// new ASP.NET Core Web API project

dotnet new webapi -n UserManagementAPI

// project directory

cd UserManagementAPI

// NuGet packages for JWT authentication:

dotnet add package Microsoft.AspNetCore.Authentication.JwtBearer

dotnet add package System.IdentityModel.Tokens.Jwt

// Models/User.cs

namespace UserManagementAPI.Models

{

public class User

{

public int Id { get; set; }

public string Name { get; set; }

public string Email { get; set; }

}

}

// Services/JwtService.cs

using Microsoft.IdentityModel.Tokens;

using System;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace UserManagementAPI.Services

{

public class JwtService

{

private readonly string \_secret;

private readonly string \_issuer;

private readonly string \_audience;

public JwtService(string secret, string issuer, string audience)

{

\_secret = secret;

\_issuer = issuer;

\_audience = audience;

}

public string GenerateToken(string username)

{

var claims = new[]

{

new Claim(JwtRegisteredClaimNames.Sub, username),

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

};

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

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

var token = new JwtSecurityToken(

issuer: \_issuer,

audience: \_audience,

claims: claims,

expires: DateTime.Now.AddMinutes(30),

signingCredentials: creds);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

}

// Startup.cs

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

using UserManagementAPI.Services;

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

public void ConfigureServices(IServiceCollection services)

{

services.AddControllers();

var secret = Configuration["Jwt:Secret"];

var issuer = Configuration["Jwt:Issuer"];

var audience = Configuration["Jwt:Audience"];

services.AddSingleton(new JwtService(secret, issuer, audience));

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 = issuer,

ValidAudience = audience,

IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(secret))

};

});

}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseHttpsRedirection();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

}

}

// Controllers/UserController.cs

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

using System.Linq;

using UserManagementAPI.Models;

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

[ApiController]

[Authorize]

public class UserController : ControllerBase

{

private static List<User> users = new List<User>();

[HttpGet]

public ActionResult<IEnumerable<User>> GetUsers()

{

return users;

}

[HttpGet("{id}")]

public ActionResult<User> GetUser(int id)

{

var user = users.FirstOrDefault(u => u.Id == id);

if (user == null)

{

return NotFound();

}

return user;

}

[HttpPost]

public ActionResult<User> CreateUser(User user)

{

users.Add(user);

return CreatedAtAction(nameof(GetUser), new { id = user.Id }, user);

}

[HttpPut("{id}")]

public IActionResult UpdateUser(int id, User updatedUser)

{

var user = users.FirstOrDefault(u => u.Id == id);

if (user == null)

{

return NotFound();

}

user.Name = updatedUser.Name;

user.Email = updatedUser.Email;

return NoContent();

}

[HttpDelete("{id}")]

public IActionResult DeleteUser(int id)

{

var user = users.FirstOrDefault(u => u.Id == id);

if (user == null)

{

return NotFound();

}

users.Remove(user);

return NoContent();

}

}

// Controllers/AuthController.cs

using Microsoft.AspNetCore.Mvc;

using UserManagementAPI.Services;

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

[ApiController]

public class AuthController : ControllerBase

{

private readonly JwtService \_jwtService;

public AuthController(JwtService jwtService)

{

\_jwtService = jwtService;

}

[HttpPost("login")]

public IActionResult Login([FromBody] LoginRequest loginRequest)

{

if (loginRequest.Username == "admin" && loginRequest.Password == "password")

{

var token = \_jwtService.GenerateToken(loginRequest.Username);

return Ok(new { Token = token });

}

return Unauthorized();

}

}

public class LoginRequest

{

public string Username { get; set; }

public string Password { get; set; }

}

{

"Jwt": {

"Secret": "YourSecretKeyHere",

"Issuer": "YourIssuer",

"Audience": "YourAudience"

}

}

dotnet run