**RESTful Web Services, Web API & Microservices:**  
REST (Representational State Transfer) is an architectural style for designing network-based applications. It uses standard HTTP methods like GET, POST, PUT, and DELETE for communication and follows a stateless protocol. A Web API is a framework in .NET used to create RESTful services that can send responses in various formats like JSON or XML. Microservices are small, independent services that perform specific business tasks and communicate via lightweight protocols like HTTP.

**HTTP Request & Response:**  
An HttpRequest represents the incoming request from a client and includes method type, headers, body, and query parameters. An HttpResponse is the server’s response, which includes a status code, response headers, and data (like JSON) returned to the client.

**HTTP Action Verbs:**  
Web APIs use HTTP verbs to define actions. GET is used to fetch data, POST to create new data, PUT to update existing data, and DELETE to remove data. These are declared in Web API methods using attributes like [HttpGet], [HttpPost], etc.

**HTTP Status Codes:**  
Status codes indicate the result of the request. Common ones include 200 OK (success), 400 BadRequest (client error), 401 Unauthorized (authentication required), and 500 InternalServerError (server-side failure). These are returned using action result types like Ok(), BadRequest(), Unauthorized(), etc.

**Structure of a Web API:**  
A Web API project contains controllers that inherit from ApiController or ControllerBase. Each controller contains action methods that use HTTP attributes and handle specific requests like reading or writing data.

**Configuration Files:**  
In ASP.NET Core, Startup.cs is used to configure services and middleware. appsettings.json holds configuration values like connection strings. launchSettings.json defines how the app starts during development. In older .NET Framework versions (like 4.5), configuration is managed via WebApi.config and Route.config.

Q1)

ValuesController.cs:

using Microsoft.AspNetCore.Mvc;

[ApiController]

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

public class ValuesController : ControllerBase

{

[HttpGet]

public IEnumerable<string> Get()

{

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

}

[HttpPost]

public void Post([FromBody] string value)

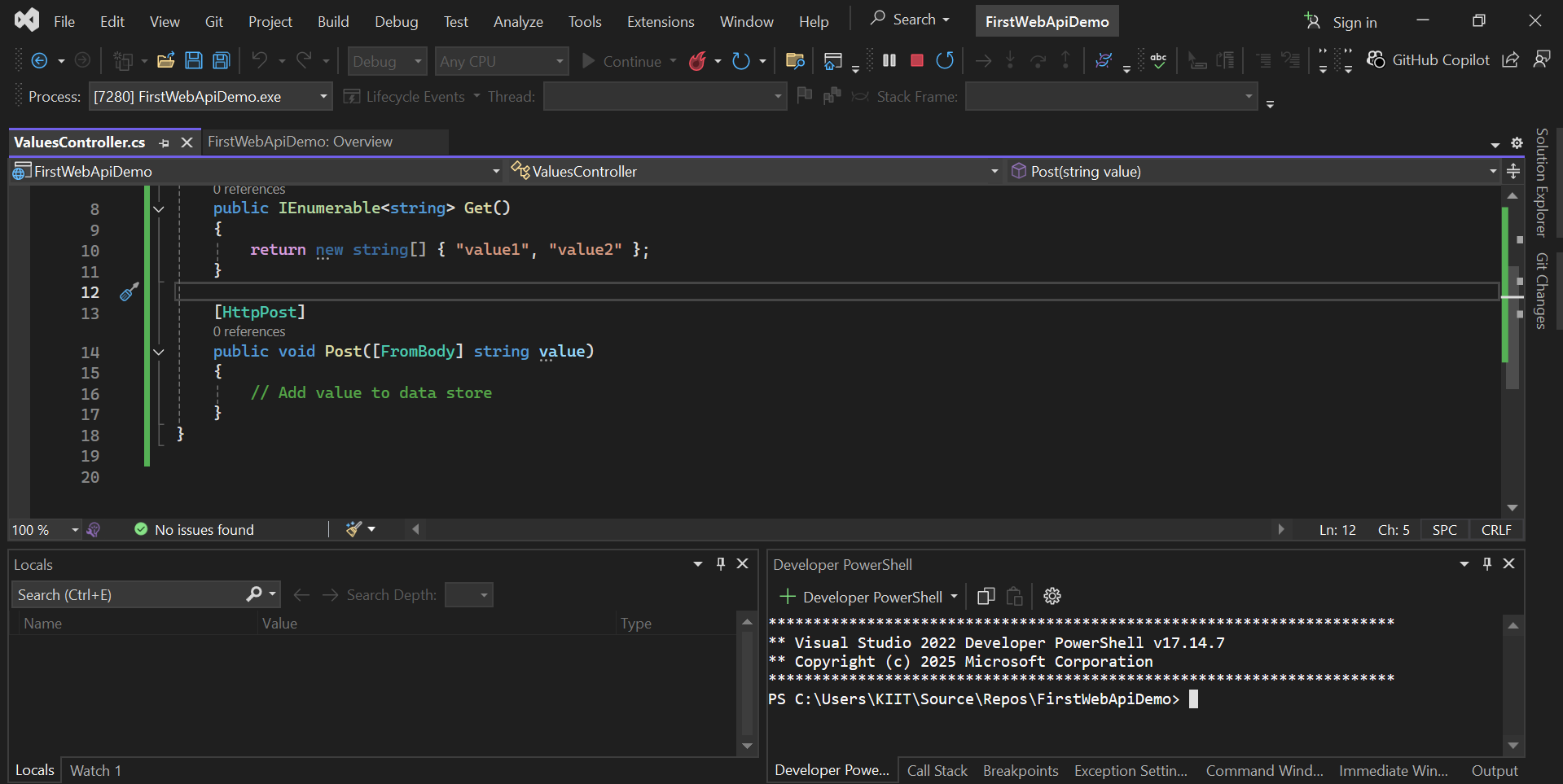
{

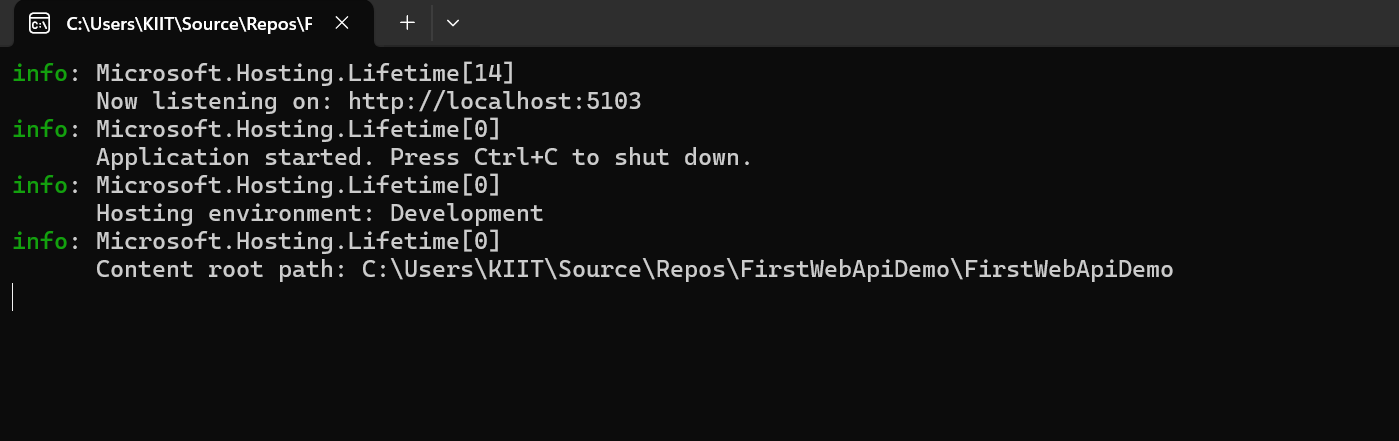
// Add value to data store

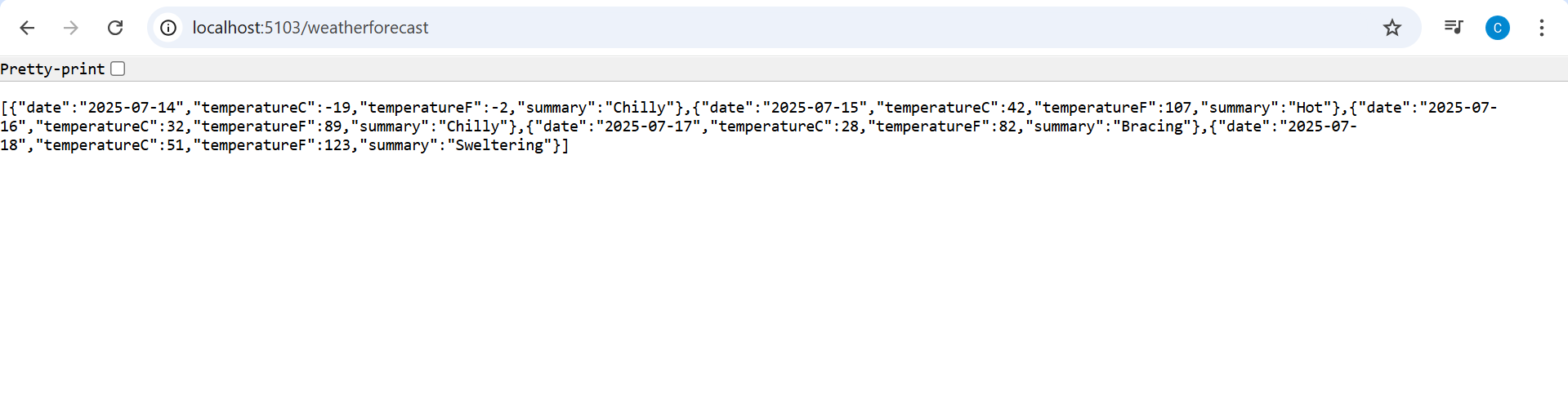
}

}

OUTPUT:



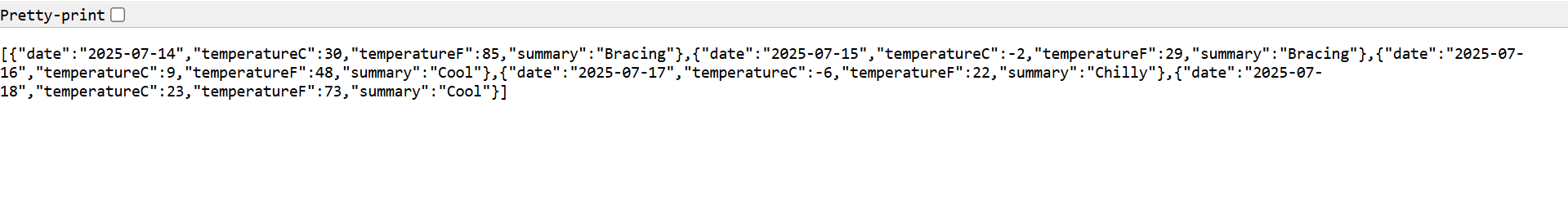


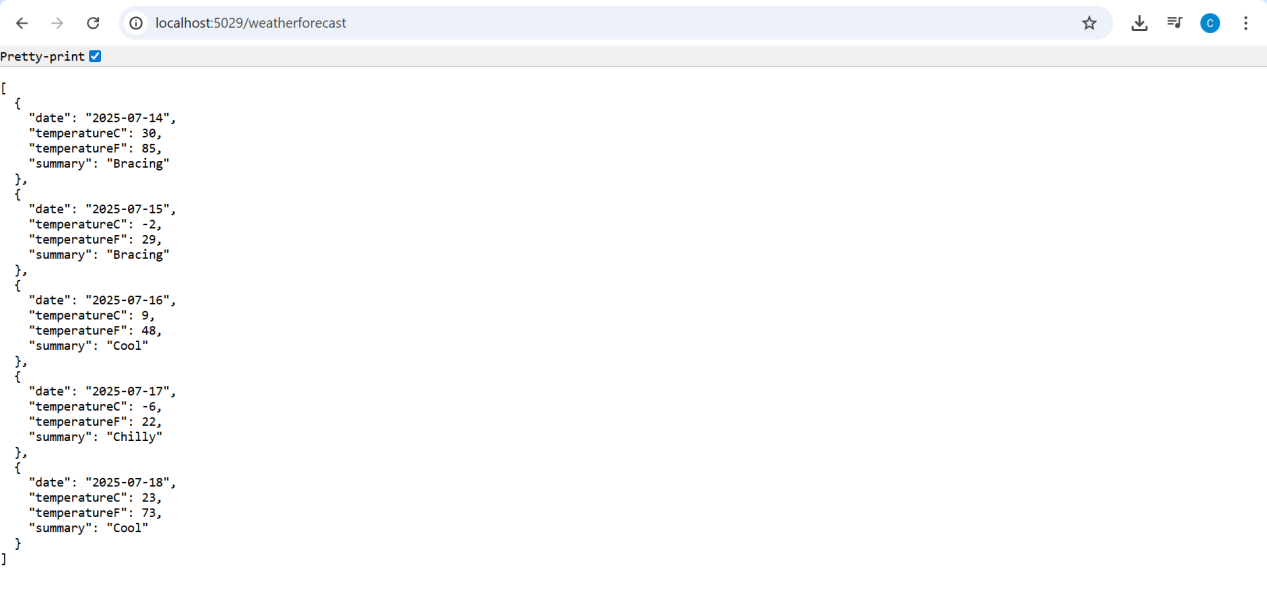


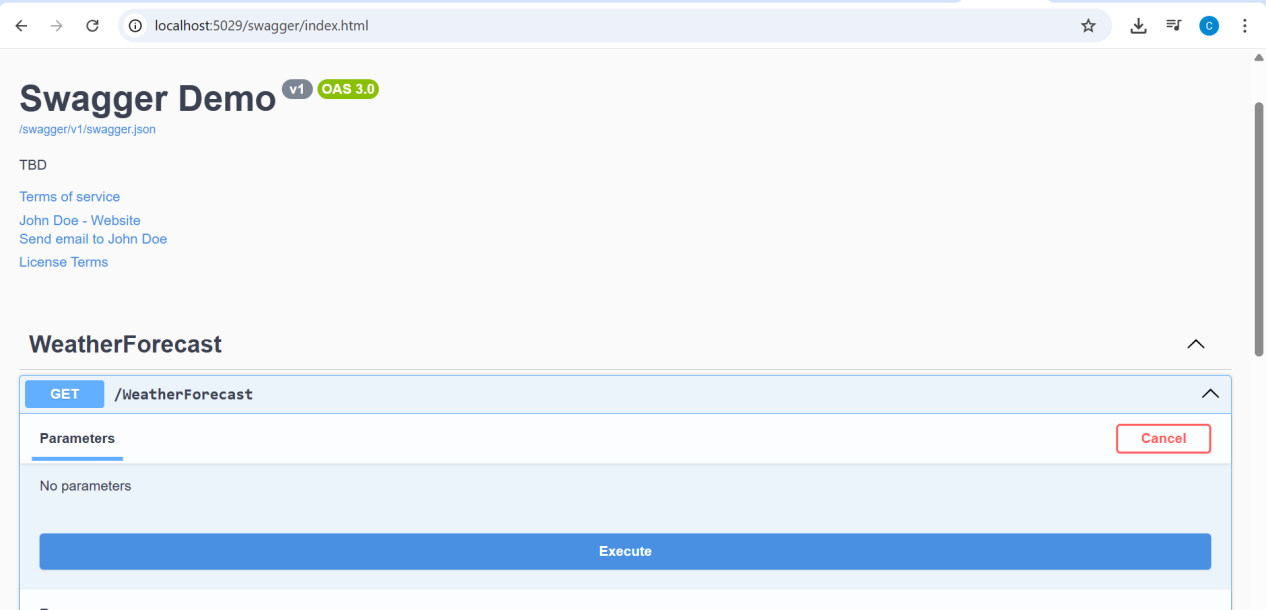


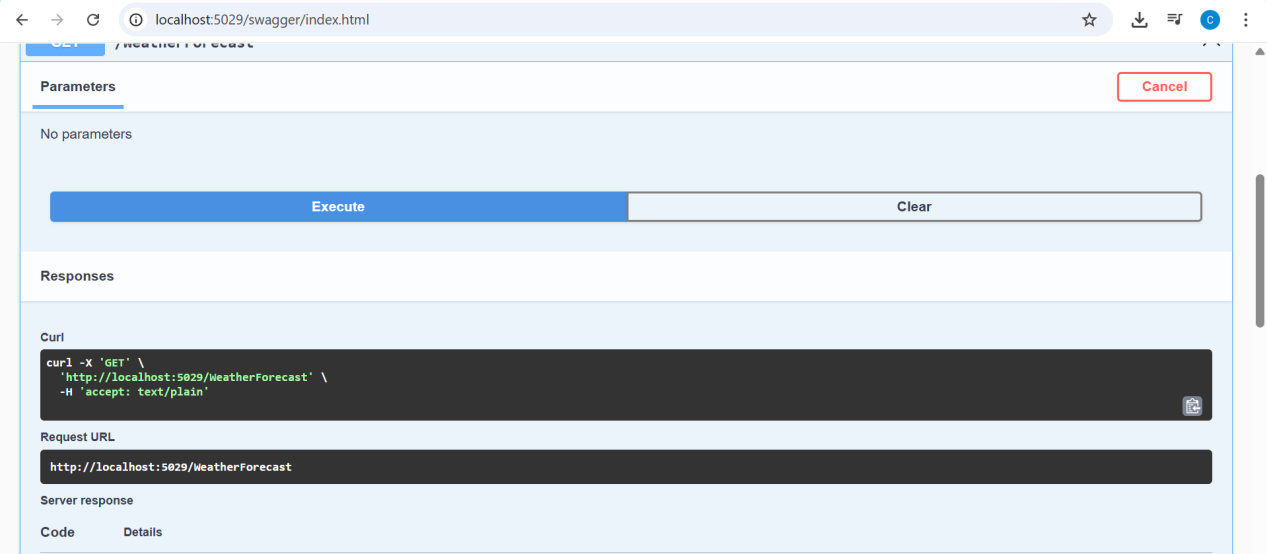
Q2)

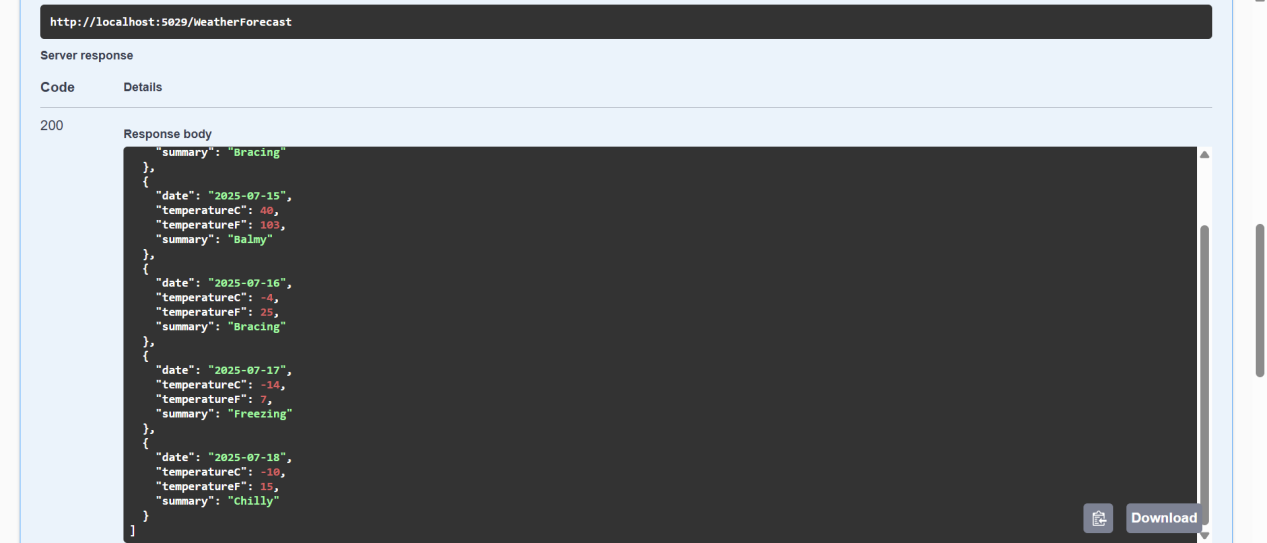
OUTPUT:

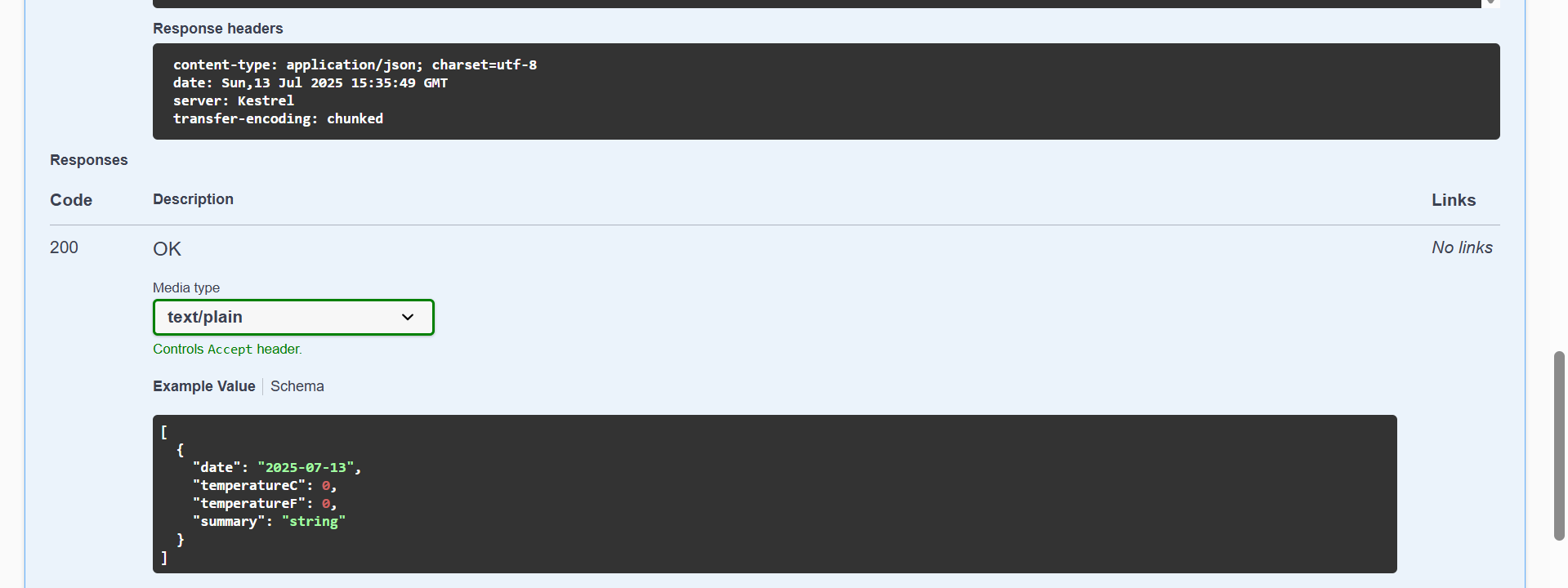




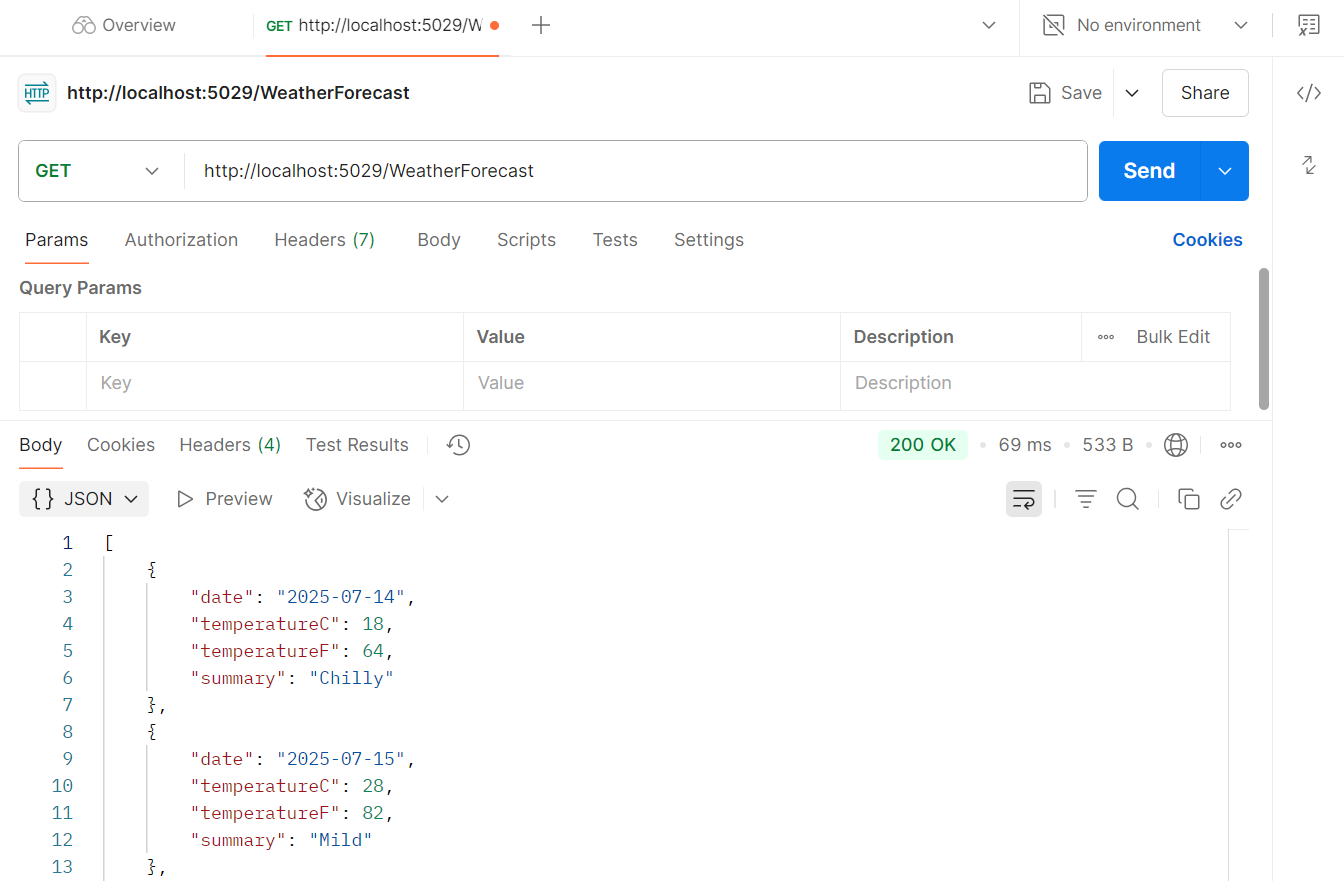








POSTMAN:



{}JSON:

[

    {

        "date": "2025-07-14",

        "temperatureC": 43,

        "temperatureF": 109,

        "summary": "Hot"

    },

    {

        "date": "2025-07-15",

        "temperatureC": 9,

        "temperatureF": 48,

        "summary": "Hot"

    },

    {

        "date": "2025-07-16",

        "temperatureC": 44,

        "temperatureF": 111,

        "summary": "Sweltering"

    },

    {

        "date": "2025-07-17",

        "temperatureC": 34,

        "temperatureF": 93,

        "summary": "Bracing"

    },

    {

        "date": "2025-07-18",

        "temperatureC": 31,

        "temperatureF": 87,

        "summary": "Freezing"

    }

]

Q3) WebAPI\_3:

EmployeeController.cs:

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using EmployeeApi.Models;

[ApiController]

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

public class EmployeeController : ControllerBase

{

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "Alice",

Salary = 50000,

Permanent = true,

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

Skills = new List<Skill> { new Skill { Id = 1, Name = "Communication" } },

DateOfBirth = new DateTime(1990, 1, 1)

}

// Add more sample employees as needed

};

}

[AllowAnonymous]

[HttpGet]

[ProducesResponseType(typeof(List<Employee>), 200)]

public ActionResult<List<Employee>> Get()

{

return Ok(GetStandardEmployeeList());

}

[HttpPost]

public IActionResult Post([FromBody] Employee employee)

{

// Add logic to save employee

return Ok(employee);

}

[HttpPut]

public IActionResult Put([FromBody] Employee employee)

{

// Add logic to update employee

return Ok(employee);

}

}

Employee.cs:

namespace EmployeeApi.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; }

}

}

Department.cs:

namespace EmployeeApi.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:

namespace EmployeeApi.Models

{

public class Skill

{

public int Id { get; set; }

public string Name { get; set; }

}

}

Program.cs:

using Microsoft.Extensions.DependencyInjection;

using Swashbuckle.AspNetCore.SwaggerGen;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

// Add services to the container.

builder.Services.AddControllers();

// Learn more about configuring OpenAPI at https://aka.ms/aspnet/openapi

builder.Services.AddOpenApi();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.MapOpenApi();

}

app.UseAuthorization();

app.MapControllers();

if (app.Environment.IsDevelopment())

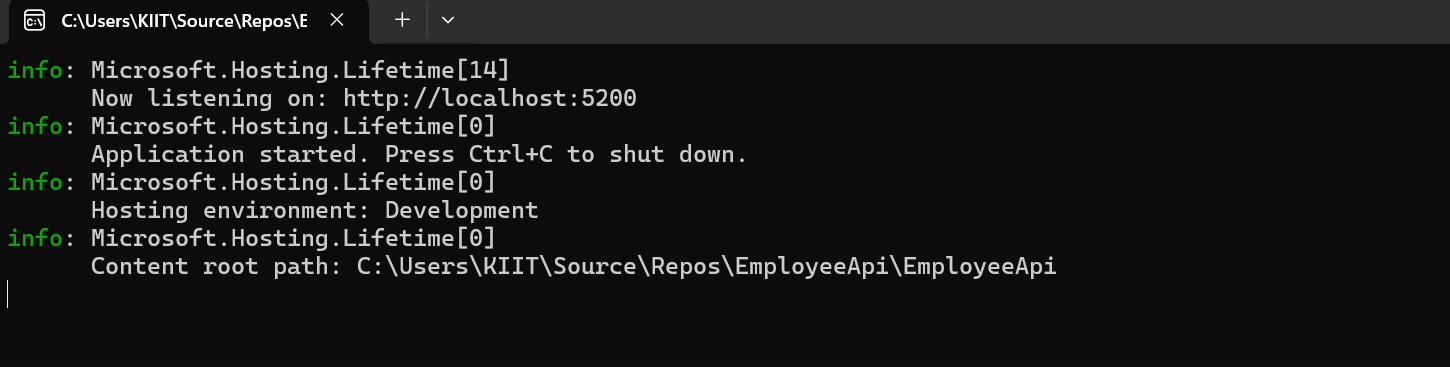
{

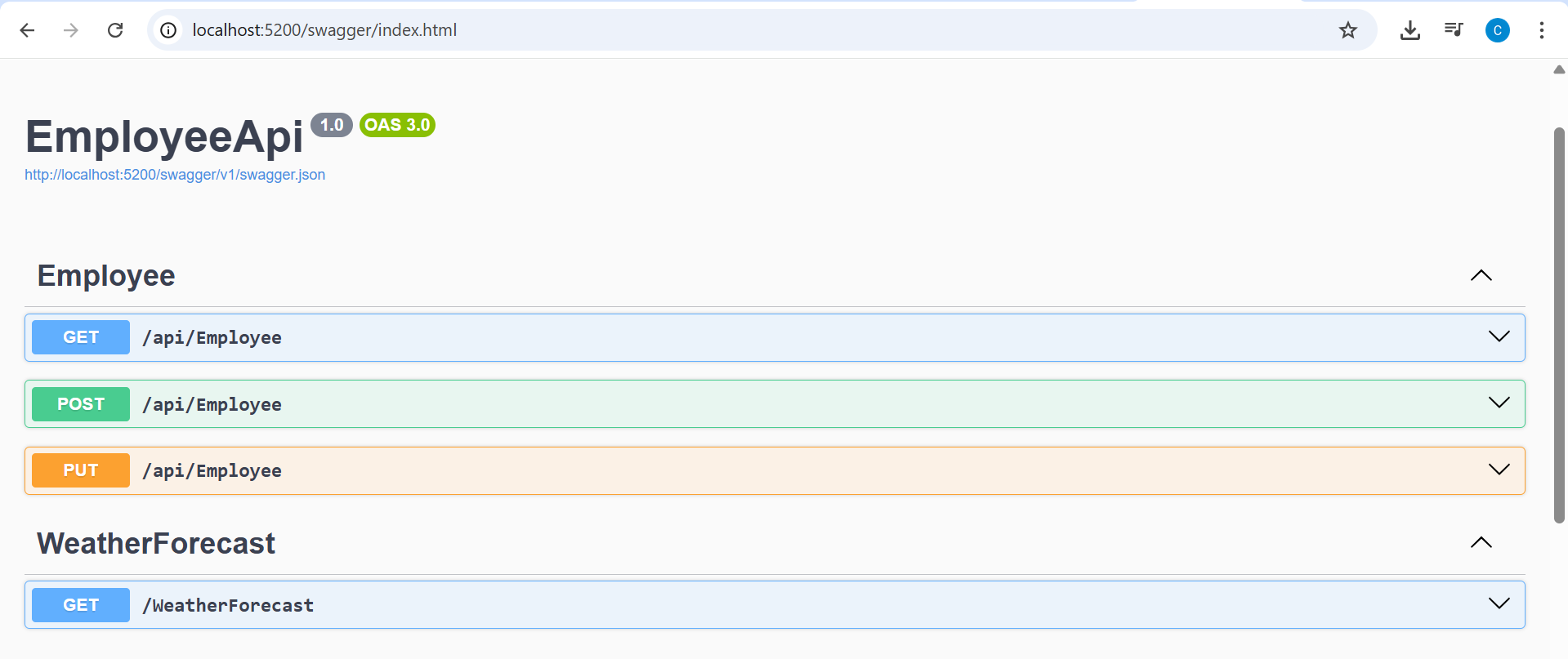
app.UseSwagger();

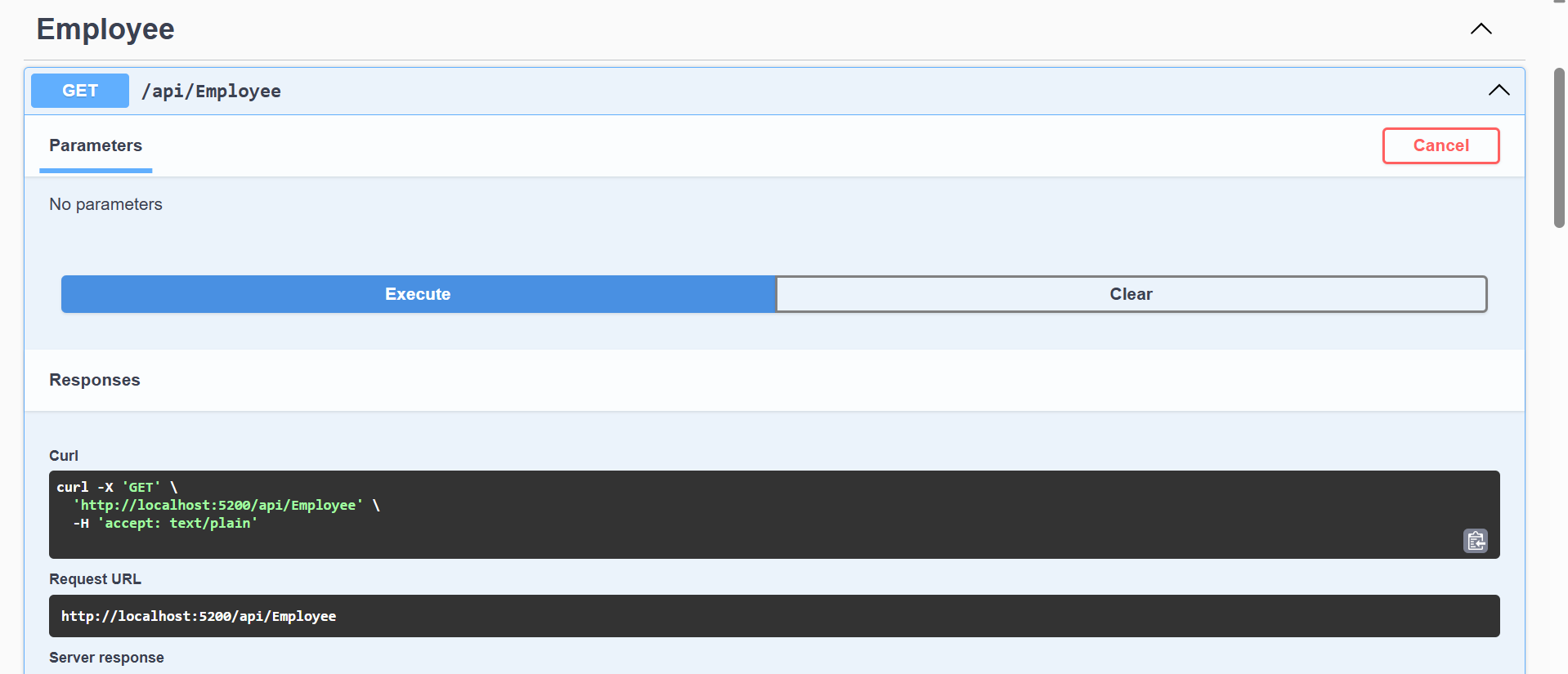
app.UseSwaggerUI();

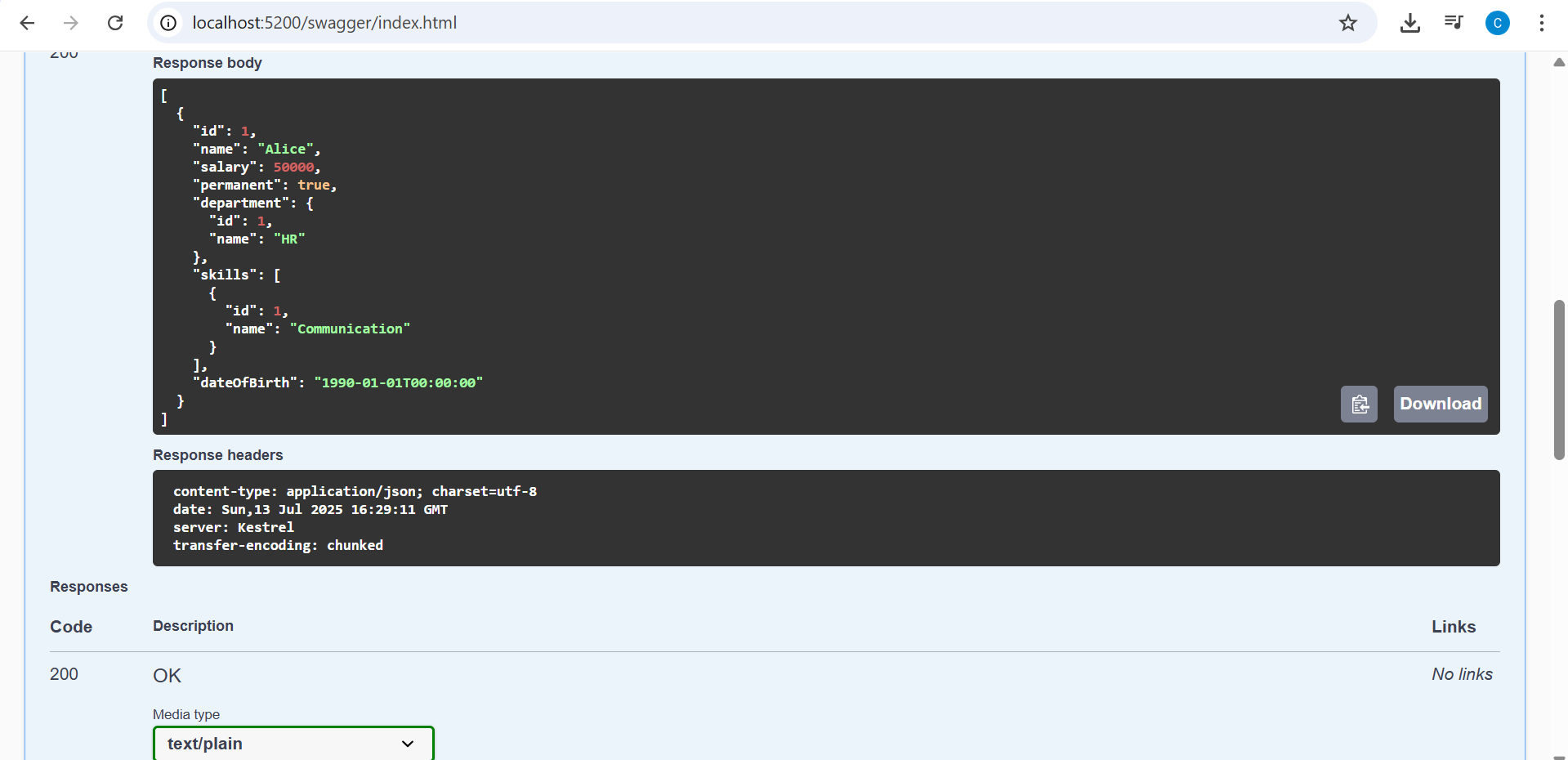
}

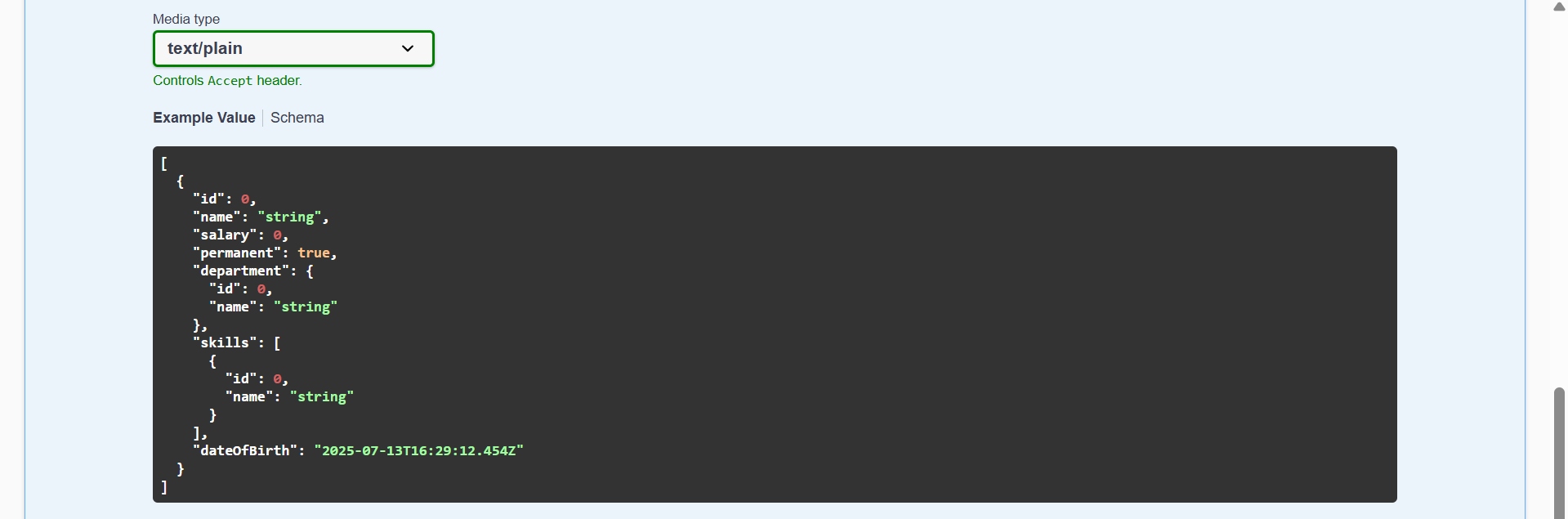
app.Run();











Q4)

EmployeeController.cs:

using EmployeeApi2.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

using System.Linq;

[ApiController]

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

public class EmployeesController : ControllerBase

{

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

{

new Employee { Id = 1, Name = "John", Department = "HR", Salary = 50000 },

new Employee { Id = 2, Name = "Jane", Department = "IT", Salary = 60000 },

new Employee { Id = 3, Name = "Bob", Department = "Finance", Salary = 55000 }

};

[HttpPut("{id}")]

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

{

if (id <= 0)

{

return BadRequest("Invalid employee id");

}

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

if (employee == null)

{

return BadRequest("Invalid employee id");

}

// Update the employee details

employee.Name = updatedEmployee.Name;

employee.Department = updatedEmployee.Department;

employee.Salary = updatedEmployee.Salary;

return Ok(employee);

}

}

Program.cs:

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

// Learn more about configuring OpenAPI at https://aka.ms/aspnet/openapi

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

builder.Services.AddOpenApi();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.MapOpenApi();

}

app.UseAuthorization();

app.MapControllers();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(options =>

{

options.SwaggerEndpoint("/swagger/v1/swagger.json", "Employee API v1");

// options.RoutePrefix = string.Empty; // Optional: Swagger UI at root

});

}

app.Run();

Employee.cs:

namespace EmployeeApi2.Models

{

public class Employee

{

public int Id { get; set; }

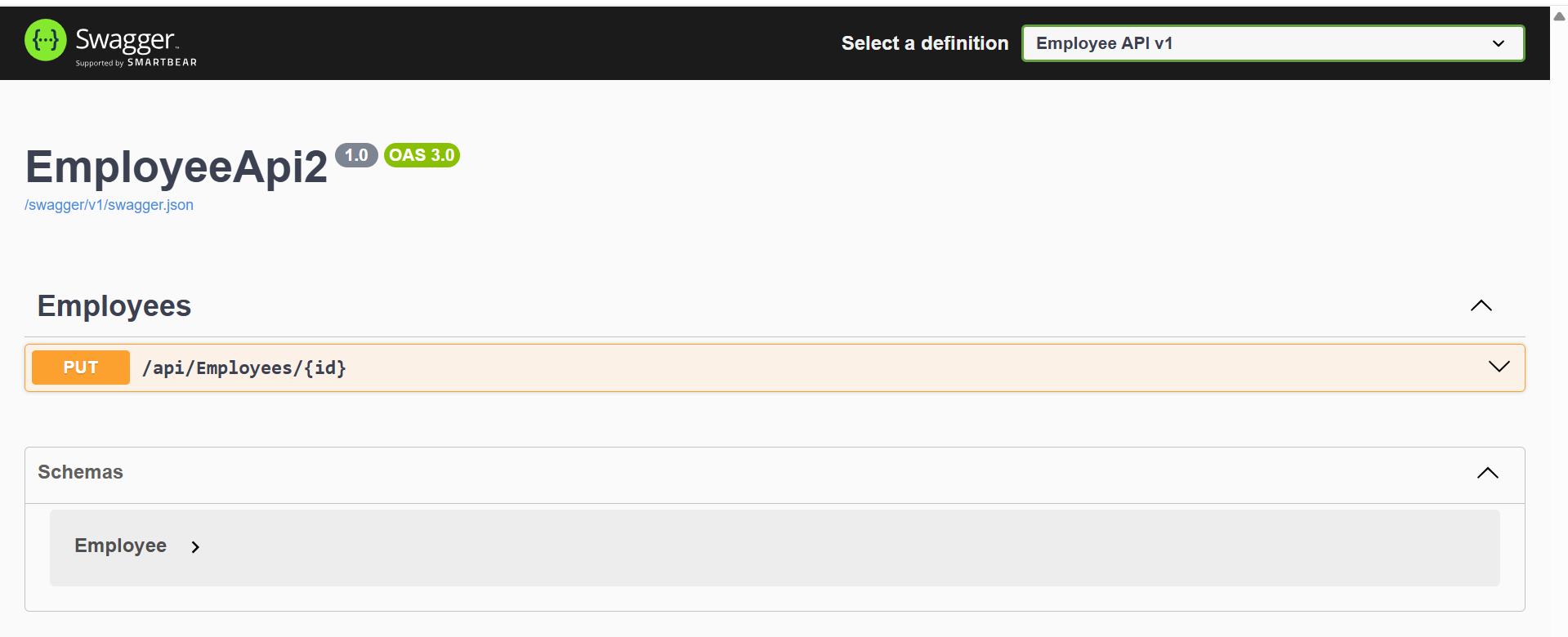
public string Name { get; set; }

public string Department { get; set; }

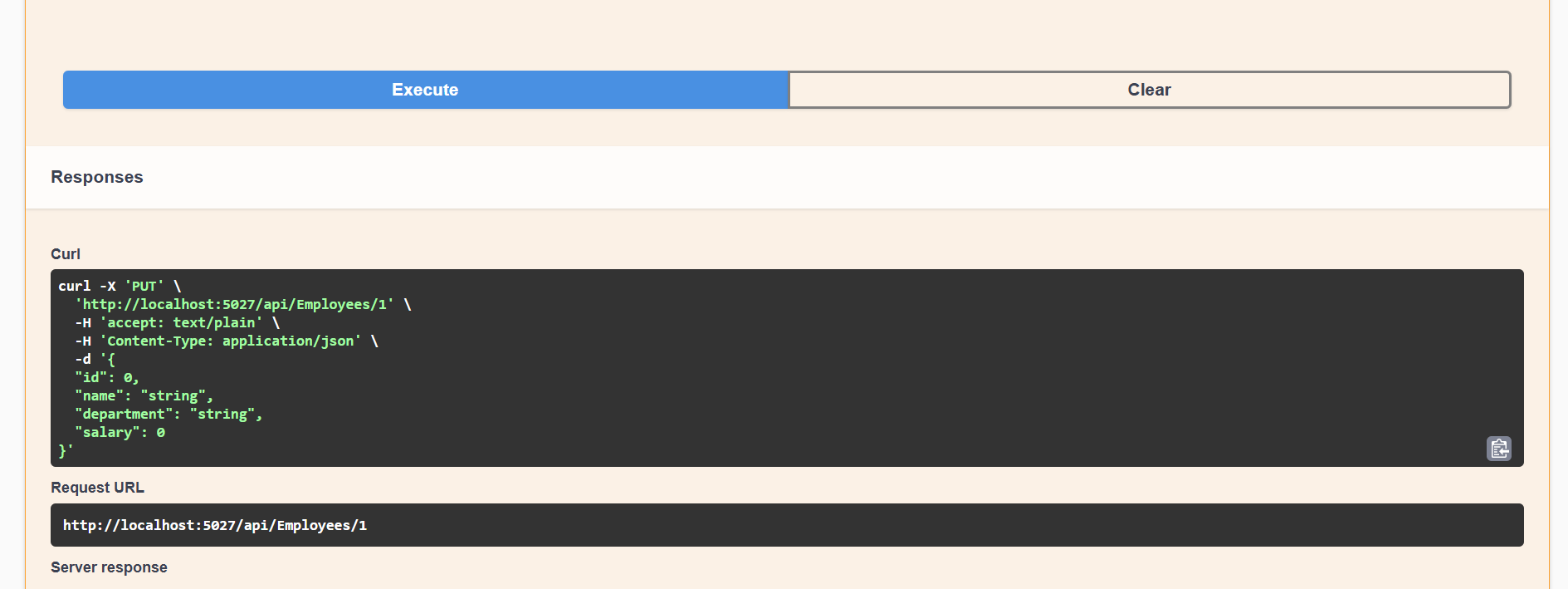
public double Salary { get; set; }

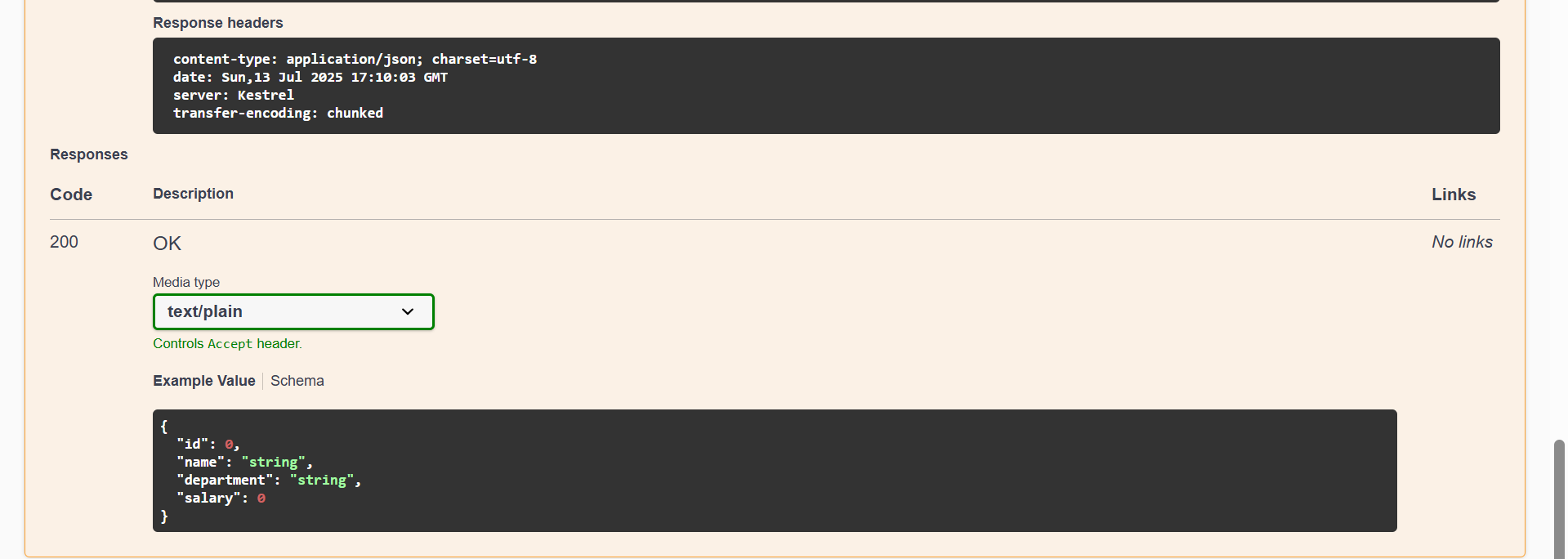
}

}









Q5)

What is CORS?

CORS (Cross-Origin Resource Sharing) is a browser mechanism to allow/disallow requests from different domains (e.g., your frontend app running on localhost:3000 accessing Web API at localhost:5000).

When your API runs on localhost:5000 and your frontend (such as React, Angular, or plain HTML/JS) runs on localhost:3000, it is imperative.

How to enable CORS thru Startup.cs, Install Cors nuget package to Web API application:

Launch the NuGet Package Manager, then install:

Microsoft.AspNetCore.Cors

Program.cs

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

// 🔐 JWT Configuration

var securityKey = "mysuperdupersecret"; // Must match AuthController

var key = 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 = key

};

});

// 🔧 Register Controllers

builder.Services.AddControllers();

var app = builder.Build();

// 🔧 Middleware pipeline

app.UseRouting();

app.UseAuthentication(); // 🔐 Must come before UseAuthorization

app.UseAuthorization();

app.MapControllers(); // 🗺️ Maps [ApiController]s like AuthController

app.Run();

