WEEK-3 WEBAPI\_HANDSON

### RESTful Web Service, Web API & Microservice:

* **REST (Representational State Transfer):**
  + Architectural style using HTTP.
  + **Stateless**, uses **standard HTTP verbs**.
  + Communicates with **JSON/XML**, but not limited to XML.
* **Web API:**
  + Framework to expose RESTful services over HTTP.
  + Web APIs can be consumed by web/mobile clients.
* **Microservice:**
  + A small, independently deployable service.
  + Implements a single business functionality.
  + Web APIs can be used as microservices.

### HTTP Request/Response:

* **HttpRequest** – carries data from client → server.
* **HttpResponse** – carries result from server → client.

### Action Verbs:

| Verb | Use | Attribute |
| --- | --- | --- |
| GET | Read data | [HttpGet] |
| POST | Create new resource | [HttpPost] |
| PUT | Update resource completely | [HttpPut] |
| DELETE | Delete resource | [HttpDelete] |

### HttpStatusCodes (commonly used):

| Status | Code | Purpose |
| --- | --- | --- |
| OK | 200 | Successful operation |
| BadRequest | 400 | Validation/format error |
| Unauthorized | 401 | No valid auth |
| InternalServerError | 500 | Server-side error |

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

// Add services

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Middleware

if (app.Environment.IsDevelopment())

{

    app.UseSwagger();

    app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

// Enable attribute-based routing (for controllers)

app.MapControllers();

app.Run();

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

namespace MyFirstWebAPI.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    public class ValuesController : ControllerBase

    {

        private static readonly List<string> values = new() { "value1", "value2" };

        [HttpGet]

        public IEnumerable<string> Get() => values;

        [HttpGet("{id}")]

        public IActionResult Get(int id)

        {

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

                return NotFound();

            return Ok(values[id]);

        }

        [HttpPost]

        public IActionResult Post([FromBody] string value)

        {

            values.Add(value);

            return CreatedAtAction(nameof(Get), new { id = values.Count - 1 }, value);

        }

        [HttpDelete("{id}")]

        public IActionResult Delete(int id)

        {

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

                return NotFound();

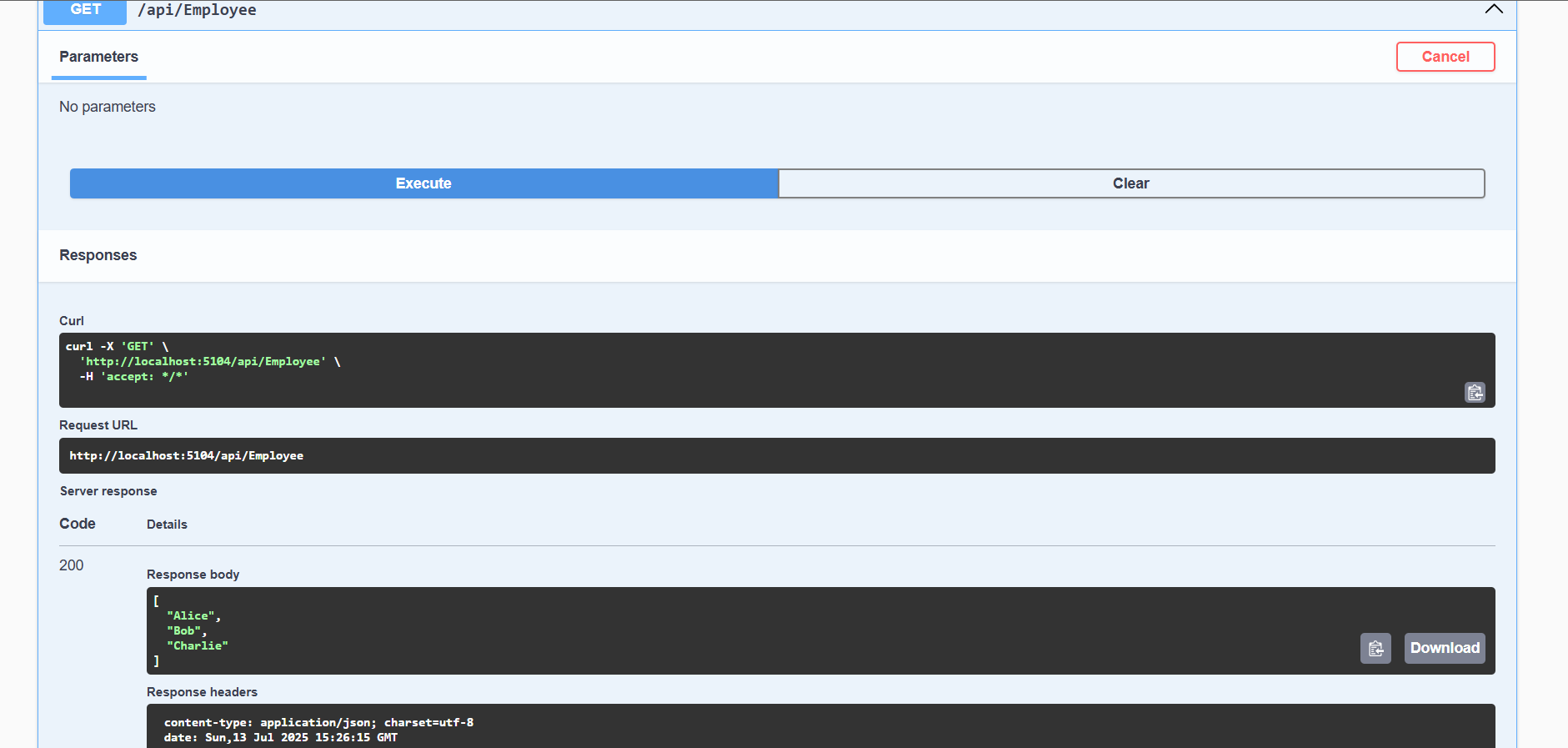
            values.RemoveAt(id);

            return NoContent();

        }

    }

}



#2 WEBAPI HANDSON

Step 1: Install Swashbuckle.AspNetCore via NuGet

Step 2: Configure Swagger in Program.cs

Step 3: Add [ProducesResponseType] in Controller

1. Demonstrate the usage of Postman to hit WebAPI Methods

## ****Methods****

### ****Structure in Postman****

* **Collections**: Group of saved API requests.
* You can **Create a new Collection** and then **Add requests** to it.

### ****Creating a New Request in a Collection****

1. Click **"New" > Request**
2. Choose request type (GET, POST, etc.)
3. Set **URL**, **Headers**, and **Body**
4. Save the request in a collection
5. Usage of Route, Name Attribute, and ActionName

The Name attribute assigns a friendly name to the route.

1. Useful for Url.Link("GetProductById", new { id = 1 })

4. Simple WebAPI with Read and Write Actions

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

// Add controller services

builder.Services.AddControllers();

// Add SwaggerGen with metadata

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://example.com")

        },

        License = new OpenApiLicense

        {

            Name = "License Terms",

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

        }

    });

});

var app = builder.Build();

app.UseHttpsRedirection();

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

app.MapControllers();

app.Run();

using Microsoft.AspNetCore.Mvc;

namespace SwaggerDemoAPI.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    public class EmployeeController : ControllerBase

    {

        private static List<string> employees = new() { "Alice", "Bob", "Charlie" };

        [HttpGet]

        [ProducesResponseType(StatusCodes.Status200OK)]

        [ProducesResponseType(StatusCodes.Status500InternalServerError)]

        [ActionName("ListEmployees")]

        public IActionResult Get()

        {

            return Ok(employees);

        }

        [HttpPost]

        [ProducesResponseType(StatusCodes.Status200OK)]

        [ProducesResponseType(StatusCodes.Status400BadRequest)]

        [ActionName("AddEmployee")]

        public IActionResult Post([FromBody] string name)

        {

            if (string.IsNullOrEmpty(name))

                return BadRequest("Name is required");

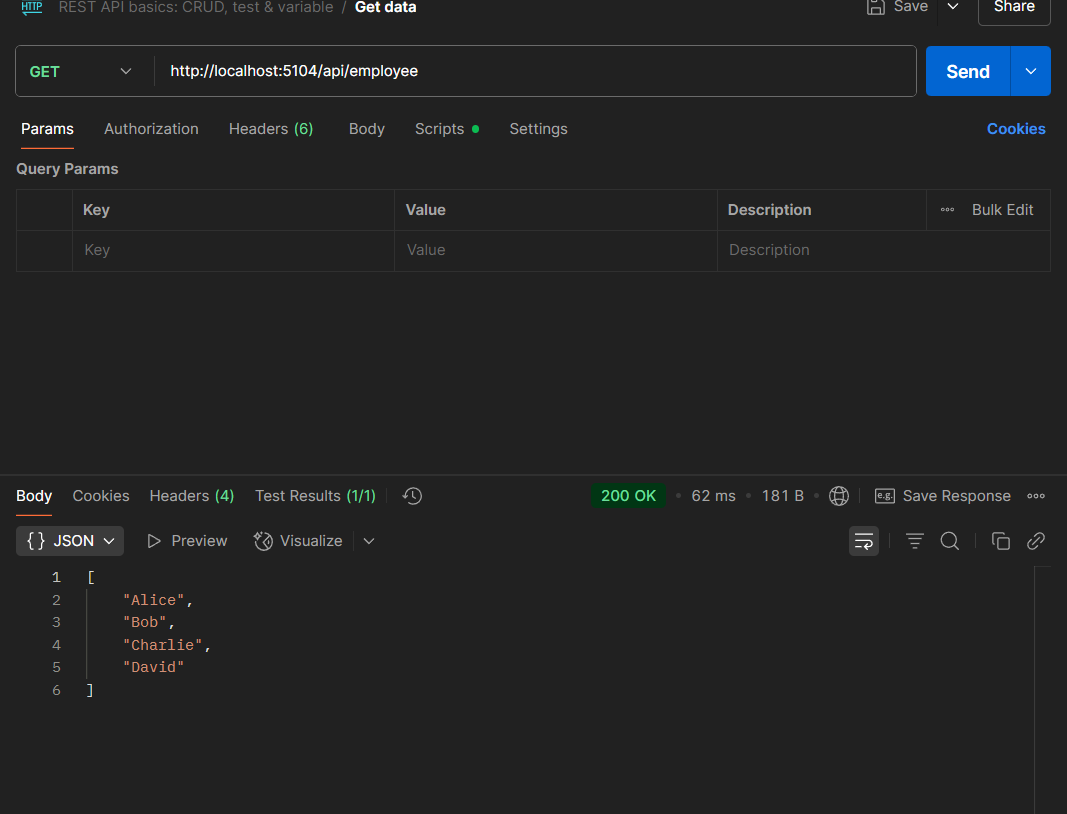
            employees.Add(name);

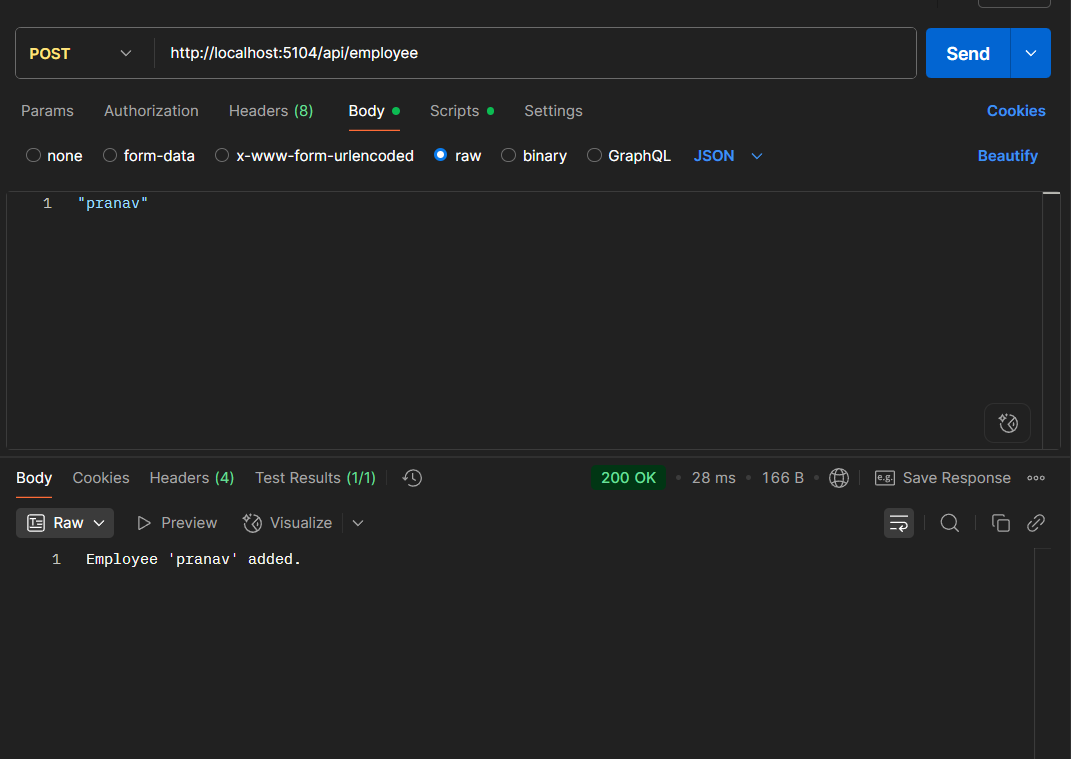
            return Ok($"Employee '{name}' added.");

        }

    }

}







#3 WEBAPI\_HANDSON

PART 1: Create a Custom Model Class

using System;

using System.Collections.Generic;

namespace SwaggerDemoAPI.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; }

    }

}

namespace SwaggerDemoAPI.Models

{

    public class Department

    {

        public int Id { get; set; }

        public string Name { get; set; }

    }

}

namespace SwaggerDemoAPI.Models

{

    public class Skill

    {

        public int Id { get; set; }

        public string Name { get; set; }

    }

}

EmployeeController

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Authorization;

using SwaggerDemoAPI.Models;

using SwaggerDemoAPI.Filters;

namespace SwaggerDemoAPI.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    [ServiceFilter(typeof(CustomAuthFilter))]

    public class EmployeeController : ControllerBase

    {

        private static List<Employee> employees = new();

        public EmployeeController()

        {

            if (!employees.Any())

                employees = GetStandardEmployeeList();

        }

        private List<Employee> GetStandardEmployeeList()

        {

            return new List<Employee>

            {

                new Employee

                {

                    Id = 1,

                    Name = "Alice",

                    Salary = 60000,

                    Permanent = true,

                    DateOfBirth = new DateTime(1991, 4, 12),

                    Department = new Department { Id = 101, Name = "Engineering" },

                    Skills = new List<Skill> { new Skill { Id = 1, Name = "C#" }, new Skill { Id = 2, Name = "SQL" } }

                }

            };

        }

        [HttpGet]

        [ProducesResponseType(StatusCodes.Status200OK)]

        [ProducesResponseType(StatusCodes.Status500InternalServerError)]

        [AllowAnonymous] // Allow unauthenticated access

        public ActionResult<List<Employee>> Get()

        {

            // Simulate exception for testing

            // throw new Exception("Custom exception for testing");

            return Ok(employees);

        }

        [HttpPost]

        [ProducesResponseType(StatusCodes.Status200OK)]

        public ActionResult Add([FromBody] Employee employee)

        {

            employees.Add(employee);

            return Ok("Employee added successfully.");

        }

    }

}

CustomAuthFilter.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace SwaggerDemoAPI.Filters

{

    public class CustomAuthFilter : ActionFilterAttribute

    {

        public override void OnActionExecuting(ActionExecutingContext context)

        {

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

            {

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

                return;

            }

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

            {

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

                return;

            }

            base.OnActionExecuting(context);

        }

    }

}

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System.IO;

namespace SwaggerDemoAPI.Filters

{

    public class CustomExceptionFilter : IExceptionFilter

    {

        public void OnException(ExceptionContext context)

        {

            var message = $"[{DateTime.Now}] {context.Exception.Message}{Environment.NewLine}";

            File.AppendAllText("error\_log.txt", message);

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

            {

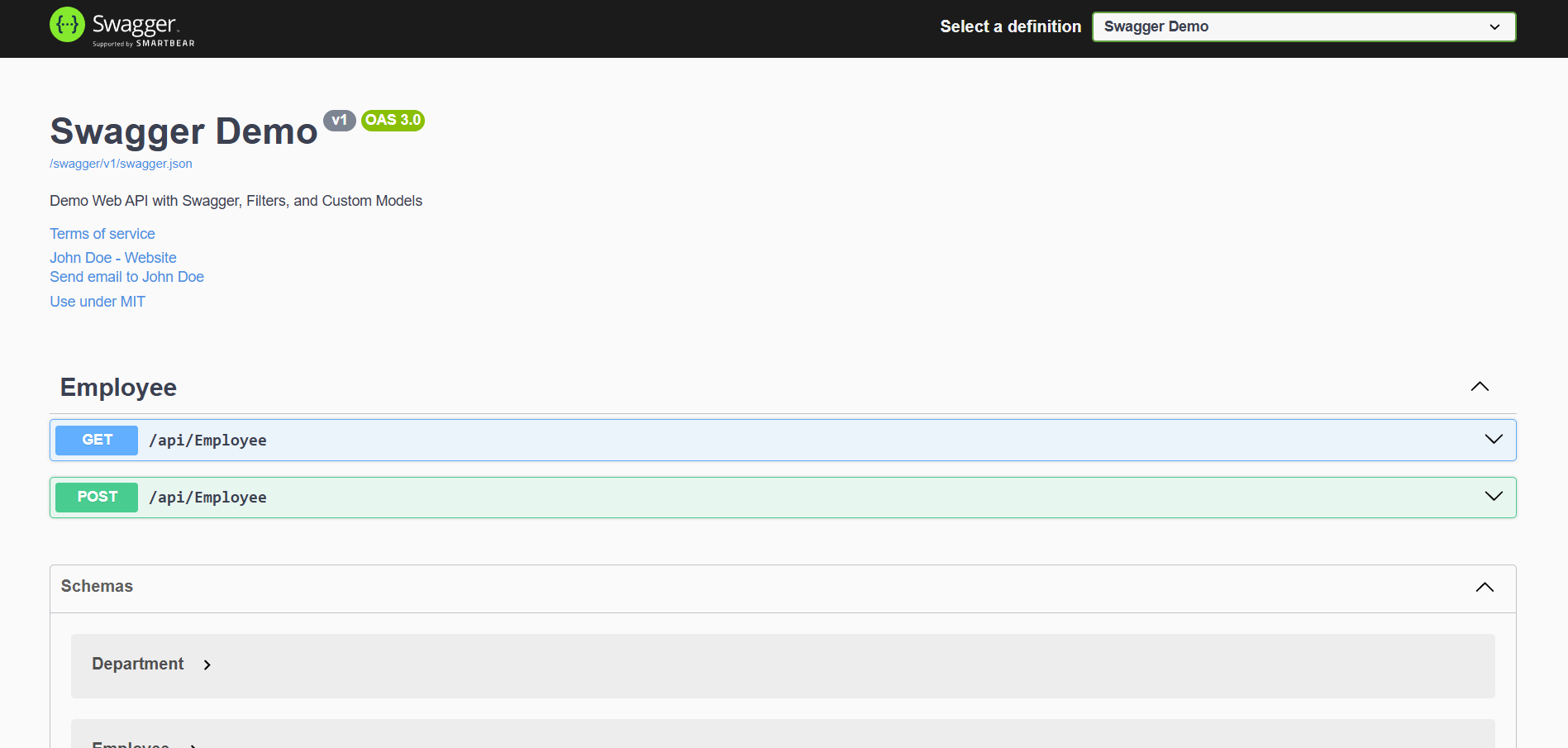
                StatusCode = 500

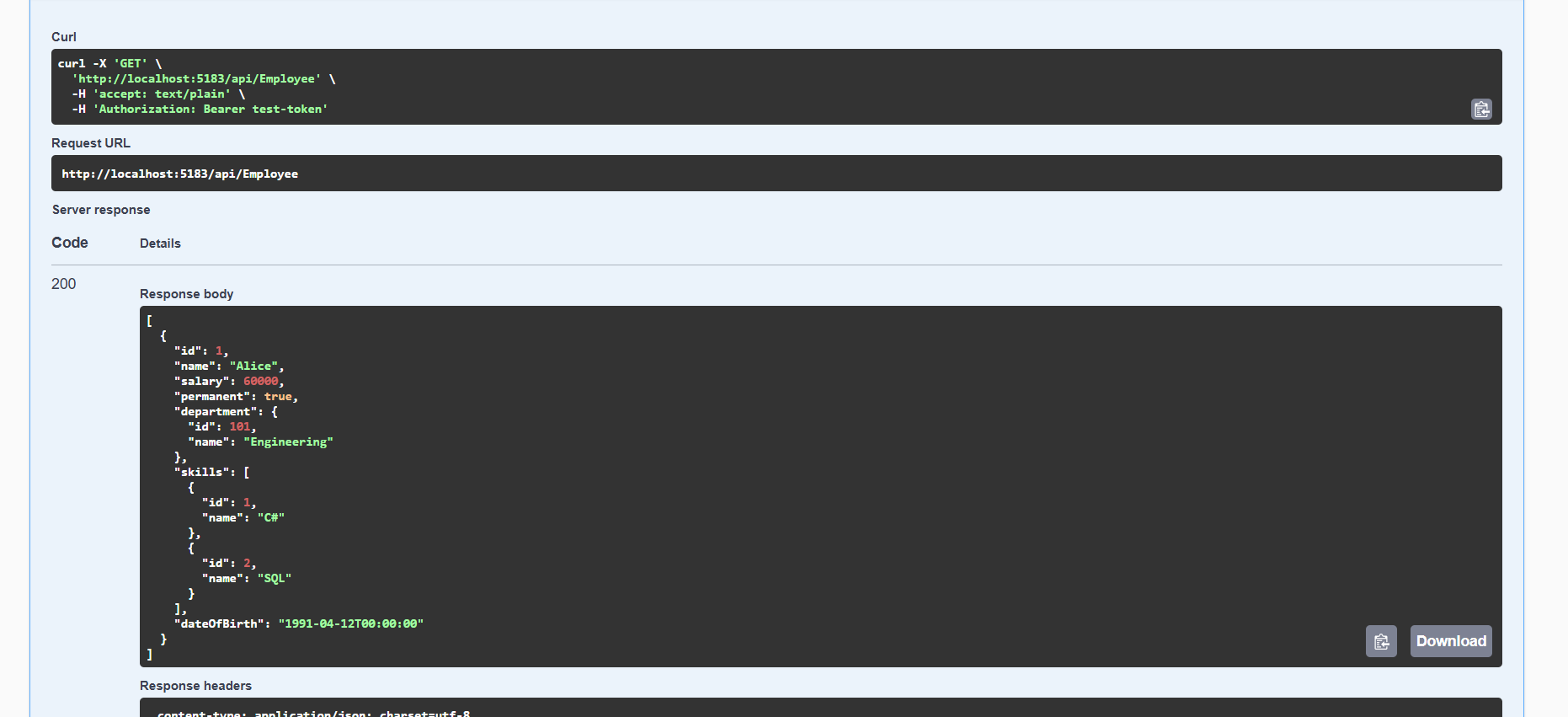
            };

        }

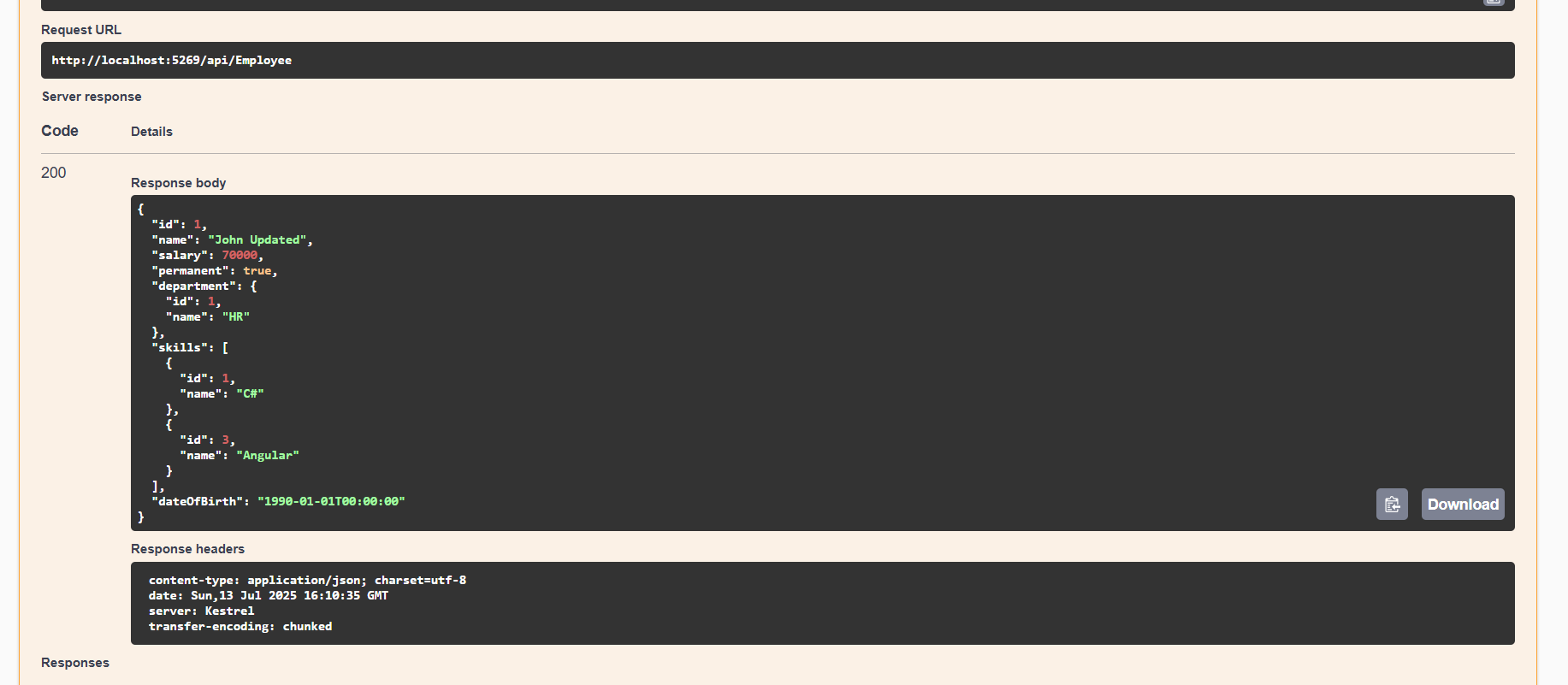
    }

}









#4WEBAPI\_HANDSON

Create Model Classes

namespace EmployeeApiDemo.Models

{

    public class Department

    {

        public int Id { get; set; }

        public string Name { get; set; }

    }

}

namespace EmployeeApiDemo.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; }

    }

}

namespace EmployeeApiDemo.Models

{

    public class Skill

    {

        public int Id { get; set; }

        public string Name { get; set; }

    }

}

Create Controller

using Microsoft.AspNetCore.Mvc;

using EmployeeApiDemo.Models;

namespace EmployeeApiDemo.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    public class EmployeeController : ControllerBase

    {

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

        {

            new Employee

            {

                Id = 1,

                Name = "John Doe",

                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)

            },

            new Employee

            {

                Id = 2,

                Name = "Jane Smith",

                Salary = 60000,

                Permanent = false,

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

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

                DateOfBirth = new DateTime(1992, 5, 15)

            }

        };

        [HttpGet]

        public ActionResult<List<Employee>> Get() => \_employees;

        [HttpPut]

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

        {

            if (input.Id <= 0)

                return BadRequest("Invalid employee id");

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

            if (existingEmp == null)

                return BadRequest("Invalid employee id");

            // Update fields

            existingEmp.Name = input.Name;

            existingEmp.Salary = input.Salary;

            existingEmp.Permanent = input.Permanent;

            existingEmp.Department = input.Department;

            existingEmp.Skills = input.Skills;

            existingEmp.DateOfBirth = input.DateOfBirth;

            return Ok(existingEmp);

        }

        [HttpPost]

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

        {

            emp.Id = \_employees.Max(e => e.Id) + 1;

            \_employees.Add(emp);

            return CreatedAtAction(nameof(Get), new { id = emp.Id }, emp);

        }

        [HttpDelete("{id}")]

        public ActionResult Delete(int id)

        {

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

            if (emp == null)

                return NotFound();

            \_employees.Remove(emp);

            return NoContent();

        }

    }

}

Program.cs

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

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

});

var app = builder.Build();

// Configure the HTTP request pipeline.

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

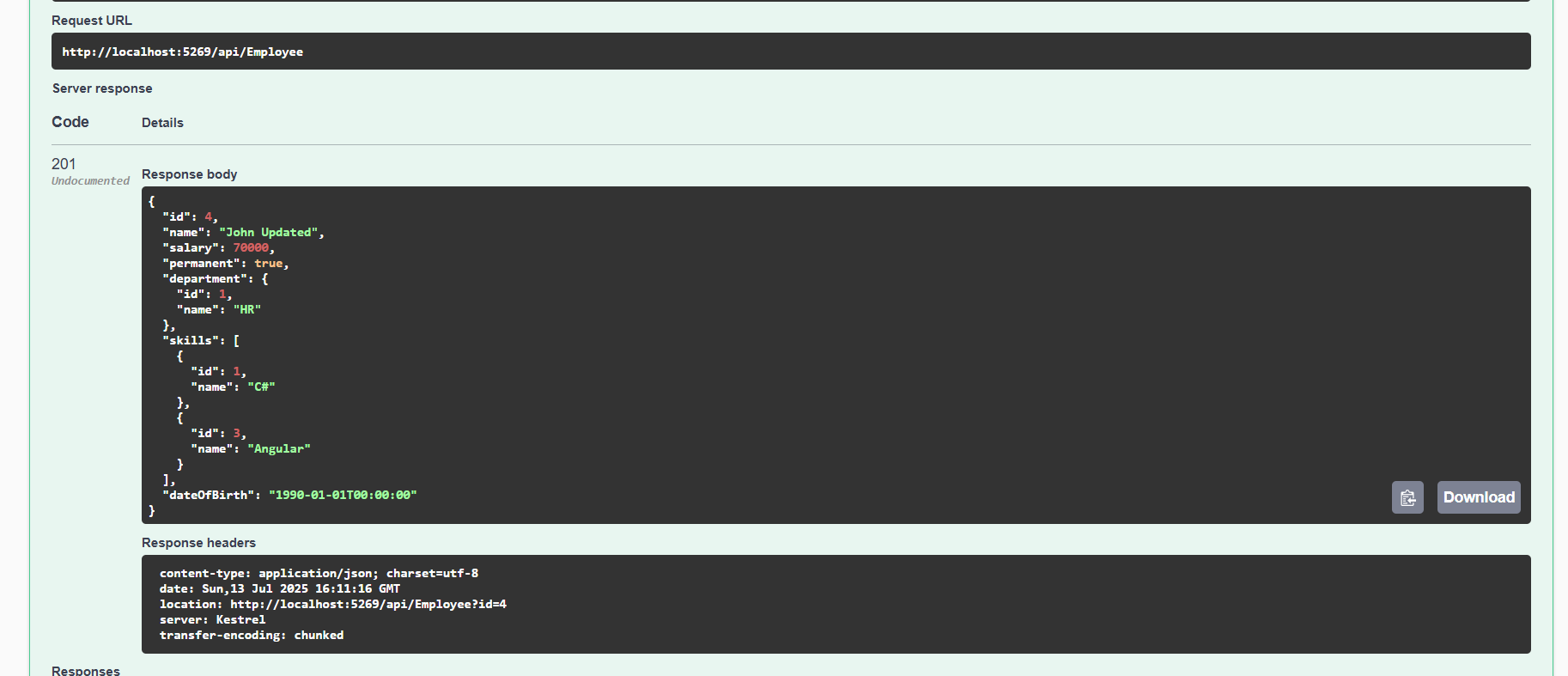
});

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();



#5 WEBAPI HANDSON

Program.cs

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

// Enable CORS

builder.Services.AddCors(options =>

{

    options.AddDefaultPolicy(policy =>

    {

        policy.WithOrigins("http://localhost:4200") // Add frontend URL

              .AllowAnyHeader()

              .AllowAnyMethod();

    });

});

// Add Controllers

builder.Services.AddControllers();

// JWT Authentication setup

var key = "ThisIsA\_very\_long\_secure\_key!1234567890123456";

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

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 = symmetricKey

    };

});

// Add Swagger with JWT support

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

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

    c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

    {

        In = ParameterLocation.Header,

        Description = "Please enter JWT with Bearer prefix",

        Name = "Authorization",

        Type = SecuritySchemeType.ApiKey

    });

    c.AddSecurityRequirement(new OpenApiSecurityRequirement

    {

        {

            new OpenApiSecurityScheme {

                Reference = new OpenApiReference {

                    Type=ReferenceType.SecurityScheme,

                    Id="Bearer"

                }

            },

            Array.Empty<string>()

        }

    });

});

var app = builder.Build();

app.UseSwagger();

app.UseSwaggerUI();

app.UseHttpsRedirection();

app.UseCors(); // CORS Middleware

app.UseAuthentication(); // JWT Middleware

app.UseAuthorization();

app.MapControllers();

app.Run();

AuthController.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using Microsoft.AspNetCore.Authorization;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace EmployeeApiDemo.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    [AllowAnonymous]

    public class AuthController : ControllerBase

    {

        [HttpGet("token")]

        public IActionResult GetToken()

        {

            var token = GenerateJSONWebToken(101, "Admin");

            return Ok(new { token });

        }

        private string GenerateJSONWebToken(int userId, string userRole)

        {

            var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("ThisIsA\_very\_long\_secure\_key!1234567890123456"));

            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);

        }

    }

}

EmployeeController.cs

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using EmployeeApiDemo.Models;

namespace EmployeeApiDemo.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    [Authorize(Roles = "Admin,POC")]

    public class EmployeeController : ControllerBase

    {

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

        {

            new Employee

            {

                Id = 1,

                Name = "John",

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

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

            }

        };

        [HttpGet]

        public IActionResult Get()

        {

            return Ok(\_employees);

        }

    }

}

Get JWT Token

