**Deep Skilling Week 4 Assignment**

1. **First Web Api using .Net core**

namespace MyFirstApi.Web;

public class WeatherApiClient(HttpClient httpClient)

{

public async Task<WeatherForecast[]> GetWeatherAsync(int maxItems = 10, CancellationToken cancellationToken = default)

{

List<WeatherForecast>? forecasts = null;

await foreach (var forecast in httpClient.GetFromJsonAsAsyncEnumerable<WeatherForecast>("/weatherforecast", cancellationToken))

{

if (forecasts?.Count >= maxItems)

{

break;

}

if (forecast is not null)

{

forecasts ??= [];

forecasts.Add(forecast);

}

}

return forecasts?.ToArray() ?? [];

}

}

public record WeatherForecast(DateOnly Date, int TemperatureC, string? Summary)

{

public int TemperatureF => 32 + (int)(TemperatureC / 0.5556);

}

1. **Web Api using .Net core with Swagger**

builder.Services.AddSwaggerGen(c =>{

c.SwaggerDoc("v1", new Microsoft.OpenApi.Models.OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "TBD",

TermsOfService = new Uri("https://example.com/terms"),

Contact = new Microsoft.OpenApi.Models.OpenApiContact

{

Name = "John Doe",

Email = "john@xyzmail.com",

Url = new Uri("https://www.example.com")

},

License = new Microsoft.OpenApi.Models.OpenApiLicense

{

Name = "License Terms",

Url = new Uri("https://www.example.com")

}

});

});

1. **Web Api using custom model class**

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

}

public class Skill

{

public int Id { get; set; }

public string Name { get; set; }

}

public class Employee

{

public int Id { get; set; }

public string Name { get; set; }

public int Salary { get; set; }

public bool Permanent { get; set; }

public Department Department { get; set; }

public List<Skill> Skills { get; set; }

public DateTime DateOfBirth { get; set; }

}

using Microsoft.AspNetCore.Mvc;

using WebApi.Models; // adjust namespace

[ApiController]

[Route("emp")]

public class EmployeeController : ControllerBase

{

private readonly List<Employee> \_employees;

public EmployeeController()

{

\_employees = GetStandardEmployeeList();

}

[HttpGet]

[AllowAnonymous]

[ProducesResponseType(StatusCodes.Status200OK)]

public ActionResult<List<Employee>> Get()

{

// Uncomment to test exception

// throw new Exception("Test exception");

return Ok(\_employees);

}

[HttpGet("standard")]

public ActionResult<Employee> GetStandard()

{

return Ok(\_employees.FirstOrDefault());

}

[HttpPost]

public IActionResult Create([FromBody] Employee emp)

{

\_employees.Add(emp);

return Created("", emp);

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John",

Salary = 50000,

Permanent = true,

Department = new Department { Id = 1, Name = "IT" },

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "C#" },

new Skill { Id = 2, Name = "SQL" }

},

DateOfBirth = new DateTime(1990, 1, 1)

}

};

}

}

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

var exception = context.Exception;

File.AppendAllText("logs.txt", $"[{DateTime.Now}] {exception.Message}\n");

context.Result = new ObjectResult("Something went wrong. Please try again later.")

{

StatusCode = 500

};

}

}

**4 Web Api CRUD operation**

public class EmployeeController : ControllerBase

{

private static List<Employee> \_employees = new()

{

new Employee

{

Id = 1,

Name = "John",

Salary = 50000,

Permanent = true,

Department = new Department { Id = 1, Name = "IT" },

Skills = new List<Skill> { new Skill { Id = 1, Name = "C#" } },

DateOfBirth = new DateTime(1990, 1, 1)

},

new Employee

{

Id = 2,

Name = "Jane",

Salary = 60000,

Permanent = false,

Department = new Department { Id = 2, Name = "HR" },

Skills = new List<Skill> { new Skill { Id = 2, Name = "Excel" } },

DateOfBirth = new DateTime(1992, 5, 5)

}

};

[HttpPut("{id}")]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

public ActionResult<Employee> UpdateEmployee(int id, [FromBody] Employee updatedEmployee)

{

if (id <= 0)

return BadRequest("Invalid employee id");

var existingEmployee = \_employees.FirstOrDefault(e => e.Id == id);

if (existingEmployee == null)

return BadRequest("Invalid employee id");

// Update employee fields

existingEmployee.Name = updatedEmployee.Name;

existingEmployee.Salary = updatedEmployee.Salary;

existingEmployee.Permanent = updatedEmployee.Permanent;

existingEmployee.Department = updatedEmployee.Department;

existingEmployee.Skills = updatedEmployee.Skills;

existingEmployee.DateOfBirth = updatedEmployee.DateOfBirth;

return Ok(existingEmployee);

}

**5.JsonWebToken**

var MyAllowSpecificOrigins = "\_myAllowSpecificOrigins";

builder.Services.AddCors(options =>

{

options.AddPolicy(name: MyAllowSpecificOrigins,

policy =>

{

policy.WithOrigins("http://localhost:4200") // allow Angular app

.AllowAnyHeader()

.AllowAnyMethod();

});

});

app.UseCors(MyAllowSpecificOrigins);

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

string securityKey = "mysuperdupersecret";

var symmetricSecurityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(securityKey));

builder.Services.AddAuthentication(x =>

{

x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = "mySystem",

ValidAudience = "myUsers",

IssuerSigningKey = symmetricSecurityKey

};

});

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

[Route("auth")]

[ApiController]

[AllowAnonymous]

public class AuthController : ControllerBase

{

[HttpGet("token")]

public IActionResult GetToken()

{

var token = GenerateJSONWebToken(1001, "Admin");

return Ok(new { Token = token });

}

private string GenerateJSONWebToken(int userId, string userRole)

{

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecret"));

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

var claims = new List<Claim>

{

new Claim(ClaimTypes.Role, userRole),

new Claim("UserId", userId.ToString())

};

var token = new JwtSecurityToken(

issuer: "mySystem",

audience: "myUsers",

claims: claims,

expires: DateTime.Now.AddMinutes(2), // change to 10 for longer life

signingCredentials: credentials);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}