**Week 5**

**Implement JWT Authentication in ASP.NET Core Web API**

Model

namespace JwtAuthApi.Models

{

public class LoginModel

{

public string Username { get; set; }

public string Password { get; set; }

}

}

AuthController

using JwtAuthApi.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

[ApiController]

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

public class AuthController : ControllerBase

{

[HttpPost("login")]

public IActionResult Login([FromBody] LoginModel model)

{

if (IsValidUser(model))

{

var token = GenerateJwtToken(model.Username);

return Ok(new { Token = token });

}

return Unauthorized();

}

private bool IsValidUser(LoginModel model)

{

// Replace with real validation (e.g., database check)

return model.Username == "DipanjanSaha" && model.Password == "qwerty1234";

}

private string GenerateJwtToken(string username)

{

var claims = new[]

{

new Claim(ClaimTypes.Name, username)

};

var key = new SymmetricSecurityKey(

Encoding.UTF8.GetBytes("ThisIsASecretKeyForJwtTokenWithProperLength"));

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

var token = new JwtSecurityToken(

issuer: "MyAuthServer",

audience: "MyApiUsers",

claims: claims,

expires: DateTime.Now.AddMinutes(60),

signingCredentials: creds);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

SecretController

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

[ApiController]

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

public class SecretController : ControllerBase

{

[HttpGet]

[Authorize]

public IActionResult GetSecret()

{

return Ok("This is a protected secret message.");

}

}

Program.cs

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

// Add services

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

// Add Swagger with JWT Authorization support

builder.Services.AddSwaggerGen(options =>

{

options.SwaggerDoc("v1", new OpenApiInfo

{

Title = "JWT Auth API",

Version = "v1"

});

// 🔐 Add JWT Bearer to Swagger

options.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

Name = "Authorization",

Type = SecuritySchemeType.ApiKey,

Scheme = "Bearer",

BearerFormat = "JWT",

In = ParameterLocation.Header,

Description = "Enter JWT token like this: Bearer <your-token>"

});

options.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Id = "Bearer",

Type = ReferenceType.SecurityScheme

}

},

Array.Empty<string>()

}

});

});

// JWT Authentication Setup

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddJwtBearer(options =>

{

var jwtSettings = builder.Configuration.GetSection("Jwt");

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = jwtSettings["Issuer"],

ValidAudience = jwtSettings["Audience"],

IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtSettings["Key"]))

};

});

builder.Services.AddAuthorization();

var app = builder.Build();

// 🔧 Middleware

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

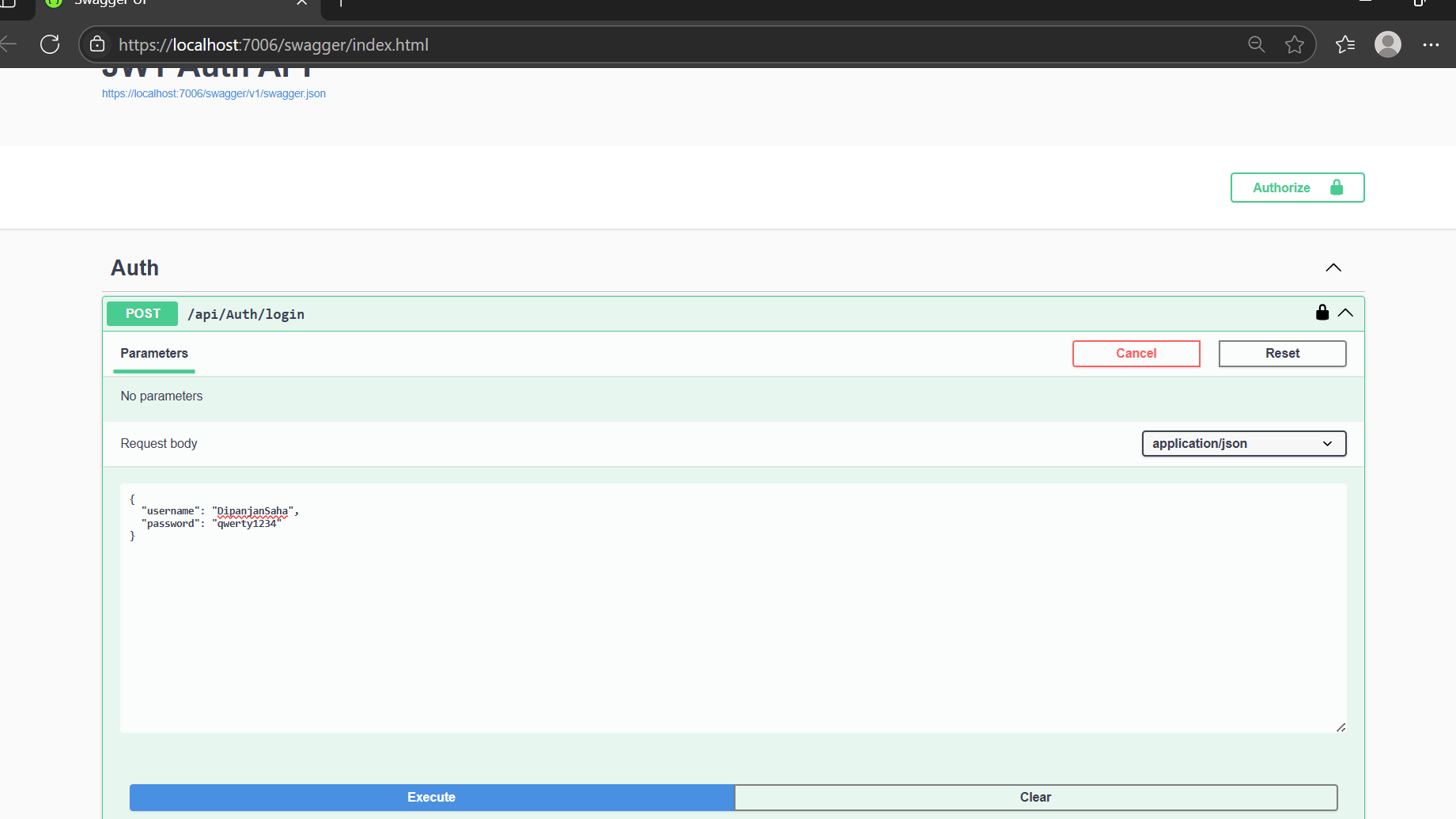
}

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();



Jwt token generation for authorization

