**WEEK 4-WEB API HANDS ON**

**Exercise 1:**

[ApiController]

[Route("[controller]")]

public class ValuesController : ControllerBase

{

[HttpGet]

public IEnumerable<string> Get()

{

return new string[] { "value1", "value2" };

}

[HttpGet("{id}")]

public string Get(int id)

{

return "value";

}

[HttpPost]

public void Post([FromBody] string value)

{

}

[HttpPut("{id}")]

public void Put(int id, [FromBody] string value)

{

}

[HttpDelete("{id}")]

public void Delete(int id)

{

}

}

**Exercise 2:**

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")

}

});

});

app.UseSwagger();

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "Swagger Demo");

});

**Exercise 3:**

**Department.cs:**  
namespace EmployeeApi.Models

{

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

}

}

**Employee.cs**:

using System;

using System.Collections.Generic;

namespace EmployeeApi.Models

{

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; }

}

}

**Skills.cs:**

namespace EmployeeApi.Models

{

public class Skill

{

public int Id { get; set; }

public string Name { get; set; }

}

}

**EmployeeController.cs:**using Microsoft.AspNetCore.Mvc;

using EmployeeApi.Models;

using Microsoft.AspNetCore.Authorization;

using EmployeeApi.Filters;

[ApiController]

[Route("[controller]")]

[Authorize(Roles = "POC")]

//[ServiceFilter(typeof(CustomAuthFilter))]

[AllowAnonymous]

public class EmployeeController : ControllerBase

{

private readonly List<Employee> \_employees;

public EmployeeController()

{

\_employees = GetStandardEmployeeList();

}

[HttpGet("GetStandard")]

//[AllowAnonymous]

[ProducesResponseType(typeof(List<Employee>), 200)]

[ProducesResponseType(500)]

public ActionResult<List<Employee>> GetStandard()

{

return \_employees;

}

[HttpPost]

public ActionResult AddEmployee([FromBody] Employee emp)

{

\_employees.Add(emp);

return Ok(new

{

message = "Employee added",

employee = emp

});

}

[HttpPut]

public ActionResult<Employee> UpdateEmployee([FromBody] Employee emp)

{

if (emp.Id <= 0)

return BadRequest("Invalid employee id");

var existingEmp = \_employees.FirstOrDefault(e => e.Id == emp.Id);

if (existingEmp == null)

return BadRequest("Invalid employee id");

existingEmp.Name = emp.Name;

existingEmp.Salary = emp.Salary;

existingEmp.Permanent = emp.Permanent;

existingEmp.Department = emp.Department;

existingEmp.Skills = emp.Skills;

existingEmp.DateOfBirth = emp.DateOfBirth;

return Ok(existingEmp);

}

[HttpDelete("{id}")]

public ActionResult DeleteEmployee(int id)

{

if (id <= 0)

return BadRequest("Invalid employee id");

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

if (empToDelete == null)

return BadRequest("Invalid employee id");

\_employees.Remove(empToDelete);

return Ok($"Employee with ID {id} has been deleted.");

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "Emma",

Salary = 60000,

Permanent = true,

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

Skills = new List<Skill>

{

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

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

},

DateOfBirth = new DateTime(1997, 5, 12)

}

};

}

}

**Exercise 4:**

**CustomAuthFilter.cs:**  
using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace EmployeeApi.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

if (!context.HttpContext.Request.Headers.TryGetValue("Authorization", out var token))

{

context.Result = new BadRequestObjectResult("Invalid request - No Auth token");

return;

}

if (!token.ToString().Contains("Bearer"))

{

context.Result = new BadRequestObjectResult("Invalid request - Token present but Bearer unavailable");

return;

}

base.OnActionExecuting(context);

}

}

}

**Program.cs:**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

using EmployeeApi.Filters;

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

var key = "secretlooooongkeys1234567adgezedffe";

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

builder.Services.AddControllers(options =>

{

options.Filters.Add<CustomExceptionFilter>();

});

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(options =>

{

options.SwaggerDoc("v1", new OpenApiInfo { Title = "Employee API", Version = "v1" });

options.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

Name = "Authorization",

Type = SecuritySchemeType.ApiKey,

Scheme = "Bearer",

BearerFormat = "JWT",

In = ParameterLocation.Header,

Description = "Enter: Bearer <your JWT token>"

});

options.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme {

Reference = new OpenApiReference {

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

new string[] {}

}

});

});

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddJwtBearer(options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = "mySystem",

ValidAudience = "myUsers",

IssuerSigningKey = symmetricSecurityKey

};

});

builder.Services.AddScoped<CustomAuthFilter>();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

**Exercise 5:**

**AuthController.cs:**using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

using Microsoft.AspNetCore.Authorization;

[ApiController]

[Route("[controller]")]

[Authorize]

public class AuthController : ControllerBase

{

[HttpGet("token")]

[AllowAnonymous]

public IActionResult GetToken()

{

var token = GenerateJSONWebToken(1, "Admin");

return Ok(new { token });

}

private string GenerateJSONWebToken(int userId, string userRole)

{

var key = "secretlooooongkeys1234567adgezedffe";

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

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),

signingCredentials: credentials);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}