1) Why we use IActionResult not ActionResult support ur answer with scenario or problems

IActionResult is more flexible than ActionResult. It allows you to return different types of responses, not just views.

Key Differences:
ActionResult can only return specific result types (like ViewResult, JsonResult, etc.).
IActionResult can return any result type, making it more versatile.
Example Scenario
Imagine you are building an API that sometimes returns a view and sometimes JSON:

```
public IActionResult GetProduct(int id)
{
   var product = _db.Products.Find(id);

   if (product == null)
      return NotFound(); // Returns a 404 response

   if (Request.Headers["Accept"] == "application/json")
      return Json(product); // Returns JSON if requested

   return View(product); // Otherwise, return a view
}
```

If you used ActionResult, you would have to specify a fixed return type (ViewResult, JsonResult, etc.), limiting flexibility.

When to Use ActionResult
Use ActionResult if you only return one specific type, like a view:

```
public ViewResult ShowProduct()
{
   return View();
}
```

2) What the httpcontext request and response message consist of ?

HTTP Request Message: Contains method (GET, POST, etc.), headers, URL, query parameters, body (for POST/PUT).
HTTP Response Message: Contains status code (200, 404, etc.), headers, body (HTML, JSON, etc.).

3) What's the diff btw https and http

HTTP sends data in plain text, making it vulnerable to attacks.
HTTPS encrypts data using SSL/TLS, ensuring security and data integrity.

4) What's the segments and fragments in URL with real URL Example

Segments: Parts of the URL separated by slashes (/), defining the path.
Example: https://example.com/products/electronics/laptops

Segments: products, electronics, laptops
Fragment: The part after #, used for navigation within a page.

Example: https://example.com/docs#section2
Fragment: section2 (scrolls to that section on the page).

5) What's Builder and Dependency injection with a real life example clarify it

Builder Pattern creates complex objects step by step, like making a custom pizza by adding ingredients.
Ex:

```
public class PizzaBuilder
{
   private Pizza _pizza = new Pizza();

   public PizzaBuilder AddDough(string dough) { _pizza.Dough = dough; return this; }
   public PizzaBuilder AddSauce(string sauce) { _pizza.Sauce = sauce; return this; }
   public PizzaBuilder AddTopping(string topping) { _pizza.Topping = topping; return this; }
   public Pizza Build() => _pizza;
}
var pizza = new
PizzaBuilder().AddDough("Thin").AddSauce("Tomato").AddTopping("Cheese").Build();
```

Dependency Injection (DI) provides required dependencies automatically, like a coffee machine getting water and beans without manual setup.

```
public interface ICoffeeService { void Brew(); }
public class CoffeeService : ICoffeeService { public void Brew() =>
Console.WriteLine("Brewing Coffee..."); }

public class CoffeeMachine
{
   private readonly ICoffeeService _coffeeService;
   public CoffeeMachine(ICoffeeService coffeeService) { _coffeeService = coffeeService; }
   public void MakeCoffee() => _coffeeService.Brew();
}

services.AddScoped<ICoffeeService, CoffeeService>();
```

6) What's the difference btw Web Pages(Razor) and MVC and state two business cases

Web Pages (Razor) is simple, best for small apps with mixed UI and logic. MVC separates concerns, ideal for scalable apps.
Example: A personal blog fits Razor Pages, while an e-commerce site benefits from MVC's structure.

7) What's Content type in response message and where we use it and why

Content-Type in a response message tells the client what type of data is being sent (e.g., text/html, application/json).
Usage: APIs, web pages, and file downloads to ensure the correct processing of data.
Why: The browser or client knows how to handle the response properly (e.g., render HTML, parse JSON).

8) what's minification, web bundle, webPack and lazy loading of client side and what's its role in increasing performance through the network

Minification: Removes unnecessary characters (spaces, comments) from CSS/JS files, reducing size for faster loading.
Web Bundling: Combines multiple CSS/JS files into one to reduce HTTP requests, improving load speed.
Webpack: A module bundler that optimizes, bundles, and minifies assets like JavaScript, CSS, and images.
Lazy Loading: Loads only the needed content instead of everything at once, reducing initial load time.