Step-By-Step guide to implement JWT Authentication in ASP.NET Core Web API

1. Create new ASP.NET Core Web API Project using Visual Studio 2022

2. Add NuGet Packages

Install the following packages Package Manager Console:

```
Install-Package Microsoft.AspNetCore.Authentication.JwtBearer   -Version 8.0.6
Install-Package System.IdentityModel.Tokens.Jwt     -Version 8.6.0
```

3. Update appsettings.json

In Solution Explorer, double-click `appsettings.json`, add "Jwt" section as follows:

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "Jwt": {
    "Key": "Your_Secret_Key_Here_ASKJFALKDJF57454897454",
    "Issuer": "YourIssuer",
    "Audience": "YourAudience"
  }
}
```

4. Add Models

1. Right-click project > Add > New Folder > Name it "Models"
2. Right-click Models folder > Add > Class

Product.cs:

```
namespace WebApplication22.Models
{
    public class Product
    {
        public int Id { get; set; }
```

```
        public string Name { get; set; }
        public decimal Price { get; set; }
    }
}
```

UserModel.cs:

```
namespace WebApplication22.Models
{
     public class UserModel
 {
     public string UserName { get; set; }
     public string Password { get; set; }
     public string? Role { get; set; }
 }
}
```

5. Add Controllers

1. Right-click Controllers folder > Add > Controller
2. Select "API Controller - Empty"

AuthController.cs:

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;
using WebApplication4.Models;

namespace WebApplication4.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    [AllowAnonymous]
    public class AuthenticateController : ControllerBase
    {
        public List<UserModel> usersList = null;

        public AuthenticateController()
        {
            usersList = new List<UserModel>()
            {
                new UserModel() { UserName = "Admin", Password = "Admin123",
Role="Admin" },
                new UserModel() { UserName = "Scott", Password = "Scott123",
```

```
Role="Default"}
            };
        }


        [HttpPost]
        public IActionResult Login(UserModel requestUser)
        {
            // 1.  Verify the user credentails
            UserModel userObj = usersList.Where(x => x.UserName ==
requestUser.UserName && x.Password == requestUser.Password).FirstOrDefault();


            if (userObj != null)
            {
                // 2.  Generate JWT Token
                string tokenStr = GenerateJSONWebToken(userObj);
                return Ok(new { token = tokenStr });
            }
            else
            {
                return BadRequest("Invalid user id or password");
            }
        }


        private string GenerateJSONWebToken(UserModel userObj)
        {
            SymmetricSecurityKey securityKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes("Your_Secret_Key_Here_ASKJFALKDJF57454
897454"));

            SigningCredentials credentials = new SigningCredentials(securityKey,
            SecurityAlgorithms.HmacSha256);


            List<Claim> authClaims = new List<Claim>
            {
                    new
Claim(ClaimTypes.NameIdentifier,Convert.ToString(userObj.UserName)),
                    new Claim(ClaimTypes.Name, userObj.UserName),
                    new Claim(JwtRegisteredClaimNames.Jti,
Guid.NewGuid().ToString()), // (JWT ID) Claim
                    new Claim(ClaimTypes.Role, userObj.Role)
            };

            JwtSecurityToken token = new JwtSecurityToken(
                        issuer: "YourIssuer",
                        audience: "YourAudience",
                        claims: authClaims,
                        expires: DateTime.Now.AddMinutes(2),
                        signingCredentials: credentials);

            string tokenString = new JwtSecurityTokenHandler().WriteToken(token);
```

```
            return tokenString;
        }

    }
}
```

ProductController.cs:

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using WebApplication22.Models;

namespace WebApplication22.Controllers
{

    [Route("api/[controller]")]
    [ApiController]
    [Authorize]
    // [Authorize(Roles = "Admin")]
    public class ProductsController : ControllerBase
    {
        private readonly ProductDbContext _context;

        public ProductsController(ProductDbContext context)
        {
            _context = context;
        }

        // GET: api/products
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Product>>> GetProducts()
        {
            return await _context.Products.ToListAsync();
        }

        // GET: api/products/5
        [HttpGet("{id}")]
        public async Task<ActionResult<Product>> GetProduct(int id)
        {
            var product = await _context.Products.FindAsync(id);
            if (product == null) return NotFound();
            return product;
        }

        // POST: api/products
        [HttpPost]
        public async Task<ActionResult<Product>> PostProduct(Product product)
        {
            _context.Products.Add(product);
            await _context.SaveChangesAsync();
```

```
            return CreatedAtAction(nameof(GetProduct), new { id = product.Id },
product);
        }

        // PUT: api/products/5
        [HttpPut("{id}")]
        public async Task<IActionResult> PutProduct(int id, Product product)
        {
            if (id != product.Id) return BadRequest();

            _context.Entry(product).State = EntityState.Modified;
            await _context.SaveChangesAsync();
            return NoContent();
        }

        // DELETE: api/products/5
        [HttpDelete("{id}")]
        public async Task<IActionResult> DeleteProduct(int id)
        {
            var product = await _context.Products.FindAsync(id);
            if (product == null) return NotFound();

            _context.Products.Remove(product);
            await _context.SaveChangesAsync();
            return NoContent();
        }
    }
}
```

6. Update Program.cs

```
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.IdentityModel.Tokens;
using System.Text;
using WebApplication4.CustomMiddlewares;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();
builder.Services.AddSwaggerGen();

// 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"]))
        };
});




var app = builder.Build();

if (app.Environment.IsDevelopment())
{
    app.UseDeveloperExceptionPage(); // Enables detailed error page
}

app.UseSwagger();
app.UseSwaggerUI();

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


app.MapControllers();


app.Run();
```

7. Run and Test