**Name: Deepanshu Rathore**

**Superset ID: 6358199**  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Week 4  
**5. WebApi\_Handson**Cross Origin Resource Sharing or CORS is a browser security feature that restricts web applications running in one origin from requesting resources from a different origin unless the response from the other origin includes the proper headers. When a browser makes a request from, for example, https://localhost:4200 to https://localhost:5001, the API must explicitly allow that origin. Without CORS enabled the browser will block the request even if the server responds. To enable CORS in a .NET Core Web API you first install the Microsoft.AspNetCore.Cors NuGet package into your project. In the Startup class you then register the CORS services in the ConfigureServices method. You call services.AddCors, define a policy name along with the allowed origins headers and methods, and then in the Configure method you call app.UseCors with that policy name before you call app.UseRouting or app.UseEndpoints so that incoming requests are checked for the CORS policy and the appropriate headers are added to responses.

JSON Web Token authentication is a method of transmitting claims securely between two parties. A token is generated by the server when the user supplies valid credentials. The token contains claims such as roles and user identifiers and is signed with a secret key known only to the server. Subsequent API calls include this token in the Authorization header prefixed by Bearer. In Startup.cs you call services.AddAuthentication and set the default schemes to JwtBearerDefaults.AuthenticationScheme. You then call AddJwtBearer, configuring TokenValidationParameters to validate issuer audience lifetime and signing key. In the HTTP pipeline you enable middleware by calling app.UseAuthentication before app.UseAuthorization. To issue tokens you create an AuthController marked with [AllowAnonymous]. Inside you build a JwtSecurityToken by supplying the issuer audience claims expiry and signing credentials. You then write the token out as a string. Downstream controllers can then be protected with [Authorize] marking them to require a valid bearer token. If you need role based restriction you include roles in the claim and use [Authorize(Roles="Admin,User")] on the controller or action. When a token expires the middleware rejects the request with 401 Unauthorized. Claims can be read inside controllers via User.Claims.

EmployeeController.cs:

using System.Text;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.IdentityModel.Tokens;

using Microsoft.AspNetCore.Authentication.JwtBearer;

namespace YourApiNamespace

{

public class Startup

{

private const string CorsPolicyName = "AllowAllOrigins";

private const string SecurityKey = "mysuperdupersecret";

private const string Issuer = "mySystem";

private const string Audience = "myUsers";

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

public void ConfigureServices(IServiceCollection services)

{

services.AddCors(options =>

{

options.AddPolicy(name: CorsPolicyName,

builder =>

{

builder

.AllowAnyOrigin()

.AllowAnyMethod()

.AllowAnyHeader();

});

});

var key = Encoding.UTF8.GetBytes(SecurityKey);

var signingKey = new SymmetricSecurityKey(key);

services

.AddAuthentication(options =>

{

options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

options.DefaultSignInScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(JwtBearerDefaults.AuthenticationScheme, options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = Issuer,

ValidAudience = Audience,

IssuerSigningKey = signingKey

};

});

services.AddAuthorization();

services.AddControllers();

}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

app.UseRouting();

app.UseCors(CorsPolicyName);

app.UseAuthentication();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

}

}

}

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.