**Question 3: Add Role-Based Authorization Scenario:**

**You want to allow only users with the "Admin" role to access certain endpoints.**

Steps: 1. Add roles to JWT claims.

2. Use `[Authorize(Roles = "Admin")]`.

AuthController.cs:

using JwtAuthDemo.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace JwtAuthDemo.Controllers

{

    [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 model)

        {

            if (!ModelState.IsValid)

            {

                return BadRequest(ModelState);

            }

            if (string.IsNullOrWhiteSpace(model.Username) || string.IsNullOrWhiteSpace(model.Password))

            {

                return BadRequest("Username and password are required.");

            }

            if (IsValidUser(model))

            {

                var token = GenerateJwtToken(model.Username!);

                return Ok(new { Token = token });

            }

            return Unauthorized();

        }

        private bool IsValidUser(LoginModel model)

        {

            // Demo: Only one valid hardcoded user with Admin role

            return model.Username == "mani" && model.Password == "mani123";

        }

        private string GenerateJwtToken(string username)

        {

            var claims = new[]

            {

                new Claim(ClaimTypes.Name, username),

                new Claim(ClaimTypes.Role, "Admin") // ✅ Add Admin role here

            };

            var keyString = \_config.GetValue<string>("Jwt:Key")

                ?? throw new InvalidOperationException("JWT Key is missing in configuration.");

            if (Encoding.UTF8.GetBytes(keyString).Length < 32)

                throw new InvalidOperationException("JWT key must be at least 256 bits (32 bytes) long.");

            var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(keyString));

            var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

            var token = new JwtSecurityToken(

                issuer: \_config["Jwt:Issuer"],

                audience: \_config["Jwt:Audience"],

                claims: claims,

                expires: DateTime.Now.AddMinutes(Convert.ToDouble(\_config["Jwt:DurationInMinutes"] ?? "60")),

                signingCredentials: creds

            );

            return new JwtSecurityTokenHandler().WriteToken(token);

        }

    }

}

Output:



