# **Name- Sayandeep Dey (SupersetID:** **6363427)**

# **WEEK – 4 (Handson- Exercises)**

1. **ASP.NET Core 8.0 Web API:**

**LAB-1:** **WebApi\_Handson:-**

### **Code:**

### **Create First Web API (Read/Write)**

**Create Project**

1. Create New Project → **ASP.NET Core Web API**
2. Name: FirstWebApi
3. Select **.NET 8.0**, keep OpenAPI/Swagger checked

Controllers/WeatherForecastController.cs – sample controller

Program.cs – contains builder.Services and app.MapControllers()

appsettings.json – config file

launchSettings.json – controls how the app runs (e.g., port, HTTPS)

**In ValuesController.cs:**

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

namespace SimpleWebApiDemo.Controllers

{

[ApiController]

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

public class ValuesController : ControllerBase

{

private static List<string> values = new List<string> { "PEN", "PENCIL" };

// GET: api/values

[HttpGet]

public IActionResult Get()

{

return Ok(values);

}

// POST: api/values

[HttpPost]

public IActionResult Post([FromBody] string newValue)

{

values.Add(newValue);

return Ok($"Added: {newValue}");

}

[HttpPut("{index}")]

public IActionResult Put(int index, [FromBody] string updatedValue)

{

if (index < 0 || index >= values.Count)

return BadRequest("Invalid index");

values[index] = updatedValue;

return Ok($"Updated at index {index}");

}

[HttpDelete("{index}")]

public IActionResult Delete(int index)

{

if (index < 0 || index >= values.Count)

return NotFound("Index not found");

values.RemoveAt(index);

return Ok($"Deleted value at index {index}");

}

}

}

**Output:  
A screenshot of a computer

AI-generated content may be incorrect.**

**The Full Screensort:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Lab-2:** **WebApi\_Handson:**

**Code:**

**In Program.cs:**

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer(); // required for Swagger UI

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "TBD",

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

Contact = new OpenApiContact

{

Name = "John Doe",

Email = "john@xyzmail.com",

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

},

License = new OpenApiLicense

{

Name = "License Terms",

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

}

});

});

var app = builder.Build();

// Configure the HTTP request pipeline.

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

**Output:  
in Swagger:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**In Postman:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Lab: 3:** **WebApi\_Handson:**

**Code:**

**In Program.cs:**

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

builder.Services.AddScoped<CustomAuthFilter>();

// Register the exception filter

builder.Services.AddScoped<CustomExceptionFilter>();

// Add it globally to MVC pipeline

builder.Services.AddControllers(options =>

{

options.Filters.Add<CustomExceptionFilter>();

});

builder.Services.AddEndpointsApiExplorer(); // ✅ required for Swagger UI

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "Demo API with Swagger, Filters, and Auth"

});

// 🔐 Add JWT Bearer Security Definition

c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

In = ParameterLocation.Header,

Description = "Enter 'Bearer <token>'",

Name = "Authorization",

Type = SecuritySchemeType.ApiKey,

Scheme = "Bearer"

});

c.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

new string[] {}

}

});

});

var app = builder.Build();

// Configure the HTTP request pipeline.

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

**In Employee.cs:**

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; }

}

**In Department.cs:**

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

}

**In Skill.cs:**public class Skill

{

public int Id { get; set; }

public string Name { get; set; }

}

**In CustomAuthFilter.cs:**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

var hasHeader = context.HttpContext.Request.Headers.TryGetValue("Authorization", out var token);

if (!hasHeader)

{

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");

}

}

}

**In CustomExceptionFilter.cs:**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System.IO;

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

string logPath = Path.Combine(Directory.GetCurrentDirectory(), "logs.txt");

File.AppendAllText(logPath, $"{DateTime.Now}: {context.Exception.Message}\n");

context.Result = new ObjectResult("An unexpected error occurred")

{

StatusCode = 500

};

context.ExceptionHandled = true;

}

}

**In EmployeeController.cs:**

using Microsoft.AspNetCore.Mvc;

[ApiController]

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

[ServiceFilter(typeof(CustomAuthFilter))]

public class EmployeeController : ControllerBase

{

private List<Employee> employees;

public EmployeeController()

{

employees = GetStandardEmployeeList();

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John",

Salary = 50000,

Permanent = true,

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

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

DateOfBirth = new DateTime(1990, 1, 1)

}

};

}

[HttpGet("standard")]

[ProducesResponseType(StatusCodes.Status200OK)]

public ActionResult<List<Employee>> GetStandard()

{

return Ok(employees);

}

[HttpPost]

public IActionResult AddEmployee([FromBody] Employee employee)

{

employees.Add(employee);

return Ok(employee);

}

[HttpGet("error")]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public IActionResult GetWithError()

{

throw new Exception("Test exception for filter");

}

}

**Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**With Authorization:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Log File:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Lab-4 - WebApi\_Handson:**

**Code:**

**In EmployeeController.cs:**

using Microsoft.AspNetCore.Mvc;

[ApiController]

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

[ServiceFilter(typeof(CustomAuthFilter))]

public class EmployeeController : ControllerBase

{

private List<Employee> employees;

public EmployeeController()

{

employees = GetStandardEmployeeList();

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John",

Salary = 50000,

Permanent = true,

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

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

DateOfBirth = new DateTime(1990, 1, 1)

}

};

}

[HttpGet("standard")]

[ProducesResponseType(StatusCodes.Status200OK)]

public ActionResult<List<Employee>> GetStandard()

{

return Ok(employees);

}

[HttpPost]

public IActionResult AddEmployee([FromBody] Employee employee)

{

employees.Add(employee);

return Ok(employee);

}

[HttpGet("error")]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public IActionResult GetWithError()

{

throw new Exception("Test exception for filter");

}

[HttpPut]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

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

{

if (updatedEmployee.Id <= 0)

return BadRequest("Invalid employee id");

var existingEmployee = employees.FirstOrDefault(e => e.Id == updatedEmployee.Id);

if (existingEmployee == null)

return BadRequest("Invalid employee id");

// Update fields manually

existingEmployee.Name = updatedEmployee.Name;

existingEmployee.Salary = updatedEmployee.Salary;

existingEmployee.Permanent = updatedEmployee.Permanent;

existingEmployee.DateOfBirth = updatedEmployee.DateOfBirth;

existingEmployee.Department = updatedEmployee.Department;

existingEmployee.Skills = updatedEmployee.Skills;

return Ok(existingEmployee);

}

}

**Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Lab- 5: WebApi\_Handson:**

**Code:**

**In program.cs:**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

var key = "mysuperdupersecretkeythatistotallysecure123";

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

// Services

builder.Services.AddControllers();

builder.Services.AddCors(options =>

{

    options.AddPolicy("AllowLocalhost", policy =>

    {

        policy.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader();

    });

});

builder.Services.AddAuthentication(opt =>

{

    opt.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

    opt.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

}).AddJwtBearer(opt =>

{

    opt.TokenValidationParameters = new TokenValidationParameters

    {

        ValidateIssuer = true,

        ValidateAudience = true,

        ValidateLifetime = true,

        ValidateIssuerSigningKey = true,

        ValidIssuer = "mySystem",

        ValidAudience = "myUsers",

        IssuerSigningKey = securityKey

    };

});

builder.Services.AddAuthorization();

// 🔐 Swagger JWT setup

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(options =>

{

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

    options.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

    {

        Name = "Authorization",

        Type = SecuritySchemeType.ApiKey,

        Scheme = "Bearer",

        BearerFormat = "JWT",

        In = ParameterLocation.Header,

        Description = "Enter 'Bearer' [space] and your token"

    });

    options.AddSecurityRequirement(new OpenApiSecurityRequirement

    {

        {

            new OpenApiSecurityScheme

            {

                Reference = new OpenApiReference { Type = ReferenceType.SecurityScheme, Id = "Bearer" }

            },

            new string[] {}

        }

    });

});

var app = builder.Build();

app.UseCors("AllowLocalhost");

app.UseAuthentication();

app.UseAuthorization();

app.UseSwagger();

app.UseSwaggerUI();

app.MapControllers();

app.Run();

**In AuthController.cs:**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Authorization;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

[ApiController]

[Route("[controller]")]

public class AuthController : ControllerBase

{

    [HttpGet("token")]

    [AllowAnonymous]

    public IActionResult GetToken()

    {

        var token = GenerateToken(1, "Admin"); // Change role to test

        return Ok(new { Token = token });

    }

    private string GenerateToken(int userId, string userRole)

    {

        var key = "mysuperdupersecretkeythatistotallysecure123";

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

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

        var claims = new[]

        {

            new Claim(ClaimTypes.Role, userRole),

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

        };

        var token = new JwtSecurityToken(

            issuer: "mySystem",

            audience: "myUsers",

            claims: claims,

            expires: DateTime.Now.AddMinutes(10),

            signingCredentials: credentials

        );

        return new JwtSecurityTokenHandler().WriteToken(token);

    }

}

**In EmployeeController.cs:**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

[ApiController]

[Route("[controller]")]

[Authorize(Roles = "Admin,POC")]

public class EmployeeController : ControllerBase

{

    [HttpGet]

    public IActionResult Get()

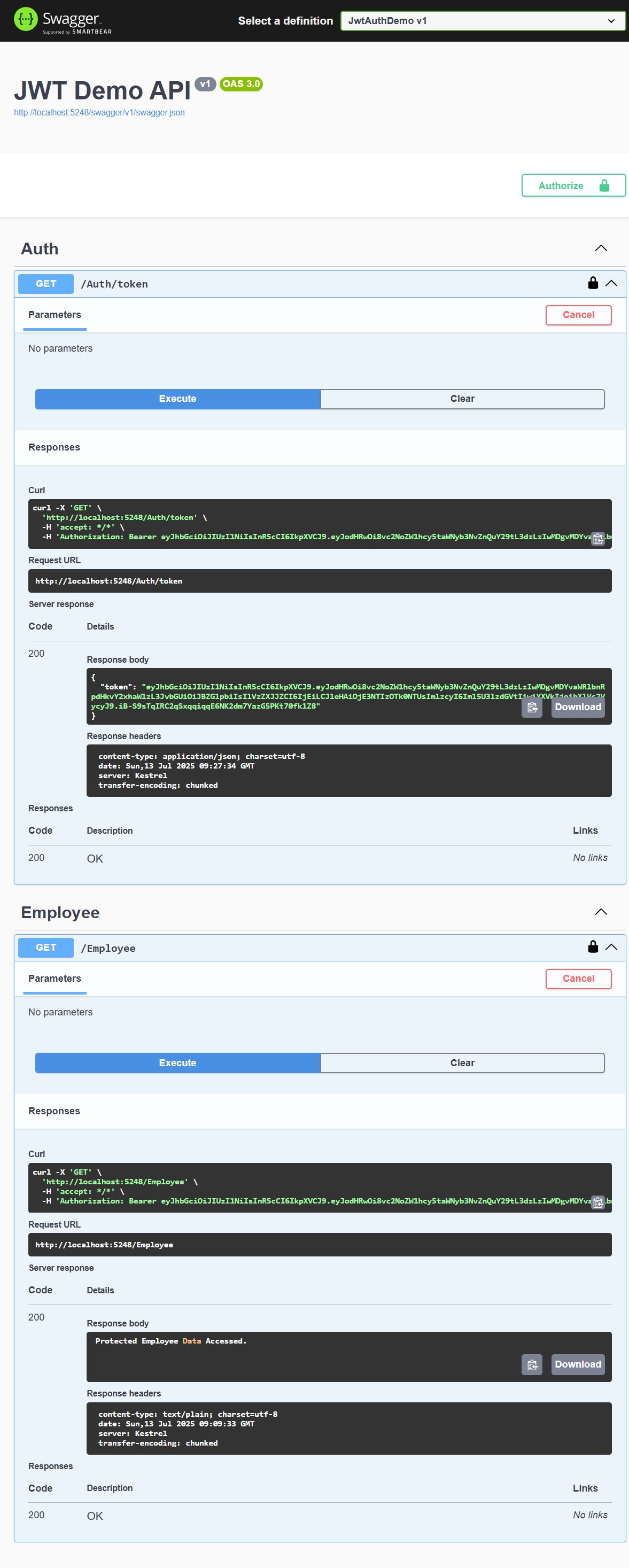
    {

        return Ok(" Protected Employee Data Accessed.");

    }

}

**Output:  
In Swagger:**

****

**In Postman:**

**GET** [**http://localhost:5248/auth/token**](http://localhost:5248/auth/token)**:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Copied the token then**

**Run GET** [**http://localhost:5248/employee**](http://localhost:5248/employee)

**Authorization > Bearer Token > Paste the Token > Send>**

**A screenshot of a computer

AI-generated content may be incorrect.**