5. WebApi\_Handson Objectives Answers

# 1. Explain CORS Enablement for Web API Access

CORS (Cross-Origin Resource Sharing) is a security feature implemented in browsers to restrict web pages from making requests to a different domain than the one that served the web page. It is required when a frontend application (e.g., React, Angular) tries to access a Web API hosted on a different origin (domain/port/protocol).

To enable CORS in ASP.NET Core Web API:

**Steps:**

1. Install the NuGet package:  
 Microsoft.AspNetCore.Cors

2. In Program.cs or Startup.cs, add the following:

In Program.cs (for .NET 6+):

builder.Services.AddCors(options =>  
{  
 options.AddPolicy("AllowAll",  
 builder => builder.AllowAnyOrigin()  
 .AllowAnyMethod()  
 .AllowAnyHeader());  
});  
  
app.UseCors("AllowAll");

# 2. Demonstrate Security in WebAPI

To secure Web API endpoints using JWT tokens and role-based access:

Key Concepts:

- Bearer and JWT Token authentication

- Claims embedded in JWT

- Role-based authorization using [Authorize(Roles = "...")]

Implementation in Program.cs:

string securityKey = "mysuperdupersecretkeyforjwtsigning1234";  
var symmetricSecurityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(securityKey));  
  
builder.Services.AddAuthentication(x =>  
{  
 x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;  
 x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;  
})  
.AddJwtBearer(JwtBearerDefaults.AuthenticationScheme, x =>  
{  
 x.TokenValidationParameters = new TokenValidationParameters  
 {  
 ValidateIssuer = true,  
 ValidateAudience = true,  
 ValidateLifetime = true,  
 ValidateIssuerSigningKey = true,  
 ValidIssuer = "mySystem",  
 ValidAudience = "myUsers",  
 IssuerSigningKey = symmetricSecurityKey  
 };  
});  
  
app.UseAuthentication();  
app.UseAuthorization();

AuthController Example (AllowAnonymous):

[ApiController]  
[Route("api/[controller]")]  
[AllowAnonymous]  
public class AuthController : ControllerBase  
{  
 [HttpGet]  
 public IActionResult GetToken()  
 {  
 string token = GenerateJSONWebToken(1, "Admin");  
 return Ok(new { Token = token });  
 }  
  
 private string GenerateJSONWebToken(int userId, string userRole)  
 {  
 var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretkeyforjwtsigning1234"));  
 var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);  
  
 var claims = new List<Claim>  
 {  
 new Claim(ClaimTypes.Role, userRole),  
 new Claim("UserId", userId.ToString())  
 };  
  
 var token = new JwtSecurityToken(  
 issuer: "mySystem",  
 audience: "myUsers",  
 claims: claims,  
 expires: DateTime.Now.AddMinutes(10),  
 signingCredentials: credentials  
 );  
  
 return new JwtSecurityTokenHandler().WriteToken(token);  
 }  
}