**Identity Framework to secure your end points**

Below are the steps to setup the Identity framework for any given project with basic assumptions.

**Step: Install Identity Framework**

Install below Nuget Packages to the project

Microsoft.AspNetCore.Identity.EntityFrameworkCore

Microsoft.AspNetCore.Authentication.JwtBearer

**Step: Configure Entity Framework to add required tables to db**

1. public class EMSDbContext : IdentityDbContext<ApplicationUser>
2. **Create ApplicationUser class in Model Folder**

public class ApplicationUser : IdentityUser

{

public string FirstName { get; set; }

public string LastName { get; set; }

public int Age { get; set; }

}

1. **Configure Porgram.cs for EF Configuration for Application User**

builder.services.AddDbContext<EMPDbContext>();

builder.services.AddIdentity<ApplicationUser, IdentityRole>()

.AddEntityFrameworkStores<EMPDbContext>();

1. **Configure identiy Password strength in program.cs**

builder.services.Configure<IdentityOptions>(options =>

{

options.Password.RequiredLength = 2;

options.Password.RequireDigit = false;

options.Password.RequireLowercase = false;

options.Password.RequireNonAlphanumeric = false;

options.Password.RequireUppercase = false;

}

);

1. **Create Migrations and update the database using EF PMConsole**

PMC> Add-migration identityadded

PMC> Update-database

**Step: Add AccountController for signup and signin activity**

1. **Add token configuration values in appsettings.json**

"Tokens": {

"Key": "Provideasecuritykeyforcreatingtoken",

"Issuer": "http://localhost:55940",

"Audience": "user"

}

1. **Add DTO Classes for SignUP and Login endpoints**
2. **SignUpDTO.cs**

public class SignUpDTO

{

public string FirstName { get; set; }

public string LastName { get; set; }

public string UserName { get; set; }

public string password { get; set; }

}

1. **LoginDTO.cs**

public class LoginDTO

{

public string Password { get; set; }

public string UserName { get; set; }

}

1. **Create Account controller, this controller will have user registration and also when user login jwt token is generated.**

public class AccountController : ControllerBase

{

public UserManager<ApplicationUser> \_userManager { get; }

public SignInManager<ApplicationUser> \_signInManager { get; }

public IConfiguration \_configuration { get; }

// signup the user

// database access for saving and retriving the data

// user manager - provide you a way to save the information

public AccountController(UserManager<ApplicationUser> userManager,

SignInManager<ApplicationUser> signInManager,

IConfiguration configuration)

{

\_userManager = userManager;

\_signInManager = signInManager;

\_configuration = configuration;

}

[HttpPost("signup")]

public async Task<IdentityResult> SignUp([FromBody] SignUpDTO signUp)

{

ApplicationUser appUser = new ApplicationUser()

{

FirstName = signUp.FirstName,

LastName = signUp.LastName,

UserName = signUp.UserName

};

return await \_userManager.CreateAsync(appUser, signUp.password);

}

// endpoint to generate the token

[HttpPost("login")]

public async Task< IActionResult> Login([FromBody] LoginDTO signUp)

{

// validation his credentials

var signInResult = await \_signInManager.PasswordSignInAsync(signUp.UserName, signUp.Password, false, false);

if (signInResult.Succeeded)

{

var user = await \_userManager.FindByNameAsync(signUp.UserName);

var issuer = \_configuration["Tokens:Issuer"];

var audience = \_configuration["Tokens:Audience"];

var key = \_configuration["Tokens:Key"];

// jwt token

var claims = new[] {

new Claim("FirstName", value: user.FirstName),

//new Claim(type:JwtRegisteredClaimNames.Email, value: user.Email),

new Claim(type:JwtRegisteredClaimNames.Jti, value: user.Id)

};

var keyBytes = Encoding.UTF8.GetBytes(key); // private key to create a token

var theKey = new SymmetricSecurityKey(keyBytes);

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

var token = new JwtSecurityToken(issuer, audience, claims, expires: DateTime.Now.AddMinutes(30), signingCredentials: creds);

return Ok(new { token = new JwtSecurityTokenHandler().WriteToken(token) });

}

return BadRequest("Invalid Credentials");

}}

**Step: Add middlewares for Validating token sent as part of headers [Bearer Token]**

1. **Secure the endpoints by adding annotations:**
   1. **To secure the full controller add authorize attribute for Class**

[Authorize]

* 1. **To secure the action add authorize attribute for methods/actions/endpoints**

[Authorize]

* 1. **To allow any methods/actions/endpoints without security add AllowAnonymous attribute**

[AllowAnonymous]

1. **configure token validation logic for identity service to validating the token whenever the token is sent to application via headers of each endpoint request.**

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

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

var key = Configuration["Tokens:Key"];

// cookie verification default

// jwt token

// facebook token -

builder.services.AddAuthentication().AddJwtBearer(options =>

{

options.RequireHttpsMetadata = false;

options.SaveToken = true;

options.TokenValidationParameters = new TokenValidationParameters

{

ValidIssuer = issuer,

ValidAudience = audience,

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

};

});

1. **Add authentication and authorization middlewares for application in program.cs**

app.Authentication();

app.Authorization();

**Step: Configure Swagger to accept the token and pass for each request.**

1. **Update the swagger service configuration in program.cs**

builder.services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo { Title = "EMSAPI", Version = "v1" });

c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

In = ParameterLocation.Header,

Description = "Please enter token",

Name = "Authorization",

Type = SecuritySchemeType.Http,

BearerFormat = "JWT",

Scheme = "bearer"

});

c.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type=ReferenceType.SecurityScheme,

Id="Bearer"

}

},

new string[]{}

}

});

});