Name-Sangram Mandal

Superset Id-6363848

Week-4(Handson-Exercise)

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" };

[HttpGet]

public IActionResult Get()

{

return Ok(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);

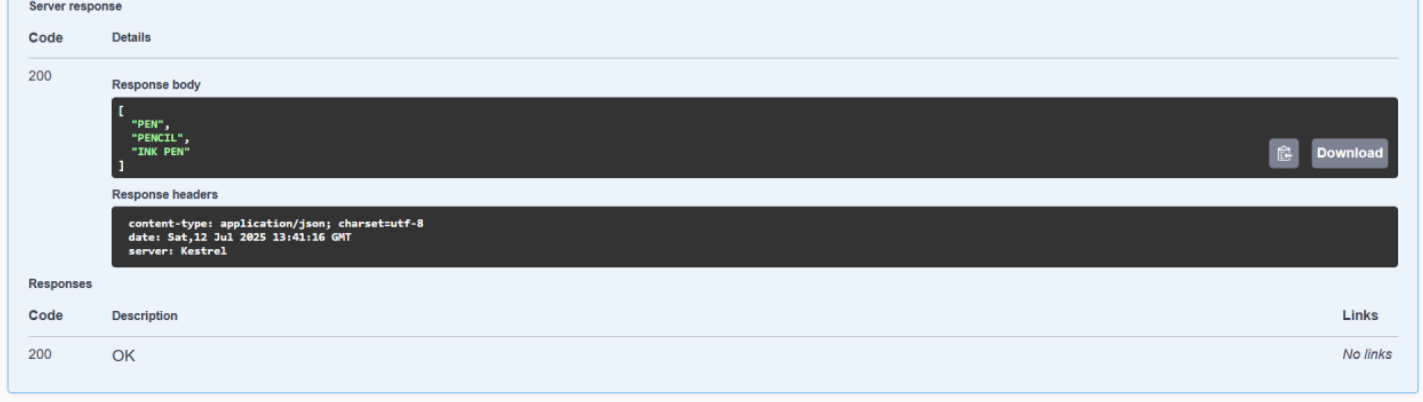
return Ok($"Deleted value at index {index}");

}

}

}

**Output:**



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

**Code:**

**In Program.cs:**

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

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();

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

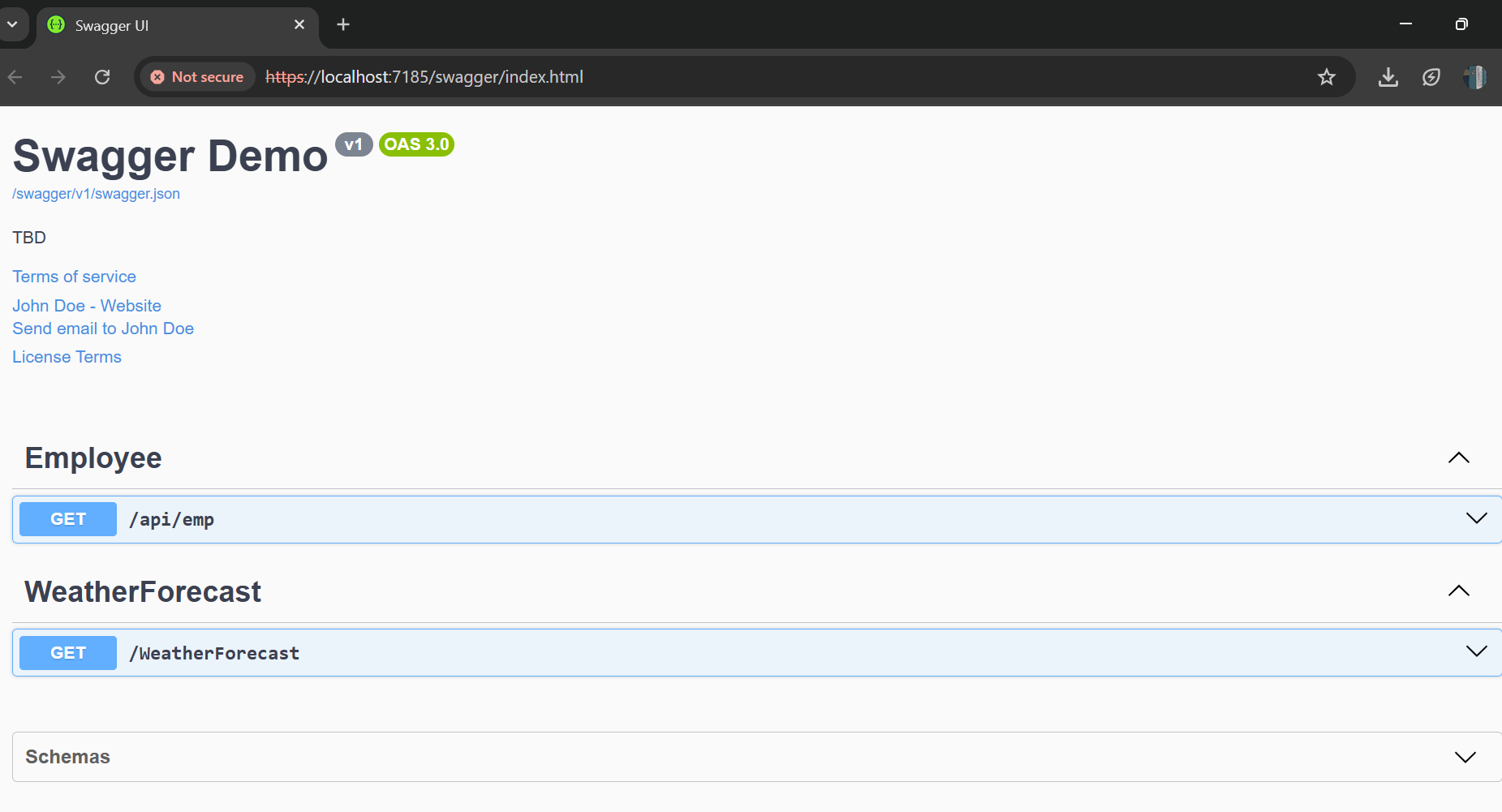
app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

**Output:  
in Swagger:**



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

**Code:**

**In Program.cs:**

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddScoped<CustomAuthFilter>();

builder.Services.AddScoped<CustomExceptionFilter>();

builder.Services.AddControllers(options =>

{

options.Filters.Add<CustomExceptionFilter>();

});

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

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

});

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();

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:**

****

**With Authorization:**



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

**Code:**

**In EmployeeController.cs:**

using Microsoft.AspNetCore.Mvc;

using Webapi\_CRUD.Models;

namespace Webapi\_CRUD.Controllers

{

[ApiController]

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

public class EmployeeController : ControllerBase

{

private static List<Employee> employees = 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]

public ActionResult<List<Employee>> GetAll()

{

return Ok(employees);

}

[HttpPut("{id}")]

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");

}

existingEmployee.Name = updatedEmployee.Name;

existingEmployee.Salary = updatedEmployee.Salary;

existingEmployee.Permanent = updatedEmployee.Permanent;

existingEmployee.Department = updatedEmployee.Department;

existingEmployee.Skills = updatedEmployee.Skills;

existingEmployee.DateOfBirth = updatedEmployee.DateOfBirth;

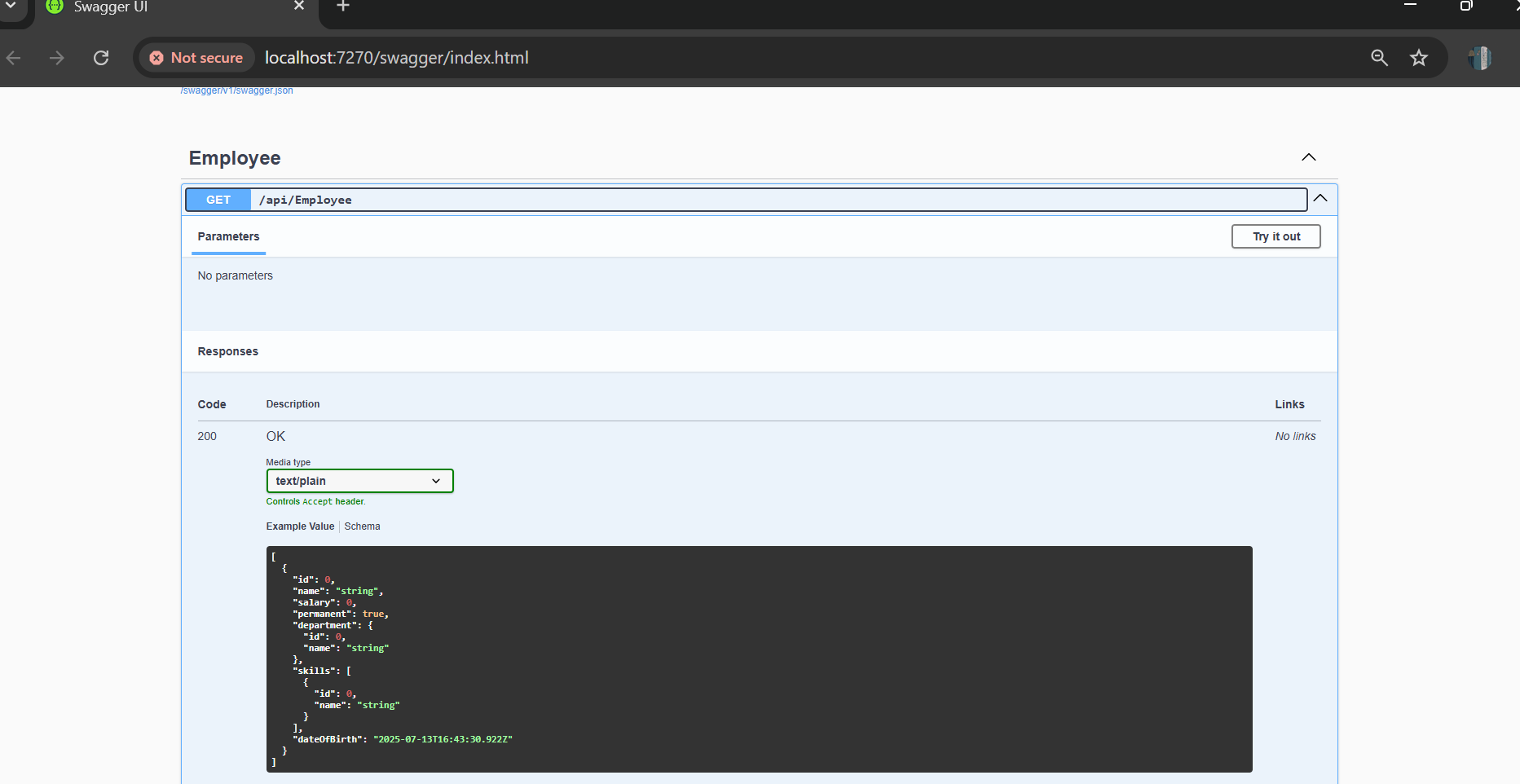
return Ok(existingEmployee);

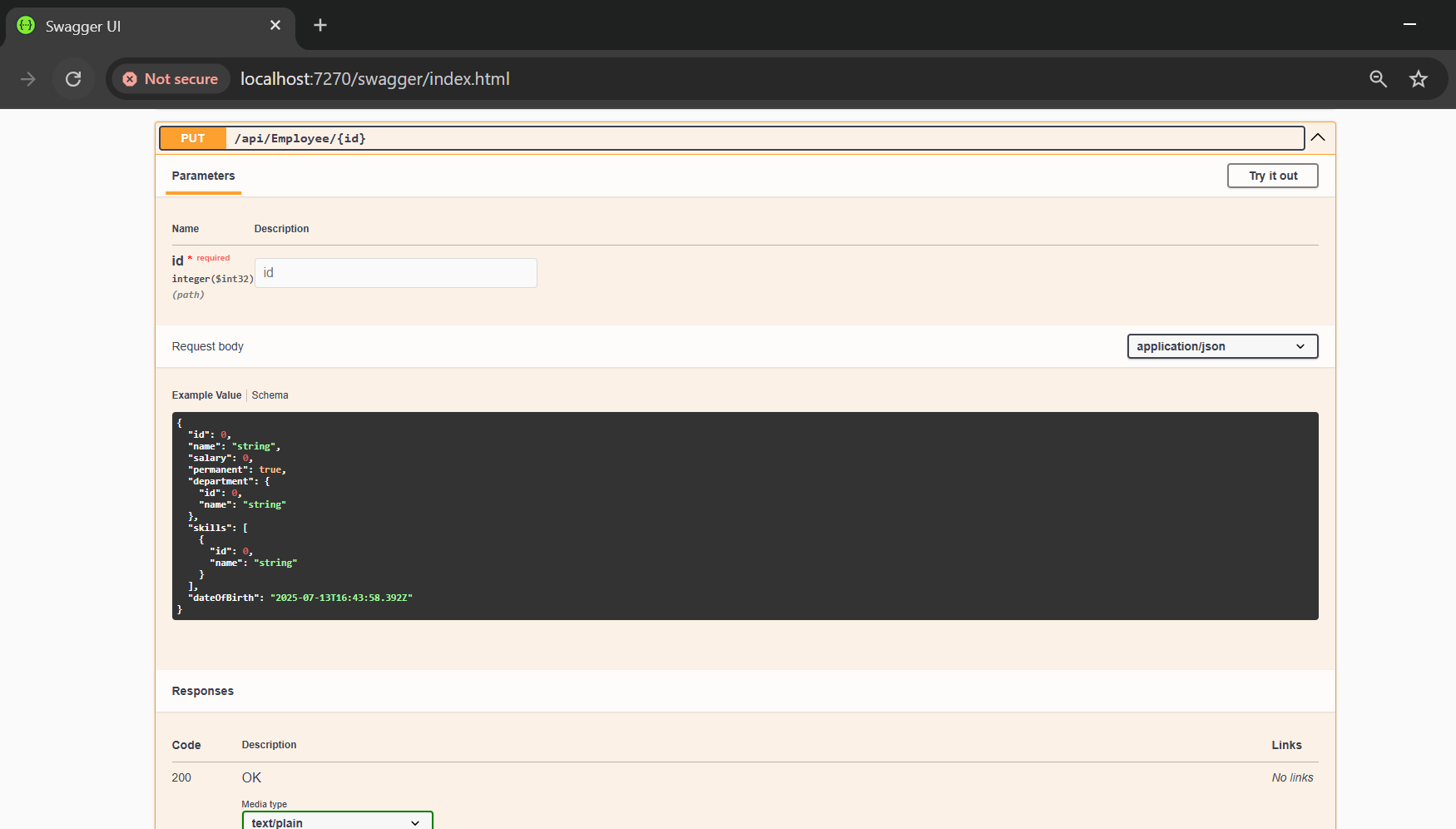
}

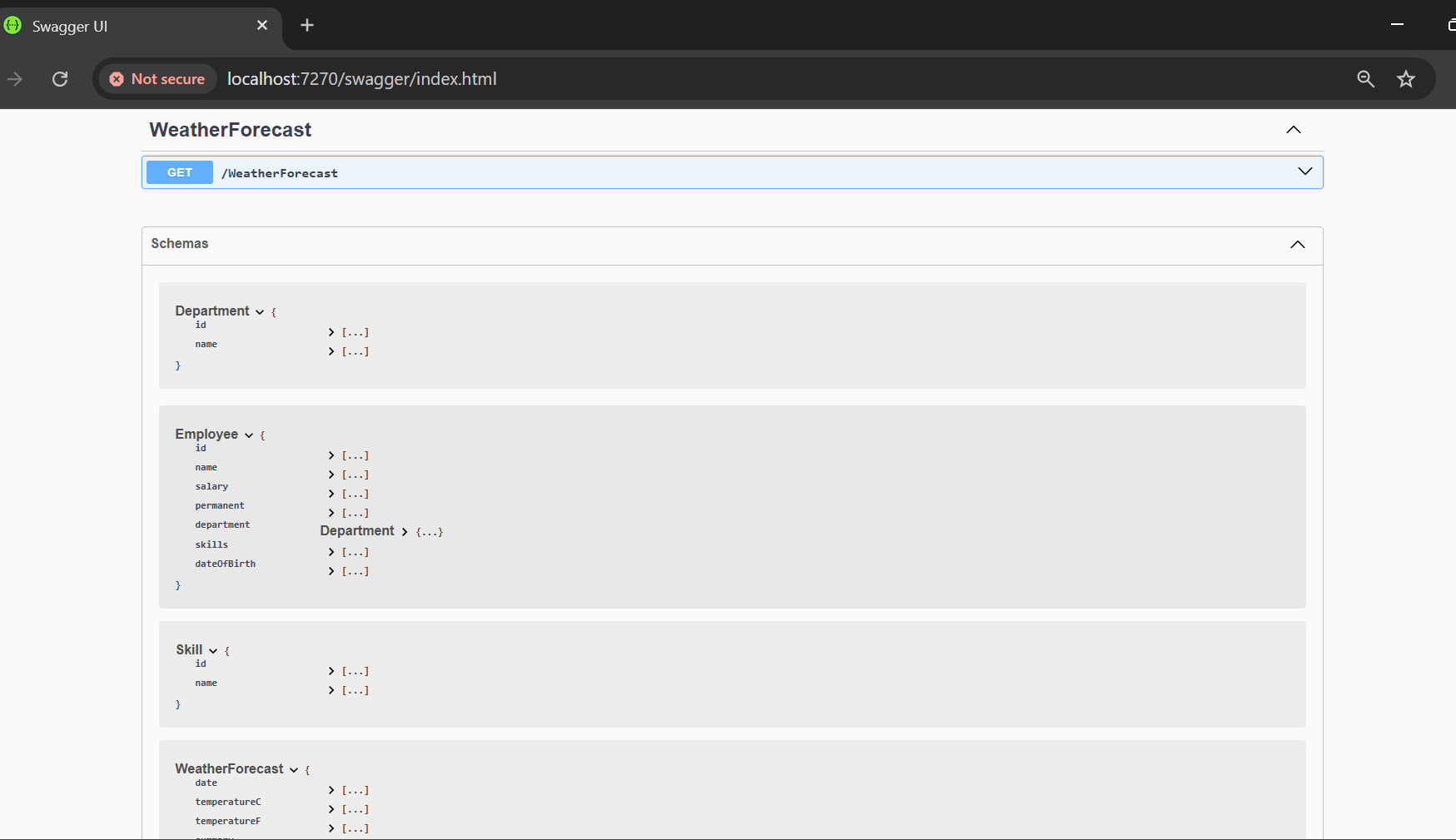
}

}

**Output:**







**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));

builder.Services.AddControllers();

builder.Services.AddCors(options =>

{

options.AddPolicy("AllowLocalhost", policy =>

{

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

});

});

builder.Services.AddAuthentication(options =>

{

options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = "mySystem",

ValidAudience = "myUsers",

IssuerSigningKey = securityKey

};

});

builder.Services.AddAuthorization();

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' followed by the token"

});

options.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

Array.Empty<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");

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:**

