**1. 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.

Program:

using Microsoft.AspNetCore.Mvc;

namespace FirstWebAPI.Controllers

{

[ApiController]

[Route("[controller]")]

public class WeatherForecastController : ControllerBase

{

private static readonly string[] Summaries = new[]

{

"Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching"

};

private readonly ILogger<WeatherForecastController> \_logger;

public WeatherForecastController(ILogger<WeatherForecastController> logger)

{

\_logger = logger;

}

[HttpGet]

public IEnumerable<WeatherForecast> Get()

{

return Enumerable.Range(1, 5).Select(index => new WeatherForecast

{

Date = DateOnly.FromDateTime(DateTime.Now.AddDays(index)),

TemperatureC = Random.Shared.Next(-20, 55),

Summary = Summaries[Random.Shared.Next(Summaries.Length)]

})

.ToArray();

}

}

}

Output:

