## What is CORS?

CORS (Cross-Origin Resource Sharing) is a security feature implemented by browsers to restrict cross-origin HTTP requests initiated from scripts running in the browser. To allow a web app hosted at one origin (domain) to access resources from a different origin, you must enable CORS on the server.

1. Configure and Add JWT Authentication in program.cs
   1. program.cs

| using Microsoft.OpenApi.Models;  using Microsoft.AspNetCore.Authentication.JwtBearer;  using Microsoft.IdentityModel.Tokens;  using System.Text;  using WebApplication1.Filters;  var builder = WebApplication.CreateBuilder(args);  builder.WebHost.UseUrls("http://localhost:5050");  builder.Services.AddControllers(opts =>  {  opts.Filters.Add<CustomExceptionFilter>();  });  builder.Services.AddScoped<CustomAuthFilter>();  builder.Services.AddCors(options =>  {  options.AddPolicy("AllowAll", p =>  p.AllowAnyOrigin()  .AllowAnyHeader()  .AllowAnyMethod());  });  const string securityKey = "mysuperdupersecretthatissufficientlylong";  var symmetricKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(securityKey));  builder.Services.AddAuthentication(options =>  {  options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;  options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;  })  .AddJwtBearer(options =>  {  options.TokenValidationParameters = new TokenValidationParameters  {  ValidateIssuer = true,  ValidateAudience = true,  ValidateLifetime = true,  ValidateIssuerSigningKey = true,  ValidIssuer = "mySystem",  ValidAudience = "myUsers",  IssuerSigningKey = symmetricKey  };  });  builder.Services.AddEndpointsApiExplorer();  builder.Services.AddSwaggerGen(c =>  {  c.SwaggerDoc("v1", new OpenApiInfo  {  Title = "Swagger Demo",  Version = "v1",  Description = "JWT + CORS demo",  Contact = new OpenApiContact { Name = "John Doe", Email = "john@xyzmail.com" }  });  c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme  {  Name = "Authorization",  Type = SecuritySchemeType.ApiKey,  Scheme = "Bearer",  BearerFormat = "JWT",  In = ParameterLocation.Header,  Description = "Enter \*\*Bearer &lt;token&gt;\*\*"  });  c.AddSecurityRequirement(new OpenApiSecurityRequirement  {  {  new OpenApiSecurityScheme  {  Reference = new OpenApiReference  {  Type = ReferenceType.SecurityScheme,  Id = "Bearer"  }  },  Array.Empty<string>()  }  });  });  var app = builder.Build();  app.UseSwagger();  app.UseSwaggerUI(c =>  {  c.SwaggerEndpoint("/swagger/v1/swagger.json", "Swagger Demo");  });  app.UseCors("AllowAll");  app.UseAuthentication();  app.UseAuthorization();  app.MapControllers();  app.Run(); |
| --- |

1. Create new controller
   1. AuthController.cs

| using Microsoft.AspNetCore.Authorization;  using Microsoft.AspNetCore.Mvc;  using Microsoft.IdentityModel.Tokens;  using System.IdentityModel.Tokens.Jwt;  using System.Security.Claims;  using System.Text;  namespace WebApplication1.Controllers  {  [ApiController]  [Route("api/[controller]")]  [AllowAnonymous]  public class AuthController : ControllerBase  {  private string GenerateJSONWebToken(int userId, string userRole)  {  var securityKey = new SymmetricSecurityKey(  Encoding.UTF8.GetBytes("mysuperdupersecretthatissufficientlylong")); // updated here  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);  }  [HttpGet]  public IActionResult GetToken([FromQuery] int userId = 1, [FromQuery] string role = "Admin")  {  var token = GenerateJSONWebToken(userId, role);  return Ok(token);  }  }  } |
| --- |

