1. A service built on the foundations of Representational State Transfer (REST) is known as a RESTful Web Service. It operates on resources, which are usually represented in XML or JSON, using standard HTTP methods. REST places a strong emphasis on stateless communication, which means that the server does not save client context in between requests and that each request includes all the information required to process it.

Web API: An Application Programming Interface that can be accessed via HTTP on the web. Web APIs can have different architectural styles or be RESTful. They make endpoints accessible so that clients can engage with the functionality and data of applications.

An application built using the microservice architectural style consists of discrete, small services that are each dedicated to a particular business function. Because each microservice can have its own database and RESTful API, it can be deployed and scaled independently.

Features of REST architecture :

* Clients use standard HTTP methods to interact with resources that are represented by URLs.
* Statelessness: Every client-server request must include all the data required to comprehend and handle the request.
* Messages: HTTP requests and responses, frequently utilizing JSON or XML, are used for communication.
* Microservice Concept: To enable autonomous development and deployment, microservices are frequently connected via RESTful APIs.

WebAPI vs. WebService:

Any service that is accessible via the internet (such as SOAP-based) is referred to as a webservice.

RESTful APIs, which are lighter and use HTTP directly, are frequently referred to as WebAPI.

Not Limited to XML: RESTful APIs can return data in a number of formats, including plain text, HTML, XML, and most frequently JSON.

HTTP Request and HTTP Response:

The incoming request from a client to the server is represented by the HttpRequest object. It includes details like the URL, headers, HTTP method, and body (client-sent data).

The server's answer to the client's request is represented by the HttpResponse. It consists of the body (data sent back to the client), headers, and the status code.

Action verbs in Web API:

|  |  |  |
| --- | --- | --- |
| Verb | Meaning | Attribute in Web API |
| GET | Retrieve data from the server | [HttpGet] |
| POST | Create new data on the server | [HttpPost] |
| PUT | Update existing data on the server | [HttpPut] |
| DELETE | Remove data from the server | [HttpDelete] |

Typical HTTP Status Codes for Web APIs:

|  |  |  |
| --- | --- | --- |
| Status | Meaning | ActionResult Type |
| 200 OK | Request Succeeded | Ok() |
| 400 Bad Request | Client sent invalid data | BadRequest() |
| 401 Unauthorized | Authentication required/failed | Unauthorized() |
| 500 Internal Server Error | Server encountered an error | InternalServerError() |

Structure of a web api:

* Controller:A basic web API controller's structure is derived from ApiController (in.NET Core, from ControllerBase).
* Action Verbs: An HTTP verb attribute is used to adorn each method.
* Action methods: are those that deal with incoming HTTP requests.

Configuration files of WebAPI:

Startup.cs: Manages dependency injection, middleware configuration, and application startup.

appSettings.json:Application settings, including connection strings and custom configuration, are stored in appSettings.json.

launchSettings.json: Contains environment variables and profiles that control how the application is launched.

Route.config & WebAPI.config (in .NET 4.5):In older.NET Framework projects, the files Route.config and WebAPI.config (in.NET 4.5) are used for routing and general Web API configuration.

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]

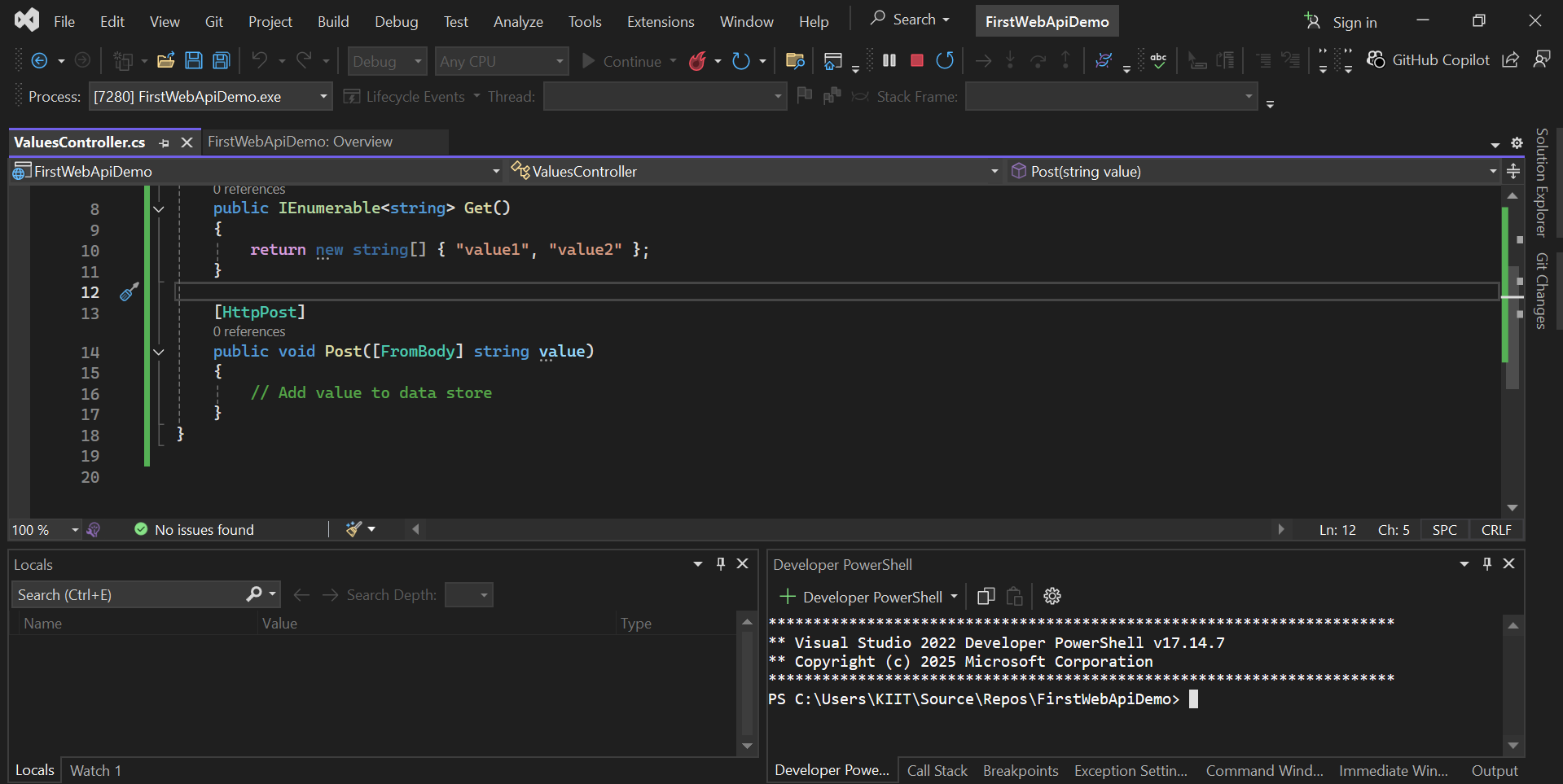
public void Post([FromBody] string value)

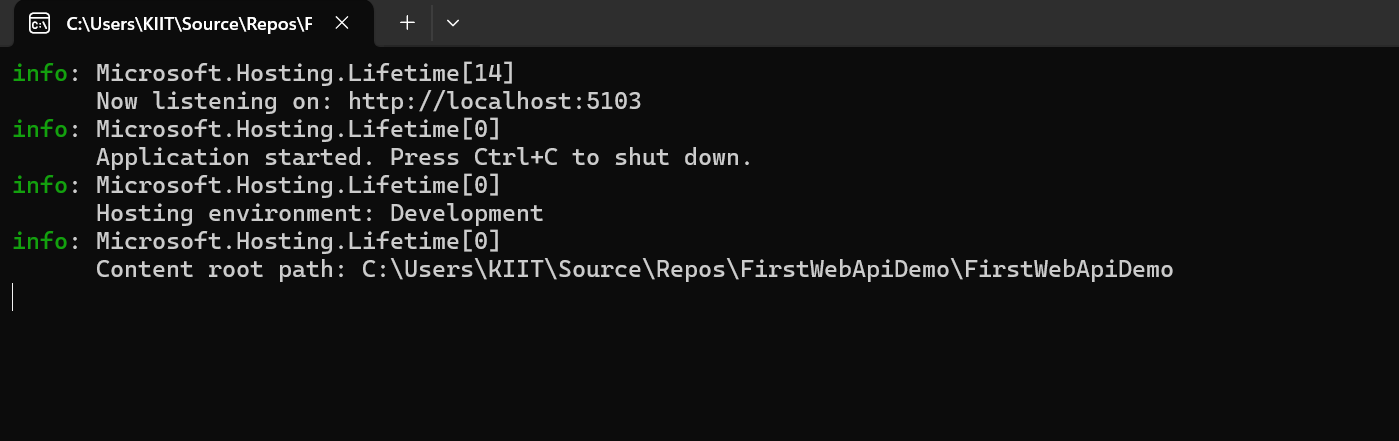
{

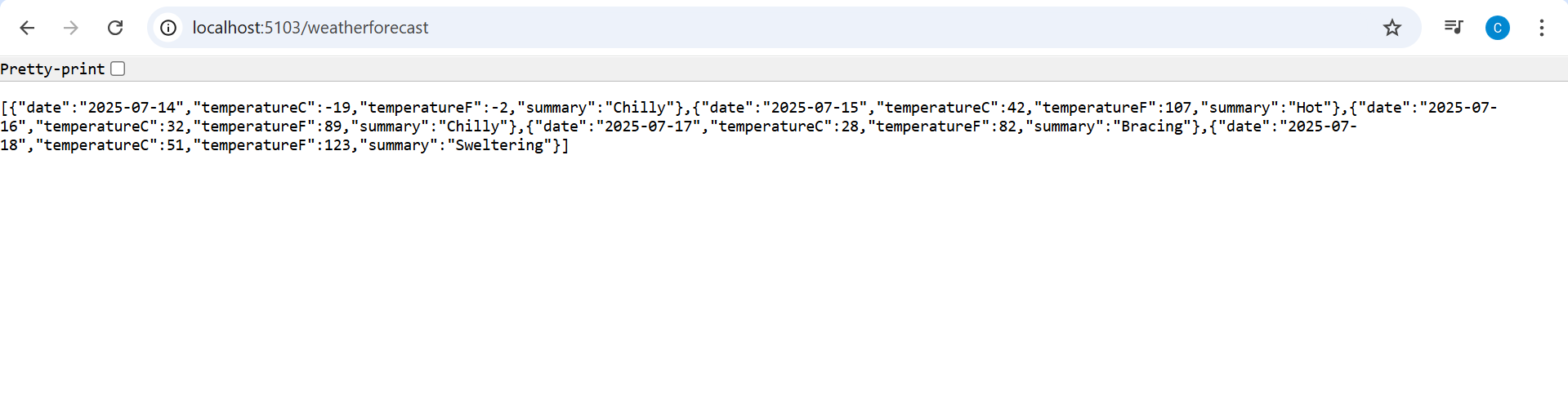
// Add value to data store

}

}

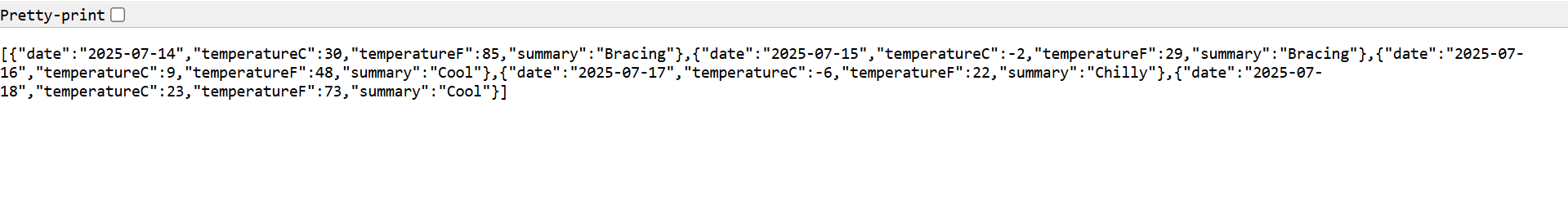


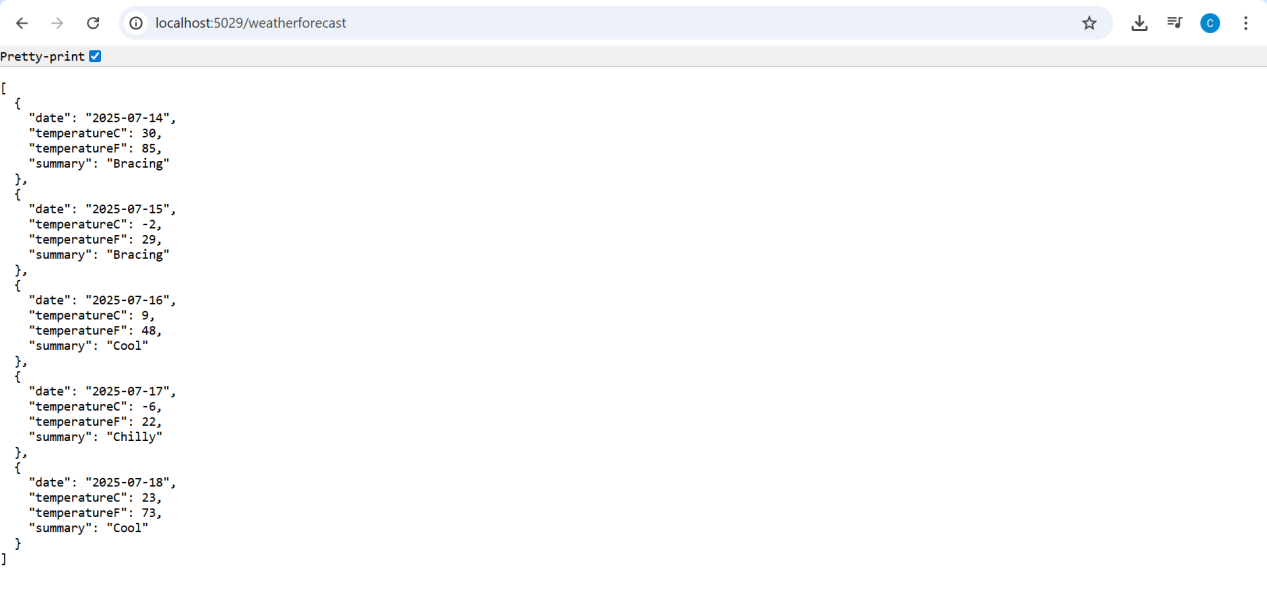


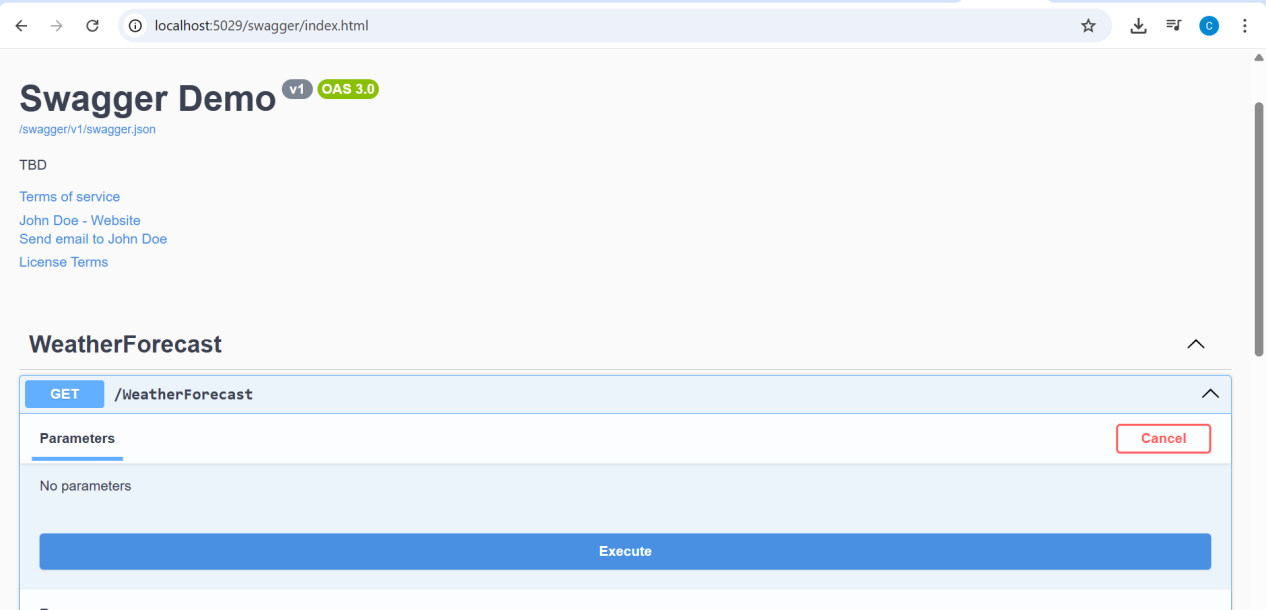


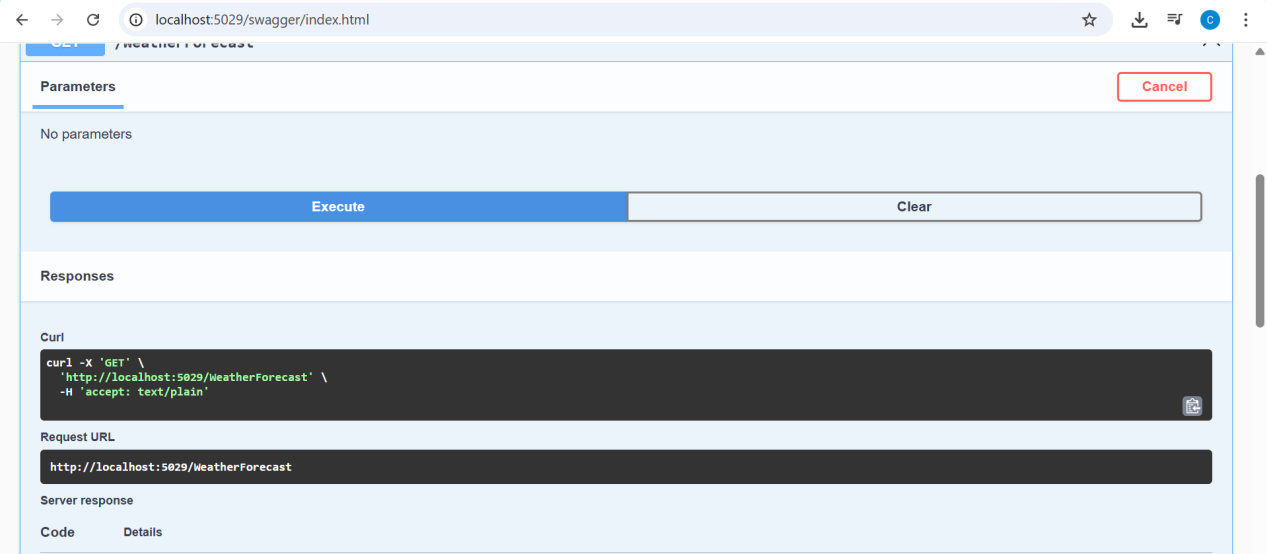


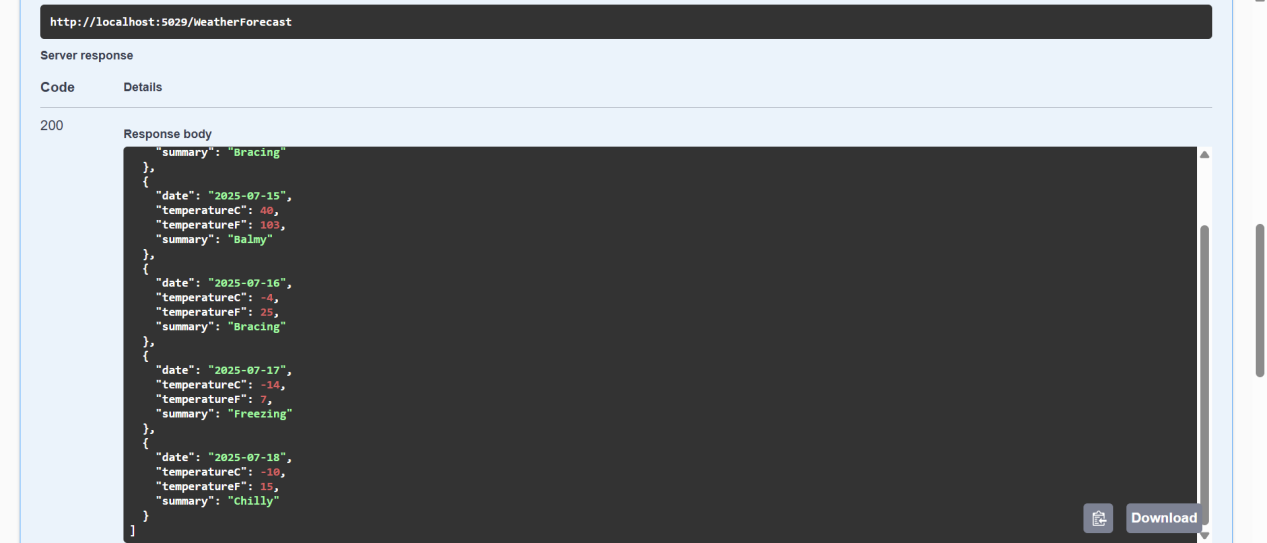
Q2)

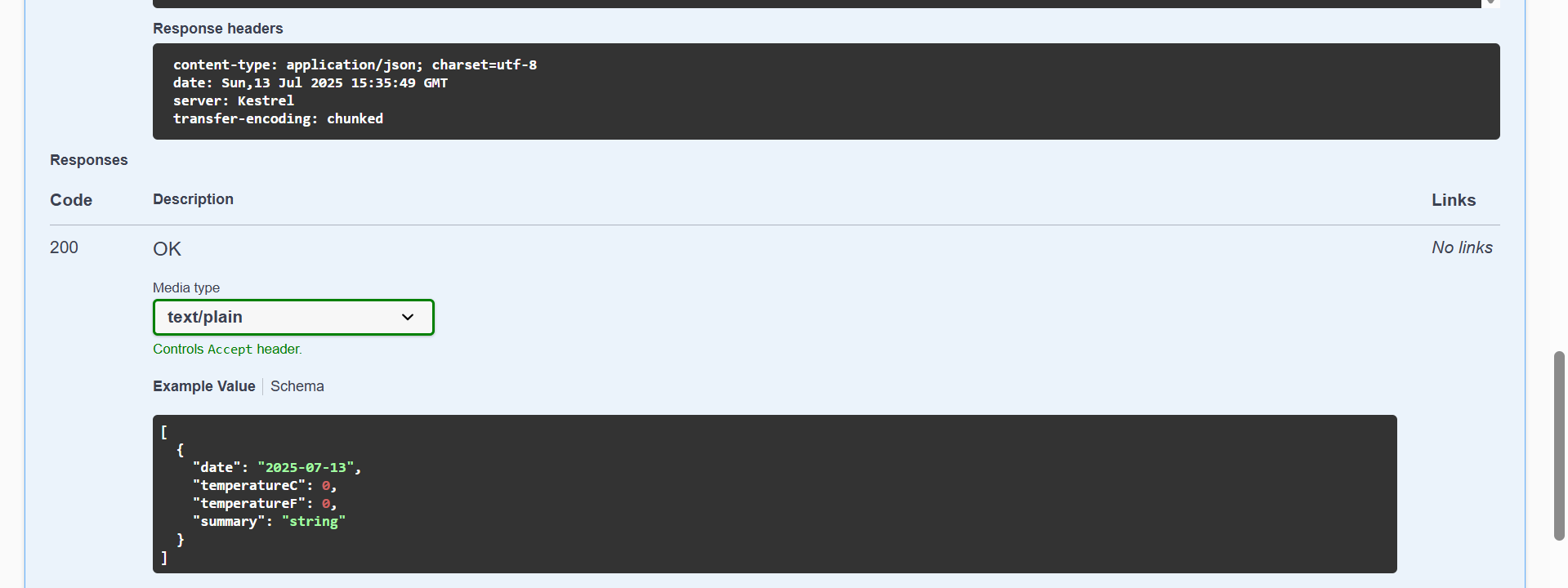




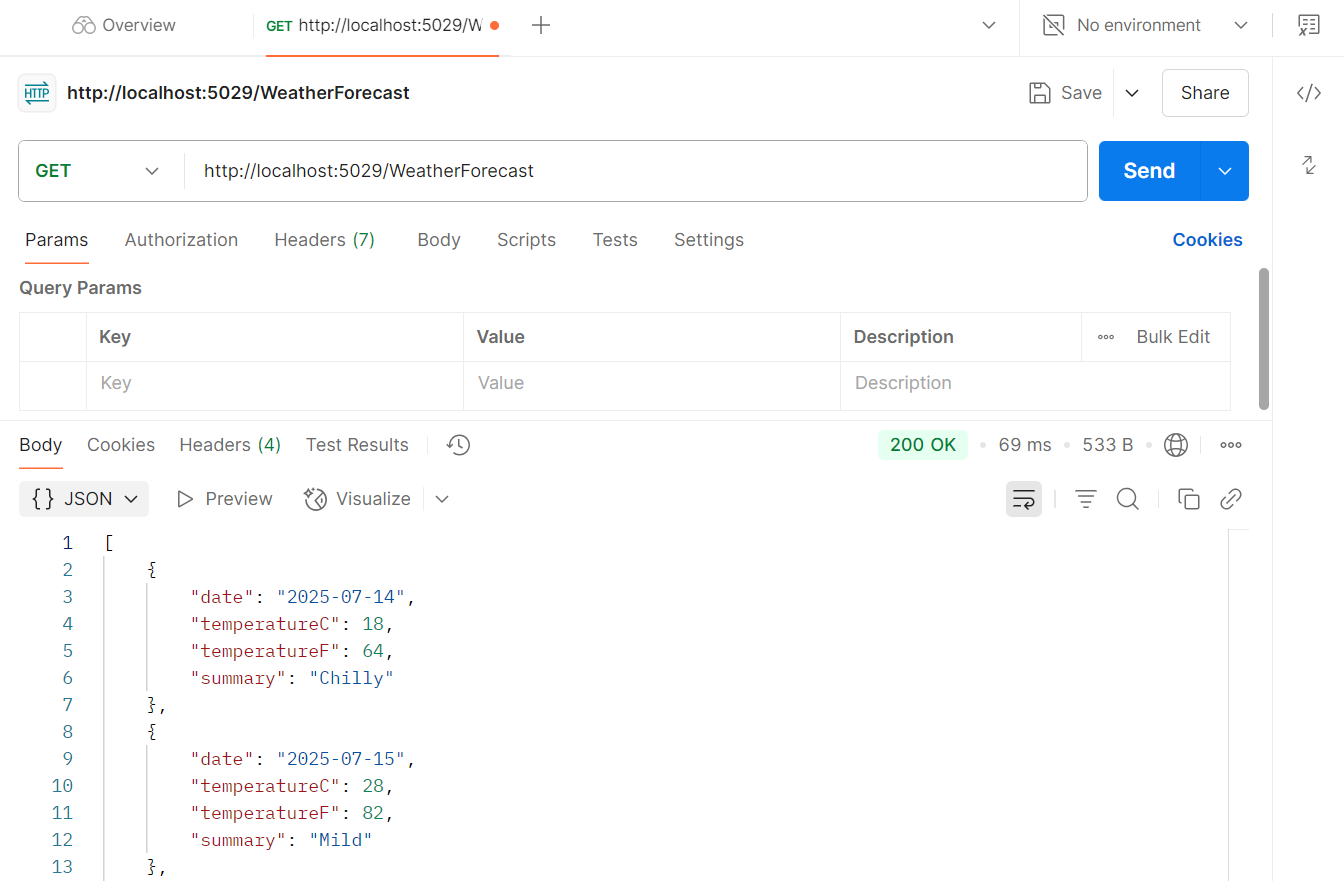








POSTMAN:



{}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"

    }

]