**Eplain the Concept of RESTful Web Service, Web API & Microservice**

**RESTful Web Service:** A service based on REST architecture that uses standard HTTP methods (GET, POST, PUT, DELETE) to perform CRUD operations. It treats everything as a resource and exposes URIs for each.

**Web API:** A framework provided by .NET (or similar platforms) to build HTTP services. It allows clients (like browsers, mobile apps) to interact with server data using RESTful conventions.

**Microservice:** An architectural style where the application is split into small, independent services. Each service handles a specific business functionality and communicates over HTTP.

**Features of REST Architecture:**

**Representational State Transfer:** Each resource has a unique URI; operations are performed via standard HTTP methods.

**Stateless:** Each HTTP request is independent. The server does not store client session information.

**Messages:** Data is sent over HTTP in standard formats like JSON or XML.

**Flexible Data Formats:** Unlike traditional Web Services, REST APIs can return JSON, XML, or any format.

**WebService vs WebAPI:**

**WebService:** Based on SOAP, returns XML, more rigid.

**WebAPI:** Based on REST, returns JSON/XML, lightweight and faster.

**What is HttpRequest & HttpResponse**

**HttpRequest:** Represents the incoming request from the client. It contains method type (GET/POST/etc.), headers, query strings, and body.

**HttpResponse:** Represents the server’s response. It contains status code, headers, and the actual content (data or error).

**Types of Action Verbs in Web API**

Each HTTP method maps to a controller action using attributes:

[HttpGet]: Used to retrieve data. Example: [HttpGet] public IActionResult Get()

[HttpPost]: Used to create a new resource. Example: [HttpPost] public IActionResult Post([FromBody] Data data)

[HttpPut]: Used to update an existing resource. Example: [HttpPut("{id}")]

[HttpDelete]: Used to delete a resource. Example: [HttpDelete("{id}")]

**Common HttpStatusCodes in Web API**

These are used to indicate the result of an API call:

200 OK: Request successful.

400 BadRequest: Request contains invalid data.

401 Unauthorized: User is not authenticated.

500 InternalServerError: Server-side issue.

Q5

[ApiController]

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

public class ValuesController : ControllerBase

{

private static List<string> values = new List<string> { "value1", "value2" };

[HttpGet]

public IActionResult Get() => Ok(values);

[HttpPost]

public IActionResult Post([FromBody] string value)

{

values.Add(value);

return Ok(values);

}

}

1. **Configuration Files in Web API**

Configures services (like DB context, dependency injection).

Defines middleware pipeline.

**appsettings.json**:

Stores configuration (DB connection string, keys, environment settings).

**launchSettings.json**:

Defines environment profiles, ports, SSL settings, and launch behavior.

**WebApiConfig.cs (in .NET Framework)**:

Sets up route templates for API endpoints.

**RouteConfig.cs (in .NET Framework)**:

Sets up MVC route templates (not for APIs directly).

**SETTING UP A WEBAPI**

Model(Product.cs)

namespace WebAPIHandsOn.Models

{

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

}

}

Controller(TestController.cs)

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using WebAPIHandsOn.Models;

namespace WebAPIHandsOn.Controllers

{

[ApiController]

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

public class ProductsController : ControllerBase

{

private static List<Product> products = new()

{

new Product { Id = 1, Name = "Pen" },

new Product { Id = 2, Name = "Notebook" }

};

[HttpGet]

public IActionResult Get() => Ok(products);

[HttpGet("{id}")]

public IActionResult GetById(int id)

{

var product = products.FirstOrDefault(p => p.Id == id);

if (product == null) return NotFound();

return Ok(product);

}

[HttpPost]

public IActionResult Post([FromBody] Product product)

{

product.Id = products.Count + 1;

products.Add(product);

return CreatedAtAction(nameof(GetById), new { id = product.Id }, product);

}

[HttpPut("{id}")]

public IActionResult Put(int id, [FromBody] Product updatedProduct)

{

var product = products.FirstOrDefault(p => p.Id == id);

if (product == null) return NotFound();

product.Name = updatedProduct.Name;

return Ok(product);

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

var product = products.FirstOrDefault(p => p.Id == id);

if (product == null) return NotFound();

products.Remove(product);

return NoContent();

}

}

}