WEEK 4

1. 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

**1. RESTful Web Service, Web API & Microservice**

**RESTful Web Service**

* **REST** stands for **Representational State Transfer**.
* It is an architectural style for designing networked applications.
* RESTful services use HTTP methods (GET, POST, PUT, DELETE) to perform operations on resources, typically represented as URLs.

**Key Features of REST Architecture**

* **Stateless:** Each request from a client contains all the information needed to process it. The server does not store any client context between requests.
* **Representations:** Resources can be represented in multiple formats (JSON, XML, HTML, etc.), but REST is not restricted to XML as older web services were.
* **Messages:** Communication is done via standard HTTP messages (requests and responses).
* **Uniform Interface:** REST uses a consistent, standardized interface for interacting with resources.
* **Cacheable:** Responses can be labelled as cacheable or non-cacheable to improve performance.

**Web API**

* A **Web API** is an interface that allows different software applications to communicate over the web using HTTP.
* Web APIs are often RESTful, but not all Web APIs follow REST strictly.
* Web APIs can return data in various formats (JSON is most common today).

**Microservice**

* **Microservice architecture** breaks down an application into small, independent services.
* Each microservice is responsible for a specific business function and communicates with others via APIs.
* Microservices are independently deployable and scalable.

**Differences: WebService vs WebAPI**

| **Feature** | **WebService (.asmx, SOAP)** | **Web API (RESTful)** |
| --- | --- | --- |
| Protocol | SOAP over HTTP | HTTP/HTTPS |
| Data Format | XML only | JSON, XML, others |
| Platform Dependency | Tightly coupled to .NET | Platform-agnostic |
| Lightweight | No | Yes |
| Browser Support | Limited | Excellent (JSON) |

1. Explain what is HttpRequest & HttpResponse

**HttpRequest:** Represents the data sent by the client to the server. Includes method (GET, POST, etc.), headers, body, and URL.

**HttpResponse:** Represents the data sent back by the server to the client. Includes status code, headers, and response body.

1. 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

These correspond to HTTP methods and define the type of operation:

* **HttpGet:** Retrieve data from the server (read-only).
* **HttpPost:** Submit new data to the server (create).
* **HttpPut:** Update existing data on the server.
* **HttpDelete:** Remove data from the server.

1. List the types of HttpStatusCodes used in WebAPI

| **Status Code** | **Name** | **Usage Example** |
| --- | --- | --- |
| 200 | Ok | Successful request |
| 400 | BadRequest | Invalid request data |
| 401 | Unauthorized | Authentication required |
| 500 | InternalServerError | Server-side error |

1. Demonstrate creation of a simple WebAPI - With Read, Write actions

[ApiController]

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

public class ValuesController : ControllerBase

{

[HttpGet]

public IActionResult GetValues() { ... }

[HttpPost]

public IActionResult AddValue() { ... }

}

1. Explain the types of Configuration files of WebAPI

**In .NET Core**

* **Startup.cs:** Configures services and middleware, including dependency injection.
* **appSettings.json:** Stores configuration settings (e.g., connection strings, app-specific settings).
* **launchSettings.json:** Contains settings for running the app locally (profiles, environment variables).

**In .NET Framework 4.5**

* **Route.config:** Defines URL routing rules for the application.
* **WebAPI.config:** Configures Web API-specific settings (routes, formatters, etc.).

1. Explain CORS enablement for Web API access for local application
   * What is CORS?, How to enable CORS thru Startup.cs, Install Cors nuget package to Web API application

CORS allows web applications running at one origin (domain) to access resources from a different origin. By default, browsers block such cross-origin requests for security. Enabling CORS in your API lets you specify which origins are allowed to interact with your API.