**SOLUTION 5:**

**a)JsonWebToken**

**b) Use the JWT generated thru the AuthController to be used in POSTMAN request**

**c) Check for JWT expiration**

**d)** **Add the roles to be authorized in the Authorize attribute**

**EmployeeController.cs:**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

[Authorize(Roles = "Admin,POC")]

[ApiController]

[Route("[controller]")]

public class EmployeeController : ControllerBase

{

[HttpGet]

public IActionResult Get()

{

return Ok("Employee data accessed.");

}

}

**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;

[AllowAnonymous]

[ApiController]

[Route("[controller]")]

public class AuthController : ControllerBase

{

[HttpGet("token")]

public IActionResult GetToken()

{

var token = GenerateJSONWebToken(1, "Admin");

return Ok(new { token });

}

private string GenerateJSONWebToken(int userId, string userRole)

{

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecret"));

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);

}

}

**Program.cs:**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

// Add CORS policy (optional)

builder.Services.AddCors(options =>

{

options.AddPolicy("AllowAllOrigins", policy =>

{

policy.AllowAnyOrigin().AllowAnyHeader().AllowAnyMethod();

});

});

// JWT secret key (must be the same in AuthController)

string securityKey = "mysuperdupersecretkey123456789012";

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

// Add Authentication

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 = symmetricSecurityKey

};

});

builder.Services.AddControllers();

var app = builder.Build();

app.UseRouting();

app.UseCors("AllowAllOrigins");

app.UseAuthentication(); // Add Authentication Middleware

app.UseAuthorization(); // Add Authorization Middleware

app.MapControllers();

app.Run();

**Output:**



