**WEBAPI HANDSON  
  
Objectives:**

* Explain the concept of RESTful web service, Web API & Microservice
  + Features of REST architecture - Representational State Transfer, Stateless, Messages, Concept of Microservice, Difference between WebService & WebAPI, Not restricted to send XML as response
* Explain what is HttpRequest & HttpResponse
* List the types of Action Verbs
  + HttpGet, HttpPost, HttpPut, HttpDelete - Meaning of action verbs and how that should be declared as attributes for Web API
* List the types of HttpStatusCodes used in WebAPI
  + Ok, InternalServerError, Unauthorized, BadRequest - All thru the action result types
* Demonstrate creation of a simple WebAPI - With Read, Write actions
  + Structure of a web api - Controller & its inheritance from ApiController, Action verbs, Action method
* Explain the types of Configuration files of WebAPI
  + Startup.cs with depdency injection, appSettings.json, launchSettings.json, Explain Route.config & WebAPI.config in .Net 4.5 framework

1. **First Web Api using .Net core**

Create a .Net core web application with API template. Use the option to create controller with Read Write permissions. Notice the ValuesController creation with Action methods corresponding to the Action verbs.

On creation of the Web API, execute the application and check if the GET action method result is returned as expected.

**SOLUTION :**

**1. Objective**

To create and test a basic ASP.NET Core Web API using Microsoft Visual Studio 2022 with support for RESTful services and Swagger UI.

**2. Key Concepts Covered**

- RESTful architecture  
- Web API structure (Controllers, Models)  
- HTTP request and response  
- Action verbs: GET, POST, PUT, DELETE  
- HTTP status codes: 200 OK, 400 Bad Request, etc.  
- Testing API with Swagger UI

**3. Tools Used**

|  |  |
| --- | --- |
| Tool | Version |
| Visual Studio | 2022 |
| .NET SDK | 6.0 or 7.0 |
| Browser | Edge / Chrome |
| Swagger | OpenAPI 3.0 |

**4. Step-by-Step Implementation**

**Step 1: Create ASP.NET Core Web API Project**

- Selected ASP.NET Core Web API template  
- Enabled Controllers (not Minimal API)  
- Enabled Swagger/OpenAPI support

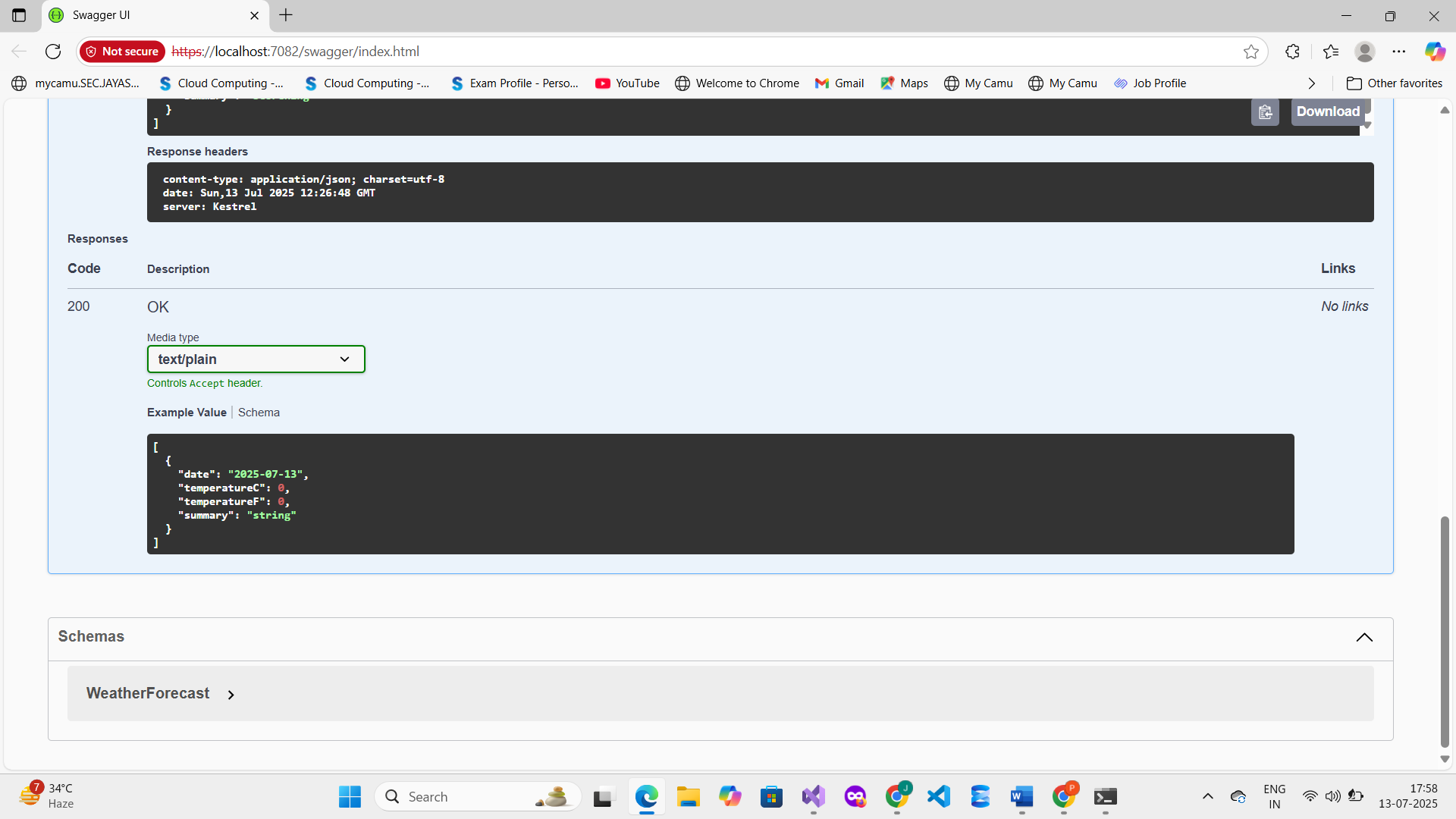
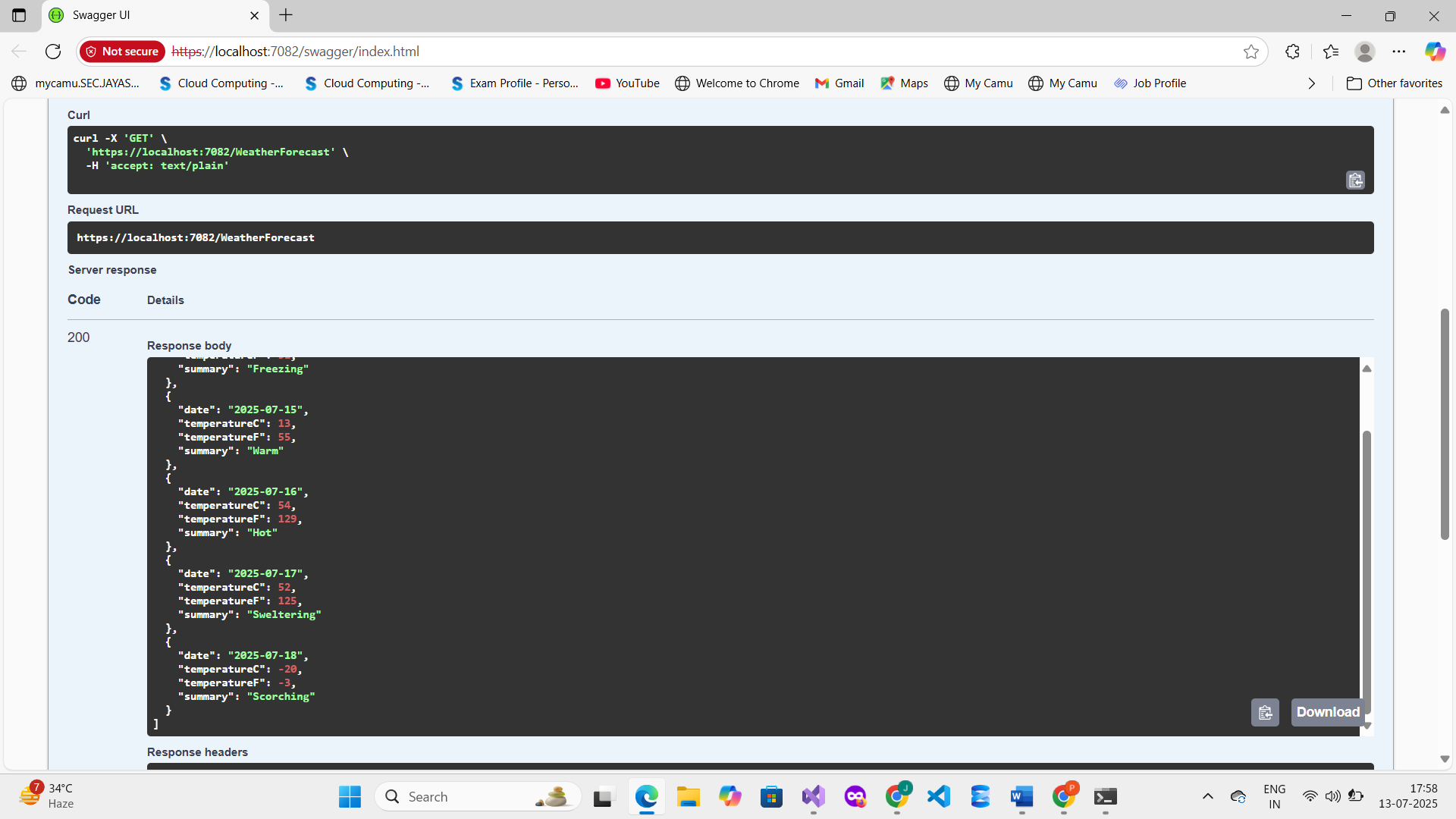
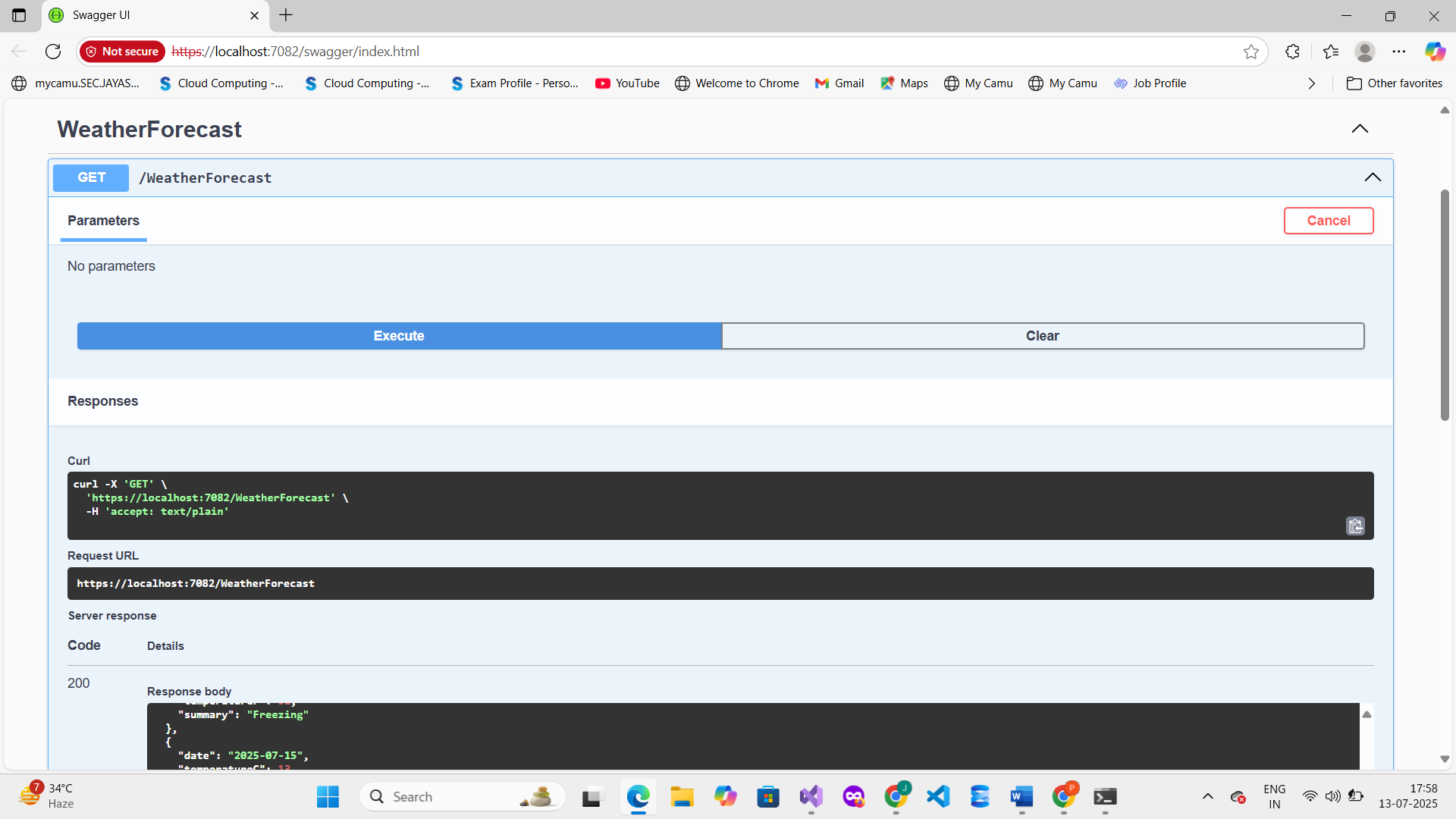
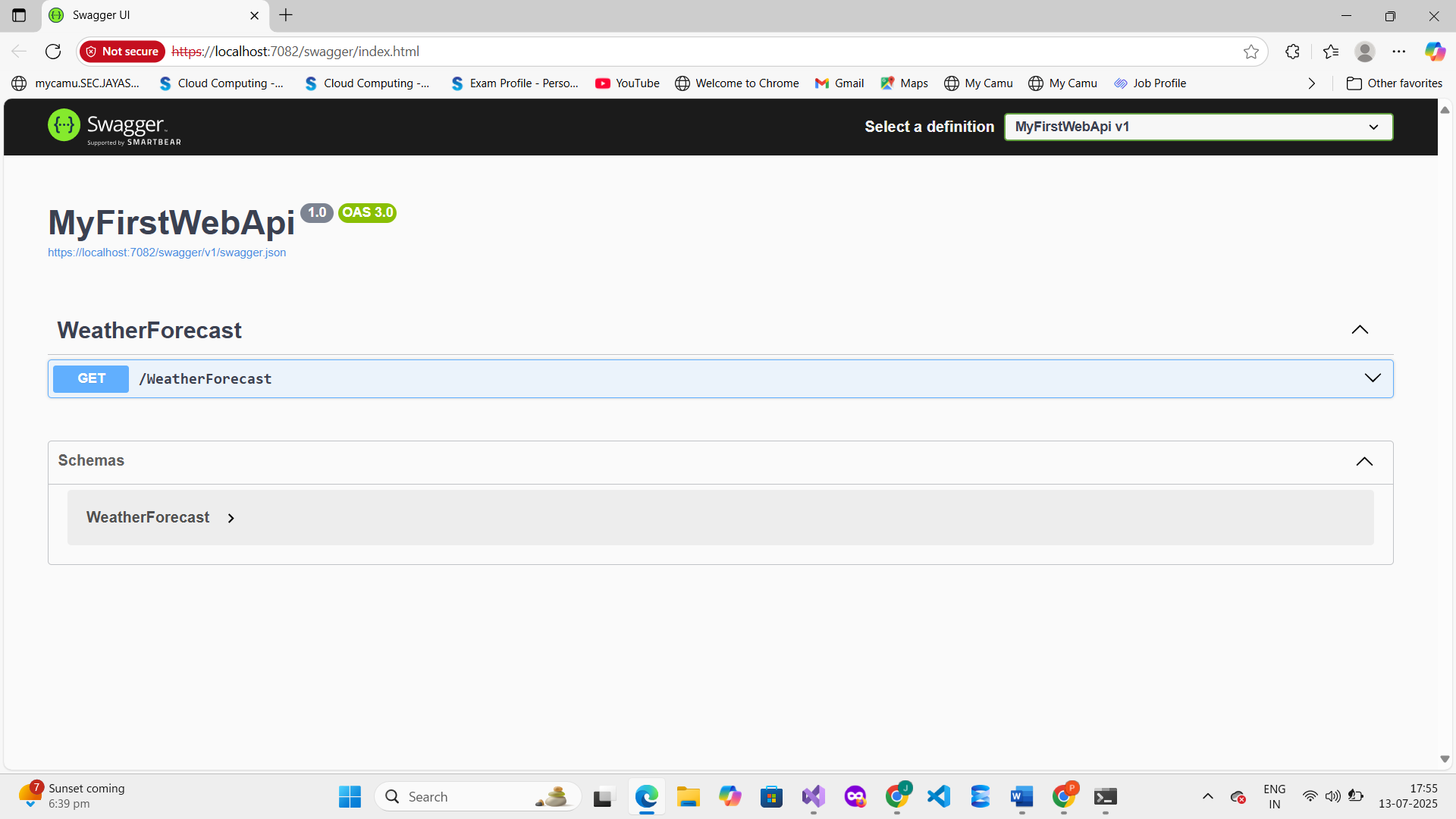
**Step 2: Project Structure**

- Created project named MyFirstWebApi  
- Auto-generated files:  
 - WeatherForecast.cs  
 - WeatherForecastController.cs  
 - Program.cs  
 - appsettings.json  
 - launchSettings.json

**Step 3: Run and Test the API**

- Pressed F5 to start debugging  
- Swagger UI opened at:  
 https://localhost:7082/swagger/index.html  
  
- Clicked GET /WeatherForecast  
- Clicked Try it out > Execute  
- API returned sample data with status 200 OK

**5. Output**



**6. Conclusion**

The Web API was successfully created, executed, and tested using Swagger UI.   
This exercise demonstrated the use of HTTP GET verbs, RESTful routing, and integration of Swagger in a .NET Core Web API project.