### Explain the concept of RESTful web service, Web API & Microservice

A RESTful web service follows the principles of REST (Representational State Transfer), which is an architectural style for building scalable and maintainable services using standard HTTP protocols. REST is stateless, meaning each request from a client contains all the information needed to process the request, and the server does not retain session state. Communication is done through messages, usually in JSON or XML format, though REST is not limited to XML. A **Web API** is a framework provided by .NET Core that allows developers to build HTTP services accessible over the web. A **microservice** is a small, independent service that performs a specific task and can be deployed and scaled independently. While a traditional Web Service might use SOAP and be tightly coupled, a Web API is typically RESTful and lightweight. Web APIs can serve data in multiple formats like JSON, XML, etc., offering greater flexibility than older web services which were limited mainly to XML.

### Explain what is HttpRequest & HttpResponse

An **HttpRequest** is the message sent by a client to a server, asking for a resource or initiating an operation. It includes elements such as the HTTP method (GET, POST, etc.), headers (like authorization and content type), query parameters, and optionally, a body (especially in POST and PUT requests). An **HttpResponse** is the message the server sends back to the client in response to the request. It contains status codes (like 200 OK or 404 Not Found), response headers, and a body which often contains the data or message. These two elements form the core communication model of web-based APIs and services.

### List the types of Action Verbs

Action verbs in Web API correspond to HTTP methods that define the operation to be performed. The most common action verbs are HttpGet, HttpPost, HttpPut, and HttpDelete. HttpGet is used to retrieve data from the server. HttpPost is used to create new data on the server. HttpPut is used to update existing data, and HttpDelete is used to remove data. In a Web API controller, these are declared using attributes like [HttpGet], [HttpPost], etc., on top of action methods to indicate the type of request each method handles.

### List the types of HttpStatusCodes used in WebAPI

Web APIs use HTTP status codes to communicate the result of a request. The most commonly used ones include 200 OK, which indicates a successful request; 400 BadRequest, which signifies invalid input or parameters; 401 Unauthorized, which means the client lacks proper authentication; and 500 InternalServerError, which signals that something went wrong on the server side. These status codes are returned using action result types like Ok(), BadRequest(), Unauthorized(), and StatusCode(500) in controller methods.

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

A simple Web API can be created using the ASP.NET Core Web API template. It involves defining a controller class that inherits from ControllerBase or ApiController. The controller includes methods decorated with HTTP verb attributes like [HttpGet] for read operations and [HttpPost] for write operations. Each action method represents an endpoint that the client can call. For example, a GetEmployees() method returns a list of employees, while a PostEmployee() method can add a new employee. The Web API structure typically includes a Controllers folder, within which controller classes are organized.

### Explain the types of Configuration files of WebAPI

In ASP.NET Core Web API, several configuration files are used. Program.cs (and previously Startup.cs) is the central file where middleware and services (like routing, authentication, and Swagger) are configured using dependency injection. appsettings.json is used to store configuration values such as connection strings and logging settings in a structured format. launchSettings.json, located in the Properties folder, configures how the application launches in different environments, including URLs and profiles for IIS Express or Kestrel. In older .NET Framework projects (like .NET 4.5), configuration was handled through Web.config and routing through Route.config. ASP.NET Core has modernized and simplified this through a more flexible and modular configuration system.