 🔐 Only allow users with Admin or POC role

public class EmployeeController : ControllerBase

{

private static List<Employee> \_employees = new List<Employee>();

public EmployeeController()

{

if (\_employees.Count == 0)

{

\_employees = GetStandardEmployeeList();

}

}

[HttpGet]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status401Unauthorized)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public ActionResult<List<Employee>> Get()

{

return Ok(\_employees);

}

[HttpPost]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

public IActionResult Post([FromBody] Employee emp)

{

if (emp == null) return BadRequest("Invalid data");

\_employees.Add(emp);

return Ok(emp);

}

[HttpPut]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

public ActionResult<Employee> Put([FromBody] Employee updatedEmployee)

{

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

{

return BadRequest("Invalid employee id");

}

var employee = \_employees.FirstOrDefault(e => e.Id == updatedEmployee.Id);

if (employee == null)

{

return BadRequest("Invalid employee id");

}

employee.Name = updatedEmployee.Name;

employee.Salary = updatedEmployee.Salary;

employee.Permanent = updatedEmployee.Permanent;

employee.Department = updatedEmployee.Department;

employee.Skills = updatedEmployee.Skills;

employee.DateOfBirth = updatedEmployee.DateOfBirth;

return Ok(employee);

}

[HttpDelete("{id:int}")]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

public IActionResult Delete(int id)

{

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

if (employee == null)

{

return BadRequest("Invalid employee id");

}

\_employees.Remove(employee);

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

}

// 🔁 Initial data method

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John",

Salary = 50000,

Permanent = true,

Department = new Department { Id = 101, Name = "HR" },

Skills = new List<Skill> {

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

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

},

DateOfBirth = new DateTime(1990, 5, 23)

}

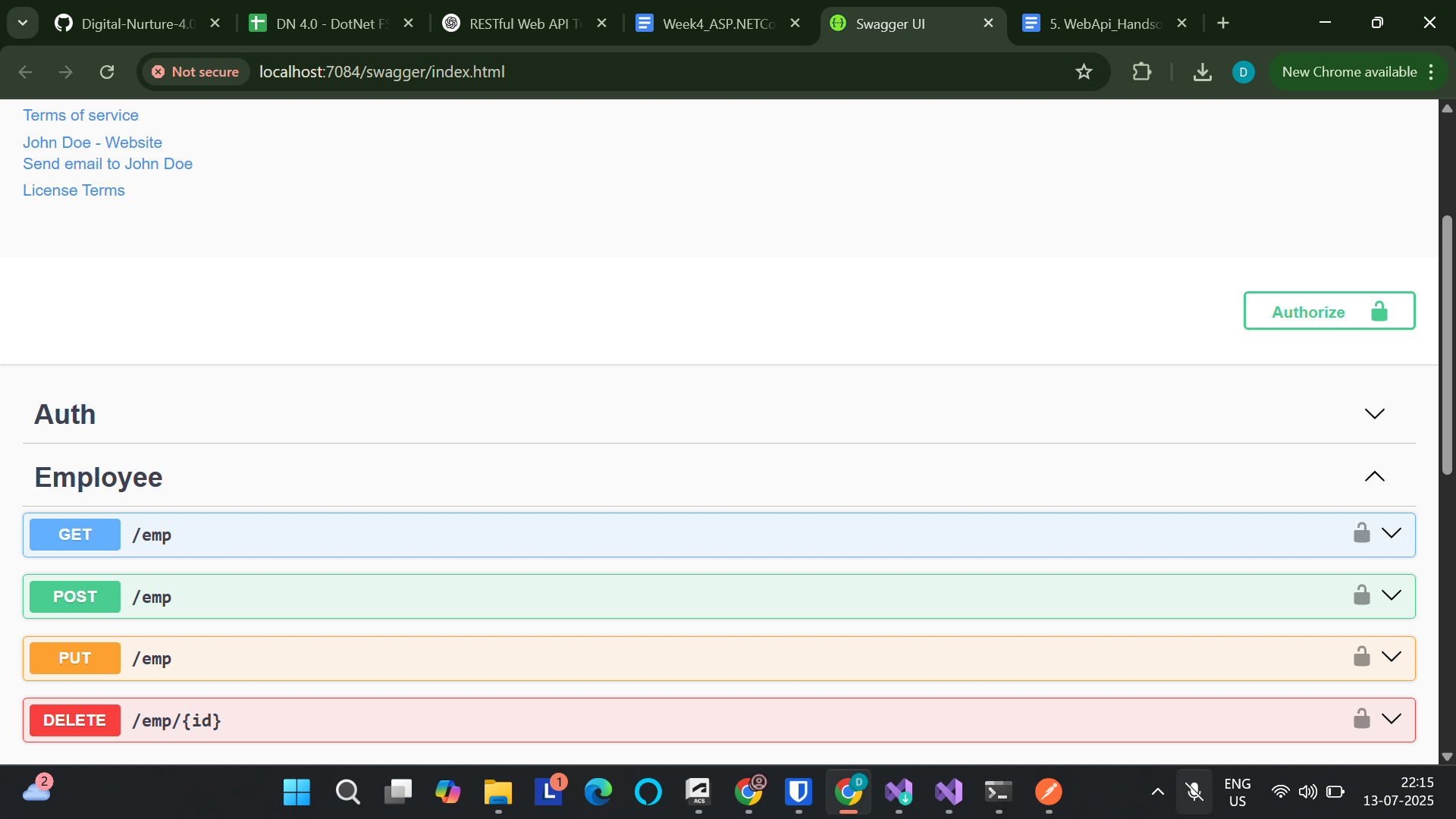
};

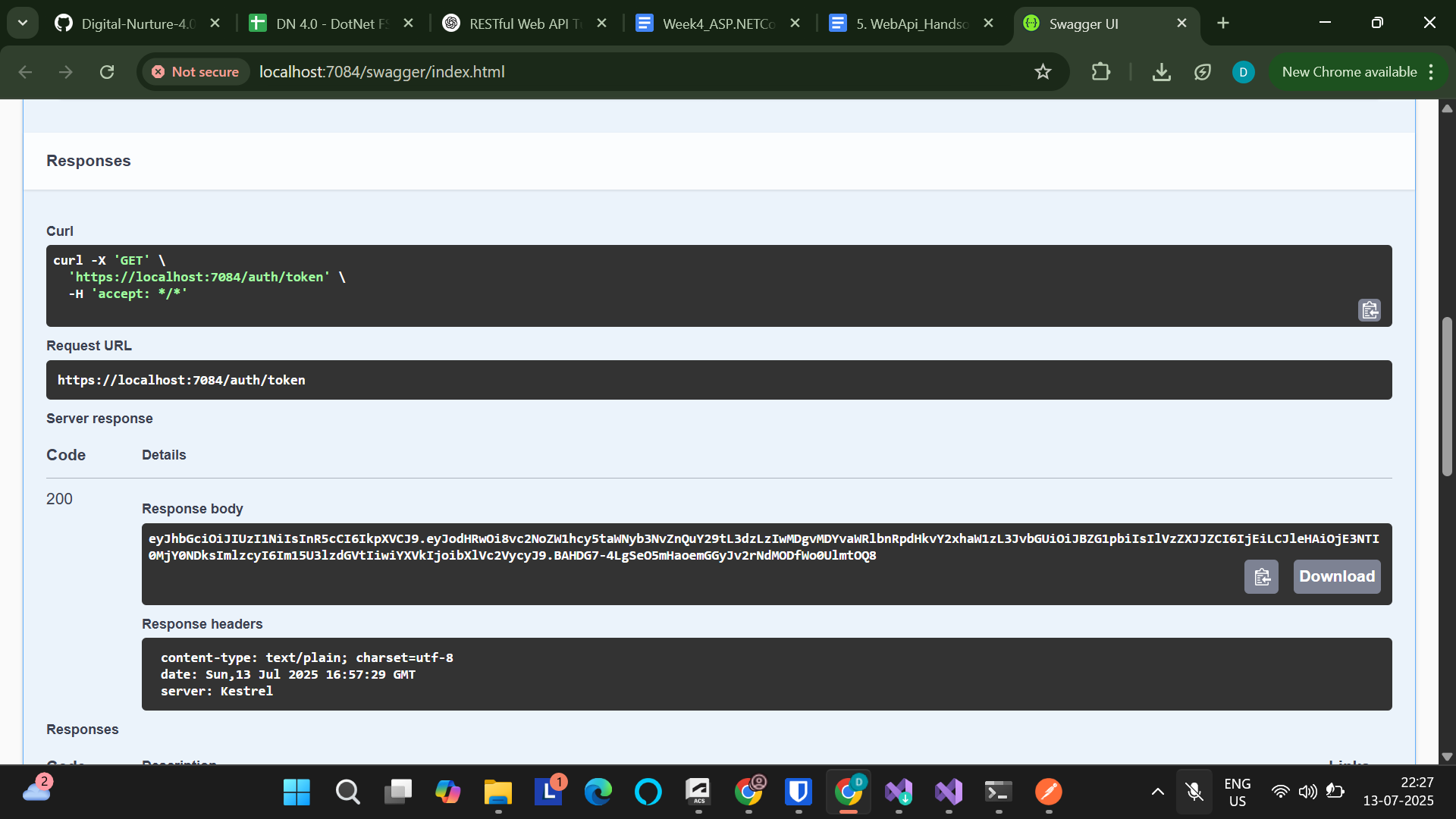
}

}

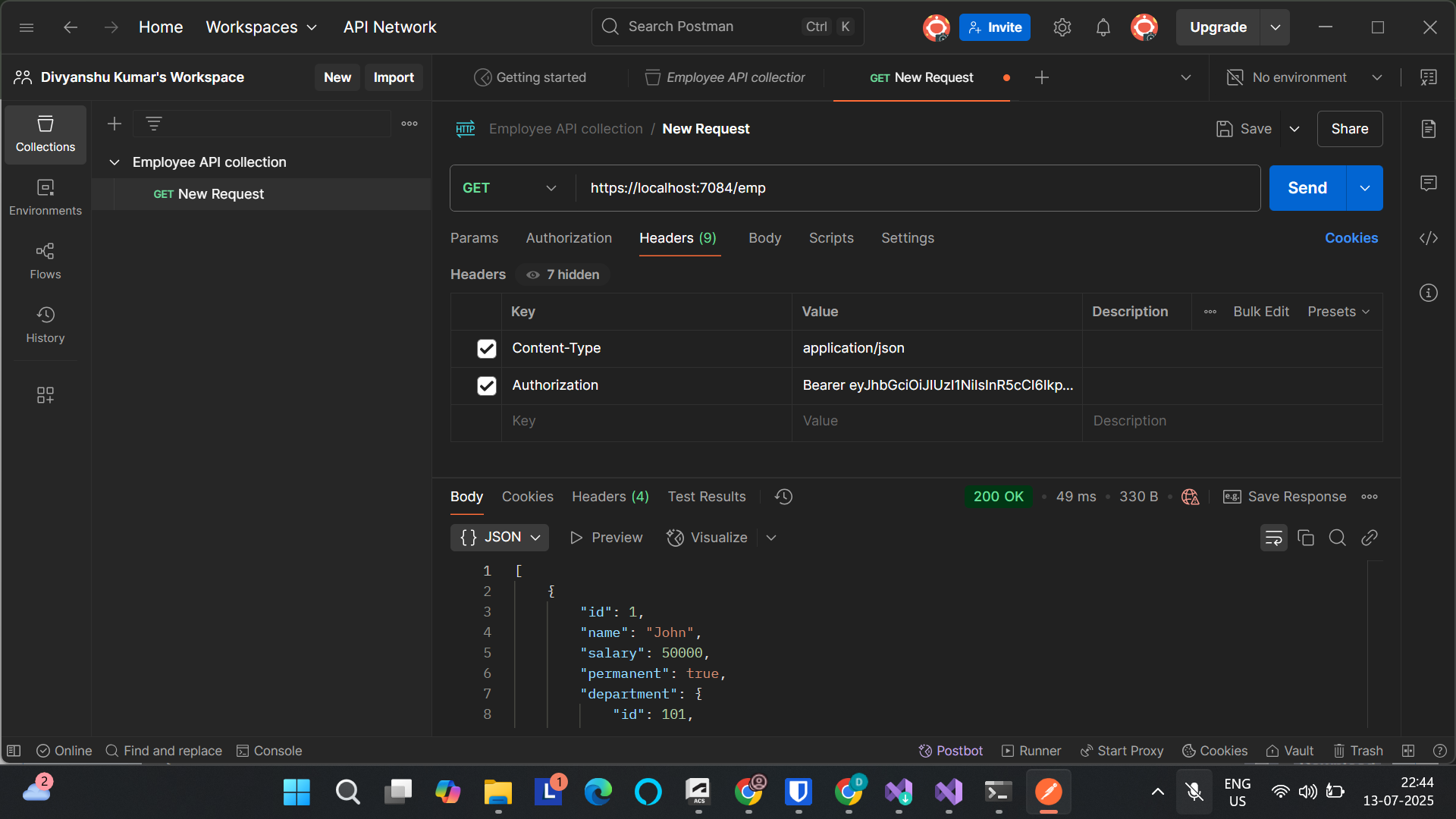
}

**Output:**

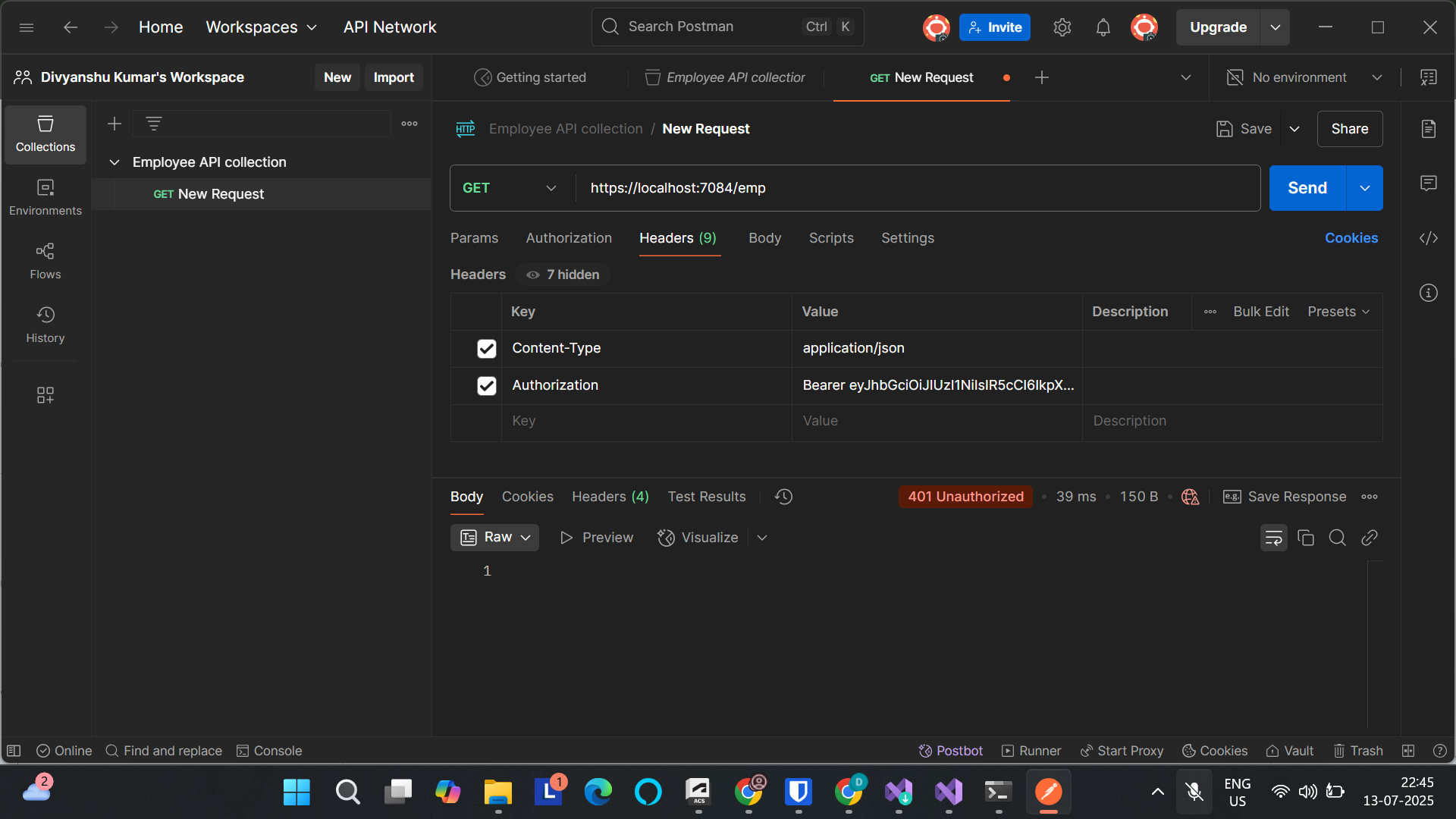
****

****

**With a JWT token:**

****

**Without a JWT token/ Invalid JWT token:**

****