# Assignment 1

# Objectives

## 1. Explain the Concept of RESTful Web Service, Web API & Microservice

REST (Representational State Transfer) is an architectural style that uses standard HTTP methods to interact with resources.  
Features of REST:  
- Stateless: Each request is independent; no session info stored.  
- Resources: Identified via URIs (e.g., /api/products)  
- Methods (verbs): Uses HTTP methods (GET, POST, PUT, DELETE)  
- Flexible Formats: Can return JSON, XML, etc. (not limited to XML)  
- Stateless communication over HTTP

Web API:  
- A framework for building HTTP services using .NET Core.  
- Allows exposing data and functions through HTTP endpoints.

Microservice:  
- A small, independently deployable service that performs a specific task.  
- Communicates via REST APIs or messaging systems.

Difference: Web Service vs Web API  
Web Service | Web API  
--------------- | ------------------  
Supports only XML | Supports JSON, XML, etc.  
Heavier | Lightweight  
SOAP-based | RESTful  
Platform-independent | Language/platform-independent

## 2. What is HttpRequest & HttpResponse?

HttpRequest: Carries the client's request information like method (GET/POST), headers, URL, and body.  
HttpResponse: The server's reply with status code, headers, and content (JSON/XML/data).

## 3. Types of Action Verbs (HTTP Methods)

Verb | Purpose | Usage  
----------|--------------|-------------------------  
HttpGet | Fetch data | [HttpGet]  
HttpPost | Create data | [HttpPost]  
HttpPut | Update data | [HttpPut("{id}")]  
HttpDelete| Delete data | [HttpDelete("{id}")]

## 4. Types of HttpStatusCode in Web API

Status Code | Meaning | Used in  
------------|--------------------|------------------  
200 OK | Success | return Ok();  
400 BadRequest | Client error | return BadRequest();  
401 Unauthorized | Auth failed | return Unauthorized();  
500 InternalServerError | Server error | return StatusCode(500);

## 5. Create a Simple Web API in .NET 8

In VS Code Terminal:

dotnet new webapi -n FirstWebAPI  
cd FirstWebAPI  
code .

**Program.cs:**

var builder = WebApplication.CreateBuilder(args);  
builder.Services.AddControllers();  
var app = builder.Build();  
app.UseHttpsRedirection();  
app.MapControllers();  
app.Run();

Controllers/ValuesController.cs:

using Microsoft.AspNetCore.Mvc;  
using System.Collections.Generic;  
  
namespace FirstWebAPI.Controllers  
{  
 [ApiController]  
 [Route("api/[controller]")]  
 public class ValuesController : ControllerBase  
 {  
 [HttpGet]  
 public ActionResult<IEnumerable<string>> Get()  
 {  
 return Ok(new string[] { "value1", "value2" });  
 }  
  
 [HttpGet("{id}")]  
 public ActionResult<string> Get(int id)  
 {  
 return Ok("value" + id);  
 }  
  
 [HttpPost]  
 public IActionResult Post([FromBody] string value)  
 {  
 return Ok("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 value with id {id}");  
 }  
 }  
}

## Test URLs

Method | URL  
-------|-------------------------------  
GET | http://localhost:5249/api/values  
POST | (Used Postman with body "hello")  
PUT | http://localhost:5249/api/values/1  
DELETE | http://localhost:5249/api/values/1

## 6. Configuration Files in .NET 8 Web API

File | Purpose  
----------------------|------------------------------  
Program.cs | Startup logic & middleware  
appsettings.json | App-level config   
launchSettings.json | Debug URL/port info

Legacy .NET Framework (4.5):  
File | Purpose  
------------------|-----------------------------  
WebApiConfig.cs | API route registration  
RouteConfig.cs | MVC route setup

## Summary

- Created .NET 8 Web API project using `dotnet new webapi`  
- Added custom controller with full CRUD (GET, POST, PUT, DELETE)  
- Explained REST, Web API, Microservices, HTTP verbs, status codes  
- Discussed config files and runtime behavior

Project: FirstWebAPI (.NET 8)

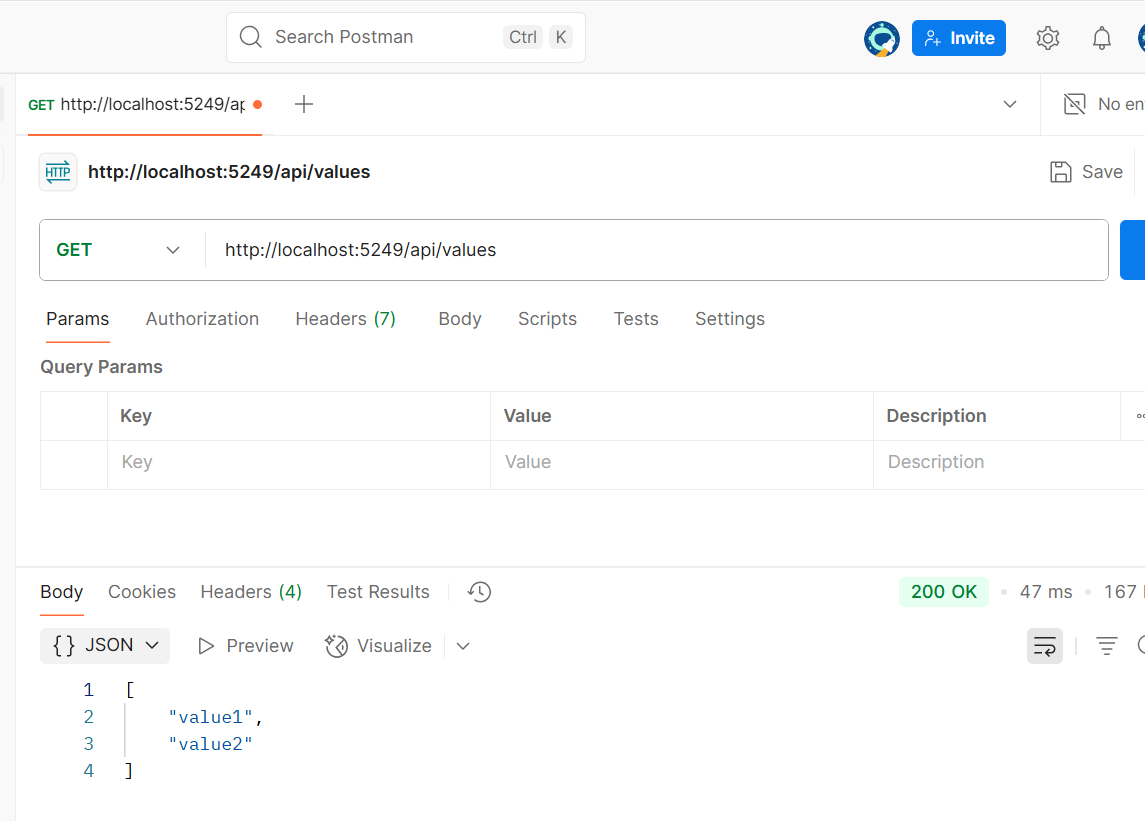
Student: Gyarina Rauniyar

This document provides a checklist of tested endpoints and placeholders for attaching screenshots of successful API responses.

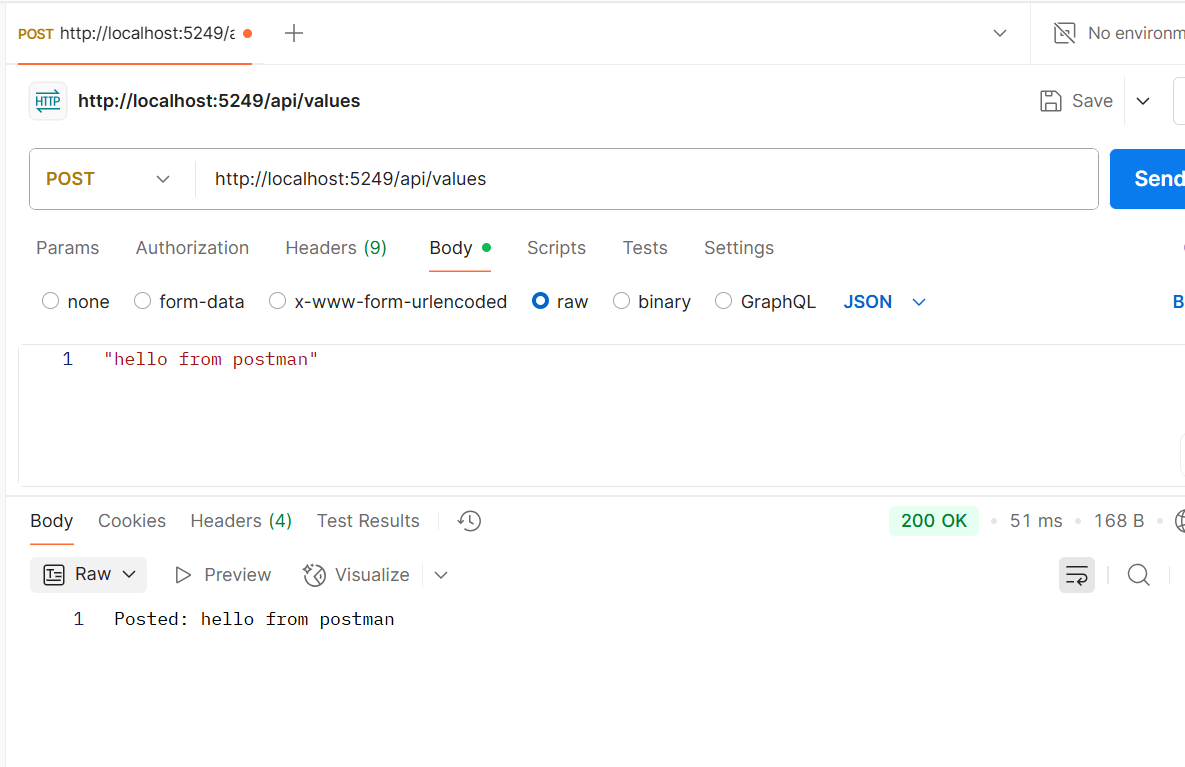
# ✅ Test Checklist

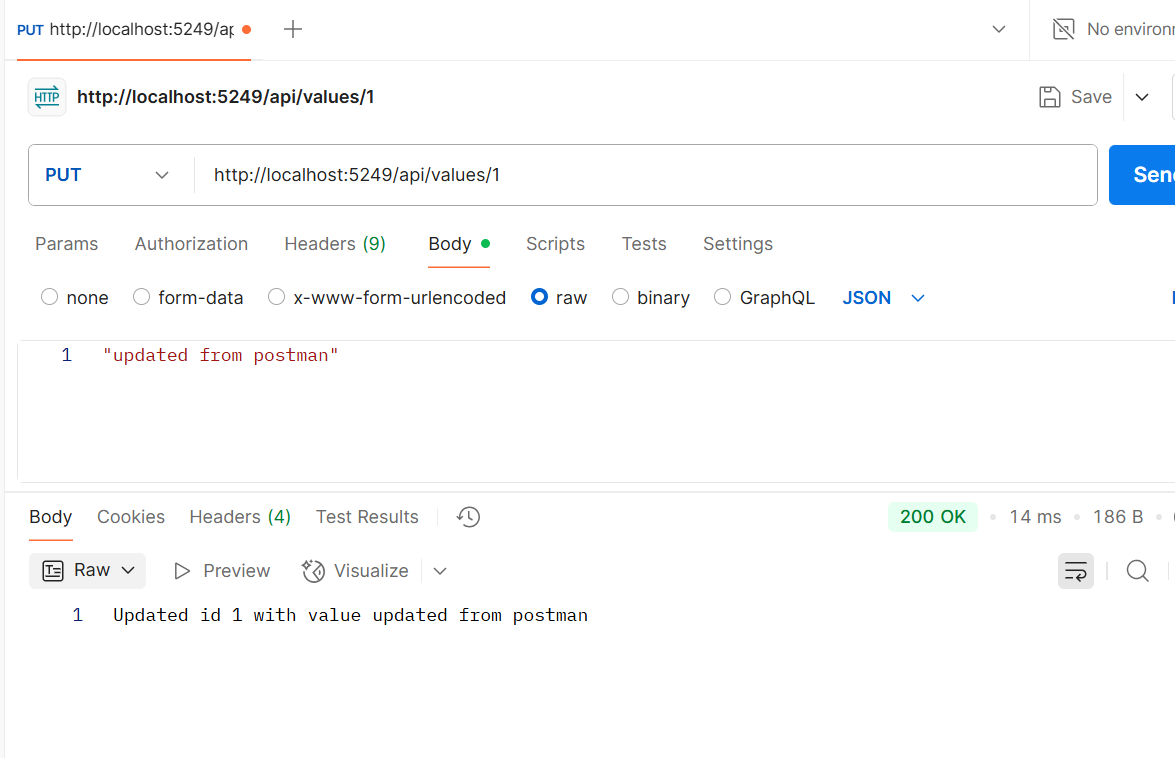
|  |  |  |
| --- | --- | --- |
| HTTP Method | Endpoint URL | Test Status |
| GET | http://localhost:5249/api/values | ✅ Passed |
| GET (by ID) | http://localhost:5249/api/values/1 | ✅ Passed |
| POST | http://localhost:5249/api/values | ✅ Passed |
| PUT | http://localhost:5249/api/values/1 | ✅ Passed |
| DELETE | http://localhost:5249/api/values/1 | ✅ Passed |

# Screenshot Placeholders

1. Screenshot of `GET /api/values` response:

2. Screenshot of `POST /api/values` with body "hello":



3. Screenshot of `PUT /api/values/1` with body "update":

1. Screenshot of `DELETE /api/values/1`:

