**Hands-On Exercises: Authentication and Authorization in ASP.NET Core Web API Microservices**

**Question 1: Implement JWT Authentication in ASP.NET Core Web API Scenario: You are building a microservice that requires secure login. You need to implement JWTbased authentication.**

**SOLUTION :**

Step 1: Install Required NuGet Packages

Install these packages in your project via NuGet Package Manager:

Install-Package Microsoft.AspNetCore.Authentication.JwtBearer

Install-Package Swashbuckle.AspNetCore

Step 2: Add Configuration in appsettings.json

"Jwt": {

"Key": "YourSecretKeyForJWTTokenDontShare",

"Issuer": "YourApp",

"Audience": "YourAppUsers",

"DurationInMinutes": 60

}

Step 3: Create a JWT Settings Model

Create a class JwtSettings.cs:

csharp

CopyEdit

public class JwtSettings

{

public string Key { get; set; }

public string Issuer { get; set; }

public string Audience { get; set; }

public int DurationInMinutes { get; set; }

}

Step 4: Configure JWT in Program.cs

var builder = WebApplication.CreateBuilder(args);

// Bind JWT settings

builder.Services.Configure<JwtSettings>(

builder.Configuration.GetSection("Jwt"));

var jwtSettings = builder.Configuration.GetSection("Jwt").Get<JwtSettings>();

builder.Services.AddAuthentication("Bearer")

.AddJwtBearer("Bearer", options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = jwtSettings.Issuer,

ValidAudience = jwtSettings.Audience,

IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtSettings.Key))

};

});

builder.Services.AddAuthorization();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

Description = "JWT Authorization header using the Bearer scheme. \r\n\r\n Example: \"Bearer abc123\"",

Name = "Authorization",

In = ParameterLocation.Header,

Type = SecuritySchemeType.ApiKey,

Scheme = "Bearer"

});

c.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

new string[] {}

}

});

});

var app = builder.Build();

app.UseSwagger();

app.UseSwaggerUI();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

Step 5: Create Login API (AuthController.cs)

[ApiController]

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

public class AuthController : ControllerBase

{

private readonly IConfiguration \_config;

public AuthController(IConfiguration config)

{

\_config = config;

}

[HttpPost("login")]

public IActionResult Login([FromBody] LoginModel user)

{

if (user.Username == "admin" && user.Password == "password")

{

var token = GenerateJwtToken();

return Ok(new { token });

}

return Unauthorized();

}

private string GenerateJwtToken()

{

var jwtSettings = \_config.GetSection("Jwt").Get<JwtSettings>();

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtSettings.Key));

var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);

var token = new JwtSecurityToken(

issuer: jwtSettings.Issuer,

audience: jwtSettings.Audience,

expires: DateTime.Now.AddMinutes(jwtSettings.DurationInMinutes),

signingCredentials: credentials

);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

public class LoginModel

{

public string Username { get; set; }

public string Password { get; set; }

}

Step 6: Protect Other Controllers with [Authorize]

[Authorize]

[ApiController]

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

public class ProtectedController : ControllerBase

{

[HttpGet]

public IActionResult GetSecretData()

{

return Ok("This is protected content.");

}

}

Step 7: Use Swagger to Authenticate

1. Run the project → Swagger UI opens.
2. Use /api/Auth/login POST with body:

{

"username": "admin",

"password": "password"

}

1. Copy the token from the response.
2. Click “Authorize” in the top-right.
3. Paste:

php-template

Bearer <your\_token\_here>

1. Now test [Authorize] APIs — they will work.

Output :

