**Web Api**

Startup.cs file

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using DocumentFormat.OpenXml.EMMA;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Logging;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using WebApplication16.Models;

using WebApplication16.Services;

namespace WebApplication16

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("ThisismySecretKey"));

services.AddAuthentication(options =>

{

options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

}).AddJwtBearer(o => o.TokenValidationParameters = new Microsoft.IdentityModel.Tokens.TokenValidationParameters()

{

ValidateIssuerSigningKey = true,

IssuerSigningKey = key,

ValidateIssuer = true,

ValidIssuer = "UserWebApi",

ValidateAudience = true,

ValidAudience = "CustomerWebApi"

});

services.AddControllers();

services.AddSwaggerGen();

services.AddDbContext<StudentDbContext>(op => op.UseSqlServer(Configuration["ConnectionStrings:StudentDbContext"]));

services.AddScoped<ITokenGeneratorService, TokenGeneratorService>();

//services.AddSwaggerGen();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseHttpsRedirection();

app.UseRouting();

app.UseAuthentication(); // proving that you are valid user

app.UseAuthorization();// talks about ur accesss rights

app.UseSwagger();

// Enable middleware to serve swagger-ui (HTML, JS, CSS, etc.),

// specifying the Swagger JSON endpoint.

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "My API");

// c.RoutePrefix = string.Empty;

});

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

}

}

}

appSettings.json File

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"UserConnectionString": "server=LAPTOP-53S2KQS8;database=PracticeWebApi;integrated security=true",

"StudentDBContext": "server=LAPTOP-53S2KQS8;database=BookStore;integrated security=true"

},

**"Jwt": {**

**"Key": "ThisismySecretKey",**

**"Issuer": "UserWebApi"**

**}**

}

**ITokenGeneratorService.cs file**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using WebApplication16.Models;

namespace WebApplication16.Services

{

public interface ITokenGeneratorService

{

string GenerateJSONWebToken(User user);

}

}

**TokenGeneratorService.cs file**

using Microsoft.Extensions.Configuration;

using Microsoft.IdentityModel.Tokens;

using System;

using System.Collections.Generic;

using System.IdentityModel.Tokens.Jwt;

using System.Linq;

using System.Security.Claims;

using System.Text;

using System.Threading.Tasks;

using WebApplication16.Models;

namespace WebApplication16.Services

{

public class TokenGeneratorService : ITokenGeneratorService

{

private IConfiguration \_config;

public TokenGeneratorService(IConfiguration config)

{

\_config = config;

}

string ITokenGeneratorService.GenerateJSONWebToken(User user)

{

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(\_config["Jwt:Key"]));

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

var claims = new Claim[]

{

new Claim(ClaimTypes.Name , user.Id.ToString()),

new Claim(ClaimTypes.Name, user.UserName)

};

var token = new JwtSecurityToken(

issuer: \_config["Jwt:Issuer"],

audience: "CustomerWebApi",

claims: claims,

expires: DateTime.Now.AddMinutes(120),

signingCredentials: credentials);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

}

**AuthenticationController.cs file**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using Microsoft.IdentityModel.Tokens;

using WebApplication16.Models;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using System.IdentityModel.Tokens.Jwt;

using System.Web.Http;

using HttpPostAttribute = Microsoft.AspNetCore.Mvc.HttpPostAttribute;

using FromBodyAttribute = Microsoft.AspNetCore.Mvc.FromBodyAttribute;

using RouteAttribute = Microsoft.AspNetCore.Mvc.RouteAttribute;

using System.Security.Claims;

using HttpGetAttribute = Microsoft.AspNetCore.Mvc.HttpGetAttribute;

using WebApplication16.Services;

namespace WebApplication16.Controllers

{

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

[ApiController]

public class AuthenticationController : ControllerBase

{

private readonly StudentDbContext \_context;

private IConfiguration \_config;

private readonly ITokenGeneratorService service;

public AuthenticationController(StudentDbContext context, IConfiguration config, ITokenGeneratorService \_service)

{

\_context = context;

\_config = config;

service = \_service;

}

//[AllowAnonymous]

[HttpPost]

public IActionResult Login([FromBody] User User)

{

//IActionResult response = Unauthorized();

var user = Authenticate(User);

if (user != null)

{

var tokenString = service.GenerateJSONWebToken(user);

return Ok(new { token = tokenString });

}

else

return BadRequest("Invalid User");

}

//[HttpGet]

//public string GetToken([FromBody] User user)

//{

// return GenerateJSONWebToken(user);

//}

public User Authenticate(User user)

{

User obj = \_context.Users.FirstOrDefault(x => x.UserName == user.UserName && x.Password == user.Password);

return obj;

}

}

}

**EmployeesController.cs (THIS CONTROLLER IS SECURED)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using WebApplication16.Models;

namespace WebApplication16.Controllers

{

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

[ApiController]

[Authorize]

public class EmployeesController : ControllerBase

{

private readonly StudentDbContext \_context;

public EmployeesController(StudentDbContext context)

{

\_context = context;

}

// GET: api/Employees

[HttpGet]

public async Task<ActionResult<IEnumerable<Employee>>> GetEmployees()

{

return await \_context.Employees.ToListAsync();

}

// GET: api/Employees/5

[HttpGet("{id}")]

public async Task<ActionResult<Employee>> GetEmployee(int id)

{

var employee = await \_context.Employees.FindAsync(id);

if (employee == null)

{

return NotFound();

}

else

{

return Ok(employee);

}

}

// PUT: api/Employees/5

// To protect from overposting attacks, enable the specific properties you want to bind to, for

// more details, see https://go.microsoft.com/fwlink/?linkid=2123754.

[HttpPut("{id}")]

public async Task<IActionResult> PutEmployee(int id, Employee employee)

{

\_context.Entry(employee).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!EmployeeExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

// POST: api/Employees

// To protect from overposting attacks, enable the specific properties you want to bind to, for

// more details, see https://go.microsoft.com/fwlink/?linkid=2123754.

[HttpPost]

public async Task<ActionResult<Employee>> PostEmployee(Employee employee)

{

\_context.Employees.Add(employee);

await \_context.SaveChangesAsync();

return Ok(employee);

}

// DELETE: api/Employees/5

[HttpDelete("{id}")]

public async Task<ActionResult<Employee>> DeleteEmployee(int id)

{

var employee = await \_context.Employees.FindAsync(id);

if (employee == null)

{

return NotFound();

}

\_context.Employees.Remove(employee);

await \_context.SaveChangesAsync();

return employee;

}

private bool EmployeeExists(int id)

{

return \_context.Employees.Any(e => e.Id == id);

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**WebClient**

Startup.cs file

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Authentication.Cookies;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.IdentityModel.Tokens;

using WebClient.Controllers;

namespace WebClient

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("ThisismySecretKey"));

services.AddControllersWithViews();

services.AddSession(options =>

{

options.IdleTimeout = TimeSpan.FromMinutes(15);

});

services.AddAuthentication(options =>

{

options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

}).AddJwtBearer(o => o.TokenValidationParameters = new Microsoft.IdentityModel.Tokens.TokenValidationParameters()

{

ValidateIssuerSigningKey = true,

IssuerSigningKey = key,

ValidateIssuer = true,

ValidIssuer = "UserWebApi",

ValidateAudience = true,

ValidAudience = "CustomerWebApi"

});

services.AddSingleton<IHttpContextAccessor, HttpContextAccessor>();

services.AddScoped<EmployeeController>();

services.AddMemoryCache();

//services.AddSession();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

app.UseExceptionHandler("/Home/Error");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseSession();

app.UseAuthentication();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

});

}

}

}

**appSettings.json file**

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*",

"WebAPIBaseUrl": "https://localhost:44371/api/",

"Jwt": {

"Key": "ThisismySecretKey",

"Issuer": "UserWebApi"

}

}

**EmployeeController.cs file**

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Net.Http;

using System.Net.Http.Headers;

using System.Text;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Authentication;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.Logging;

using Newtonsoft.Json;

using WebClient.Models;

namespace WebClient.Controllers

{

//[Authorize(JwtBearerDefaults.AuthenticationScheme)]

// [Authorize]

public class EmployeeController : Controller

{

private readonly ILogger<EmployeeController> \_logger;

**private readonly IHttpContextAccessor \_httpContextAccessor;**

**private readonly ISession \_session;**

**private IConfiguration \_Configure { get; set; }**

**string apiBaseUrl = "";**

public EmployeeController(ILogger<EmployeeController> logger, IConfiguration configuration, **IHttpContextAccessor httpContextAccessor**)

{

\_httpContextAccessor = httpContextAccessor;

**\_session = \_httpContextAccessor.HttpContext.Session;**

\_logger = logger;

\_Configure = configuration;

apiBaseUrl = \_Configure.GetValue<string>("WebAPIBaseUrl");

}

public async Task<IActionResult> Index()

{

if (TempData["msg"] != null)

ViewBag.msg = "<script> alert('Record Inserted'); </script>";

List<EmployeeViewModel> students = new List<EmployeeViewModel>();

using (var client = new HttpClient())

{

**//Send HTTP requests from here.**

**// string endpoint = "https://localhost:44371/api/";**

**client.DefaultRequestHeaders.Clear();**

**var contentType = new MediaTypeWithQualityHeaderValue**

**("application/json");**

**client.DefaultRequestHeaders.Accept.Add(contentType);**

**client.DefaultRequestHeaders.Authorization =**

**new AuthenticationHeaderValue("Bearer",**

**HttpContext.Session.GetString("token"));**

**var request = new HttpRequestMessage(HttpMethod.Get, "https://localhost:44371/api/Employees");**

**HttpResponseMessage response = await client.SendAsync(request);**

**if (response.IsSuccessStatusCode)**

**{**

**var jsonString = response.Content.ReadAsStringAsync().Result;**

**students = JsonConvert.DeserializeObject<List<EmployeeViewModel>>(jsonString);**

**}**

**return View(students);**

}

}

**// SIMILARLY ADD IT TO OTHER METHODS**

public async Task<IActionResult> Details(int Id)

{

EmployeeViewModel student = new EmployeeViewModel();

using (var client = new HttpClient())

{

//Send HTTP requests from here.

// string endpoint = "https://localhost:44371/api/";

client.BaseAddress = new Uri(apiBaseUrl);

//GET Method

HttpResponseMessage response = await client.GetAsync($"EMployees/{Id}");

if (response.IsSuccessStatusCode)

{

var jsonString = response.Content.ReadAsStringAsync().Result;

student = JsonConvert.DeserializeObject<EmployeeViewModel>(jsonString);

return View(student);

}

else

{

ViewBag.msg = response.StatusCode;

return View();

}

}

}

public IActionResult Create()

{

EmployeeViewModel employee = new EmployeeViewModel();

return View(employee);

}

[HttpPost]

public IActionResult Create(EmployeeViewModel employee)

{

HttpClient client = new HttpClient();

{

StringContent content = new StringContent(JsonConvert.SerializeObject(employee), Encoding.UTF8, "application/json");

string endpoint = "https://localhost:44371/api/employees";

client.BaseAddress = new Uri(endpoint);

var Response = client.PostAsync(endpoint, content);

Response.Wait();

var result = Response.Result;

if (result.IsSuccessStatusCode)

{

TempData["msg"] = "Record Inserted";

return RedirectToAction("Index");

}

else if (Response.Result.StatusCode == System.Net.HttpStatusCode.Conflict)

{

ModelState.Clear();

ModelState.AddModelError("Id", "Id Already Exist");

return View();

}

else

{

TempData["msg"] = Response.Result.StatusCode;

return RedirectToAction("Index");

}

}

}

public async Task<IActionResult> Delete(int Id)

{

EmployeeViewModel student = new EmployeeViewModel();

using (var client = new HttpClient())

{

//Send HTTP requests from here.

string endpoint = "https://localhost:44371/api/";

client.BaseAddress = new Uri(endpoint);

//GET Method

HttpResponseMessage response = await client.GetAsync($"EMployees/{Id}");

if (response.IsSuccessStatusCode)

{

var jsonString = response.Content.ReadAsStringAsync().Result;

student = JsonConvert.DeserializeObject<EmployeeViewModel>(jsonString);

return View(student);

}

else

{

ViewBag.msg = response.StatusCode;

return View();

}

}

}

[HttpPost]

public async Task<IActionResult> Delete(int id, EmployeeViewModel employee)

{

using (var client = new HttpClient())

{

//Send HTTP requests from here.

string endpoint = "https://localhost:44371/api/";

client.BaseAddress = new Uri(endpoint);

HttpResponseMessage response = await client.DeleteAsync($"employees/{id}");

}

TempData["msg"] = "Record Deleted";

return RedirectToAction("Index");

}

}

}