}

Models:

1.)Login Model.cs

```
using System.ComponentModel.DataAnnotations;


namespace SecureJwtApp.Models
{
    public class LoginModel
    {
        [Required, EmailAddress]
        public string Email { get; set; }


        [Required]
        public string Password { get; set; }
    }
}
```

2.)User Model.cs

```
namespace SecureJwtApp.Models
{
    public class UserModel
    {
        public string Email { get; set; }
        public string Password { get; set; }
        public string Role { get; set; }  // "Admin" or "User"
    }
}
```

-JWT Service.cs

```
using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

using SecureJwtApp.Models;


namespace SecureJwtApp.Services

{

    public class JwtService

    {

        private readonly IConfiguration _config;


        public JwtService(IConfiguration config)

        {

            _config = config;

        }


        public string GenerateToken(UserModel user)

        {

            var claims = new[]

            {

                new Claim(JwtRegisteredClaimNames.Sub, user.Email),

                new Claim(ClaimTypes.Role, user.Role),

                new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString())
            };

            var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_config["Jwt:Key"]));
            var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

            var token = new JwtSecurityToken(
                issuer: _config["Jwt:Issuer"],
                audience: _config["Jwt:Audience"],
                claims: claims,
                expires: DateTime.UtcNow.AddMinutes(15),
                signingCredentials: creds
            );

            return new JwtSecurityTokenHandler().WriteToken(token);
        }
    }
}
```

-Controllers:

1.)AuthController.cs

```
using Microsoft.AspNetCore.Mvc;
using SecureJwtApp.Models;
using SecureJwtApp.Services;

namespace SecureJwtApp.Controllers
{
```

```
[ApiController]

[Route("api/[controller]")]

public class AuthController : ControllerBase

{

    private readonly JwtService _jwtService;



    // Dummy users (replace with DB in real-world apps)

    private readonly List<UserModel> _users = new()

    {

        new UserModel { Email = "admin@test.com", Password = "Admin@123", Role = "Admin" },

        new UserModel { Email = "user@test.com", Password = "User@123", Role = "User" }

    };



    public AuthController(JwtService jwtService)

    {

        _jwtService = jwtService;

    }



    [HttpPost("login")]

    public IActionResult Login([FromBody] LoginModel model)

    {

        var user = _users.FirstOrDefault(x => x.Email == model.Email && x.Password == model.Password);

        if (user == null) return Unauthorized("Invalid credentials");



        var token = _jwtService.GenerateToken(user);
```

```
            return Ok(new { token });
        }
    }
}
```

2.)Data Controller.cs

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace SecureJwtApp.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class DataController : ControllerBase
    {
        [Authorize(Roles = "Admin")]
        [HttpGet("admin")]
        public IActionResult GetAdminData()
        {
            return Ok("This is sensitive data visible only to Admins!");
        }

        [Authorize(Roles = "User")]
        [HttpGet("user")]
        public IActionResult GetUserData()
        {
            return Ok("This is user data accessible to logged-in Users.");
        }

        [Authorize]
        [HttpGet("common")]
        public IActionResult GetCommonData()
        {
            return Ok("This is common data for all authenticated users.");
        }
    }
}
```

-Program.cs

```
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.IdentityModel.Tokens;
using SecureJwtApp.Services;
using System.Text;

var builder = WebApplication.CreateBuilder(args);

// Add JwtService
builder.Services.AddSingleton<JwtService>();

// Configure JWT Authentication
builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
```

```
    .AddJwtBearer(options =>
    {
        options.TokenValidationParameters = new TokenValidationParameters
        {
            ValidateIssuer = true,
            ValidateAudience = true,
            ValidateLifetime = true,
            ValidateIssuerSigningKey = true,
            ValidIssuer = builder.Configuration["Jwt:Issuer"],
            ValidAudience = builder.Configuration["Jwt:Audience"],
            IssuerSigningKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Configuration["Jwt:Key"]))
        };
    });

builder.Services.AddAuthorization();
builder.Services.AddControllers();

var app = builder.Build();

// Enforce HTTPS
app.UseHttpsRedirection();

app.UseAuthentication();
app.UseAuthorization();
app.MapControllers();

app.Run();
```

appsetting.json:

```
{

"ConnectionStrings": {
    "DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database=aspnet-MvcExample-
014b4166-fb86-417c-b509-
4313b550e82d;Trusted_Connection=True;MultipleActiveResultSets=true",


  "Jwt": {

    "Key": "ThisIsASecretKeyForJwtToken123!",

    "Issuer": "SecureJwtApp",

    "Audience": "SecureJwtAppUsers"

  },

  "Logging": {

    "LogLevel": {

      "Default": "Information",

      "Microsoft.AspNetCore": "Warning"

    }

  },

  "AllowedHosts": "*"

}
```