**Secure Web API with CORS and JWT Authentication**

**Step 1: Enable CORS**

CORS (Cross-Origin Resource Sharing) allows your Web API to be accessed by a frontend running on a different domain or port.

Modify `Program.cs` to include the following:

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

**Step 2: Add JWT Authentication**

Install the JWT bearer package:

dotnet add package Microsoft.AspNetCore.Authentication.JwtBearer

In `Program.cs`, add:

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)  
 .AddJwtBearer(options =>  
 {  
 options.TokenValidationParameters = new TokenValidationParameters  
 {  
 ValidateIssuer = true,  
 ValidateAudience = true,  
 ValidateLifetime = true,  
 ValidateIssuerSigningKey = true,  
 ValidIssuer = "yourdomain.com",  
 ValidAudience = "yourdomain.com",  
 IssuerSigningKey = new SymmetricSecurityKey(  
 Encoding.UTF8.GetBytes("YourSuperSecretKey12345"))  
 };  
 });  
  
app.UseAuthentication();  
app.UseAuthorization();

**Step 3: Create Token Generation Endpoint**

Create a new controller `AuthController.cs` in the `Controllers` folder:

[ApiController]  
[Route("api/[controller]")]  
public class AuthController : ControllerBase  
{  
 [HttpPost("login")]  
 public IActionResult Login([FromBody] LoginModel model)  
 {  
 if (model.Username == "admin" && model.Password == "password")  
 {  
 var tokenHandler = new JwtSecurityTokenHandler();  
 var key = Encoding.UTF8.GetBytes("YourSuperSecretKey12345");  
 var tokenDescriptor = new SecurityTokenDescriptor  
 {  
 Subject = new ClaimsIdentity(new[]  
 {  
 new Claim(ClaimTypes.Name, model.Username),  
 new Claim(ClaimTypes.Role, "Admin")  
 }),  
 Expires = DateTime.UtcNow.AddHours(1),  
 Issuer = "yourdomain.com",  
 Audience = "yourdomain.com",  
 SigningCredentials = new SigningCredentials(  
 new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature)  
 };  
  
 var token = tokenHandler.CreateToken(tokenDescriptor);  
 var jwt = tokenHandler.WriteToken(token);  
 return Ok(new { token = jwt });  
 }  
  
 return Unauthorized("Invalid credentials");  
 }  
}  
  
public class LoginModel  
{  
 public string? Username { get; set; }  
 public string? Password { get; set; }  
}

**Step 4: Run and Test Using Swagger**

1. Run the application using:  
 dotnet run  
  
2. Open Swagger UI in your browser:  
 http://localhost:5208/swagger  
  
3. Call the `POST /api/Auth/login` endpoint with this JSON body:  
{  
 "username": "admin",  
 "password": "password"  
}  
  
4. A successful login will return a JWT token.

Step 5: Secure Any Endpoint with [Authorize]

