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

Code

using Microsoft.AspNetCore.Mvc;

namespace MyFirstWebApi.Controllers

{

[ApiController]

[Route("[controller]")]

public class WeatherForecastController : ControllerBase

{

private static readonly string[] Summaries = new[]

{

"Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching"

};

private readonly ILogger<WeatherForecastController> \_logger;

public WeatherForecastController(ILogger<WeatherForecastController> logger)

{

\_logger = logger;

}

[HttpGet(Name = "GetWeatherForecast")]

public IEnumerable<WeatherForecast> Get()

{

return Enumerable.Range(1, 5).Select(index => new WeatherForecast

{

Date = DateOnly.FromDateTime(DateTime.Now.AddDays(index)),

TemperatureC = Random.Shared.Next(-20, 55),

Summary = Summaries[Random.Shared.Next(Summaries.Length)]

})

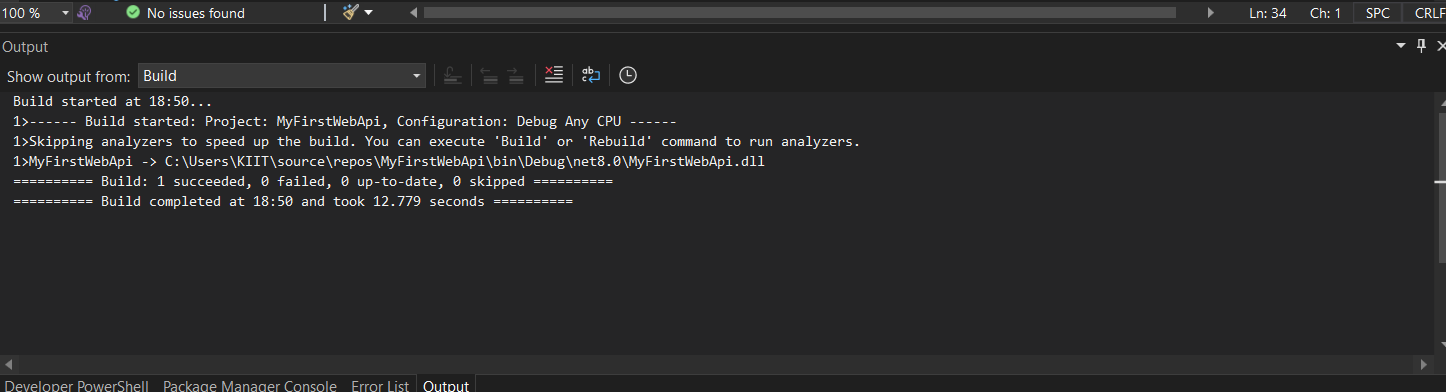
.ToArray();

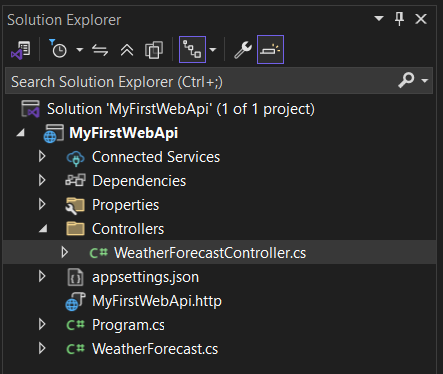
}

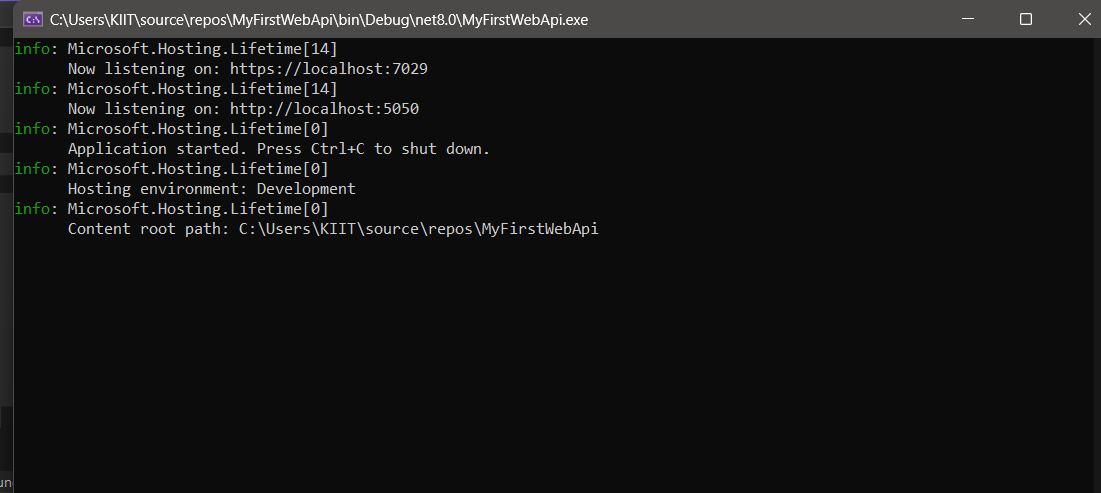
}

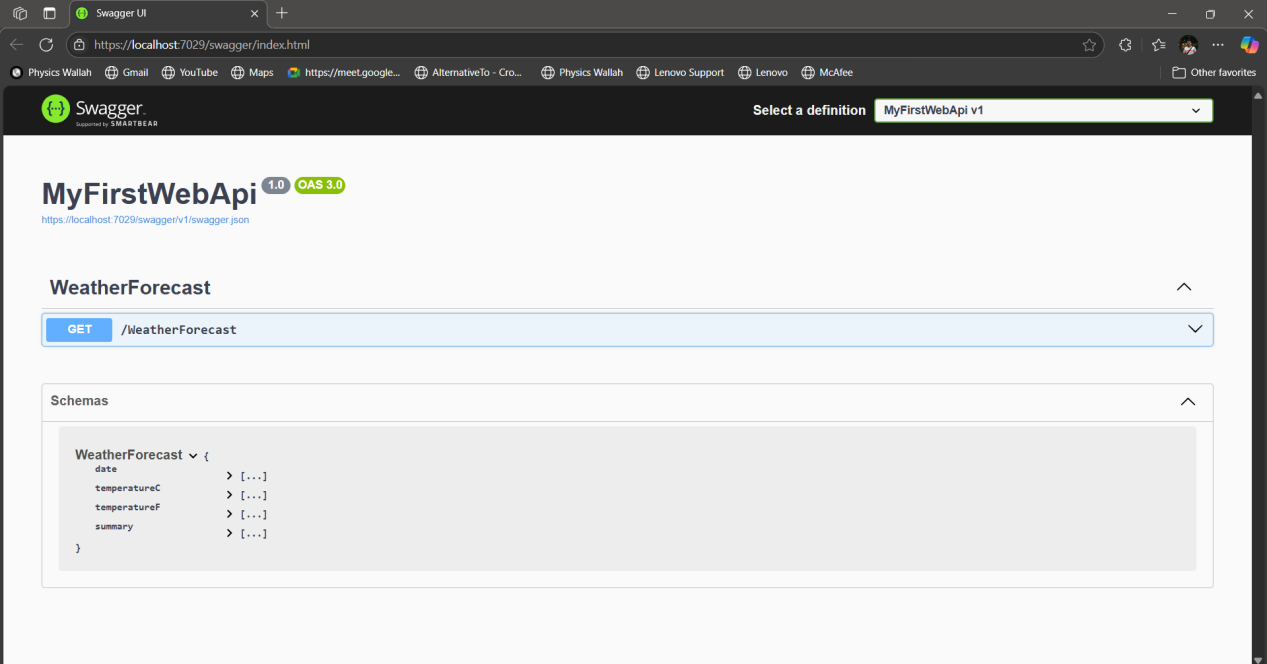
}

OUTPUT









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

**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 = "Bob Builder",

Email = "bob@example.com",

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

},

License = new OpenApiLicense

{

Name = "License Terms",

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

}

});

});

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

}

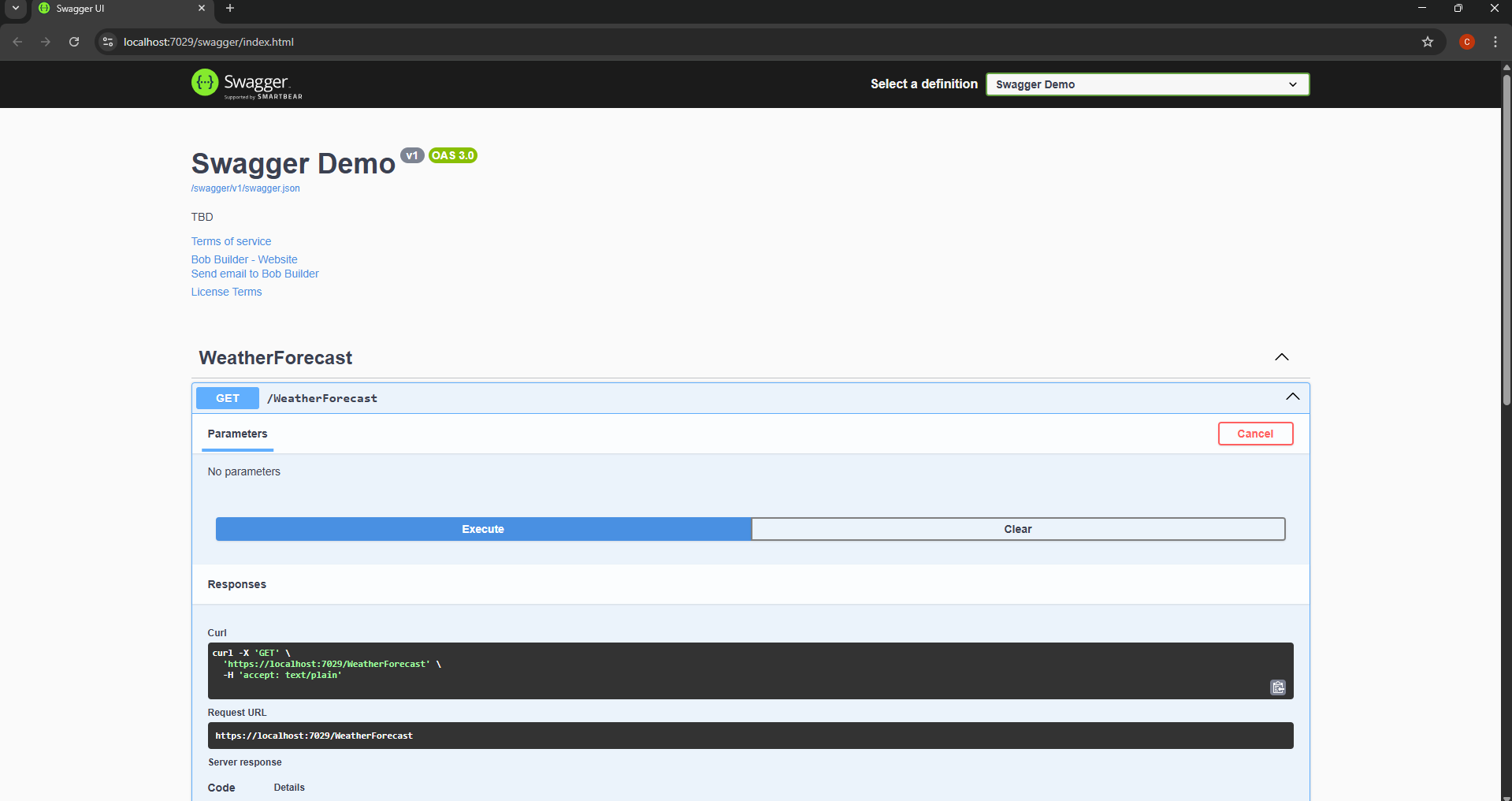
app.UseHttpsRedirection();

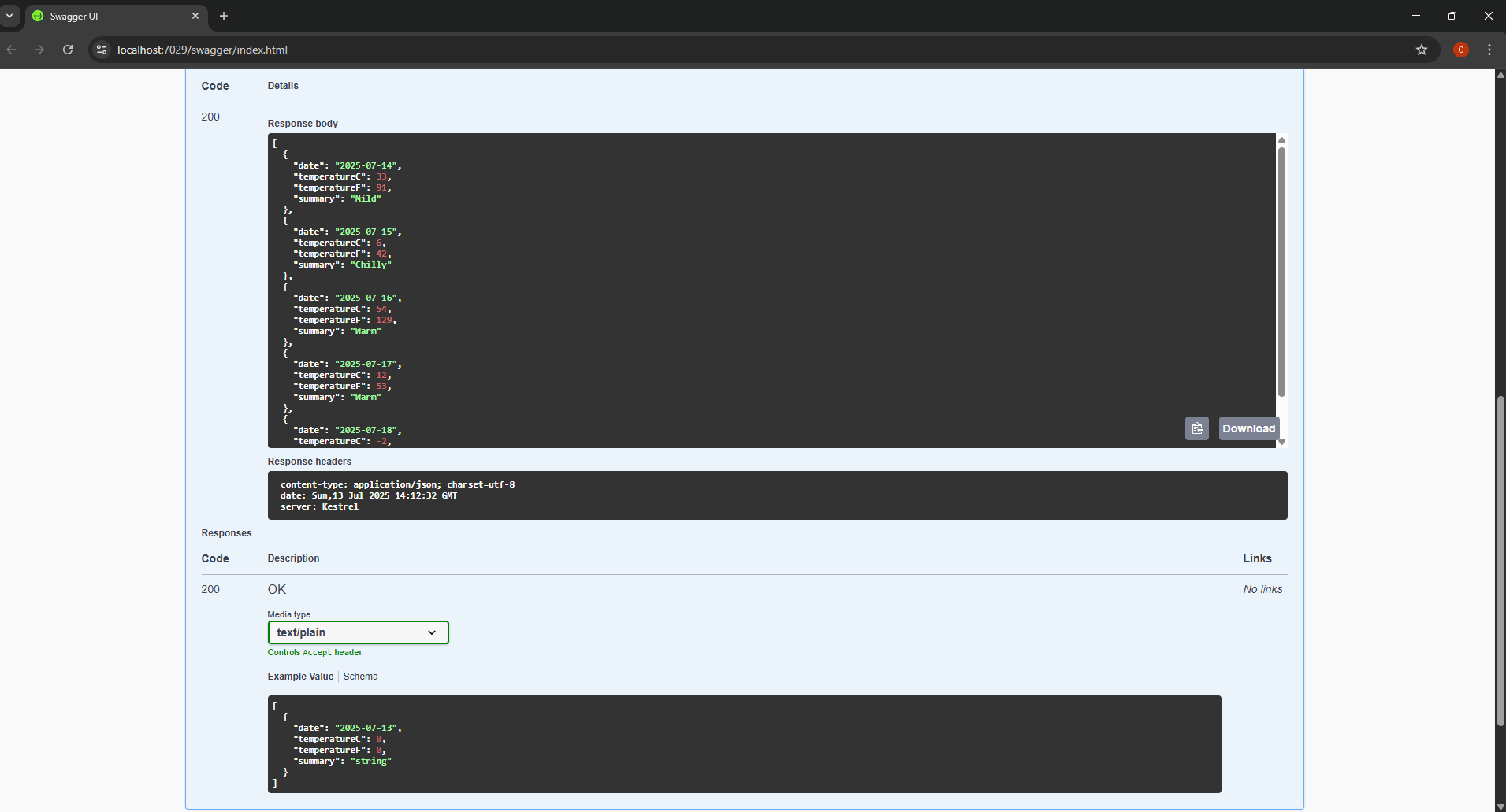
app.UseAuthorization();

app.MapControllers();

app.Run();

**OUTPUT**





Program.cs(Updated)

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 = "Bob Builder",

Email = "bob@example.com",

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

},

License = new OpenApiLicense

{

Name = "License Terms",

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

}

});

});

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

WeatherForecastController.cs

using Microsoft.AspNetCore.Mvc;

namespace MyFirstWebApi.Controllers

{

[ApiController]

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

public class WeatherForecastController : ControllerBase

{

private static readonly string[] Summaries = new[]

{

"Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching"

};

private readonly ILogger<WeatherForecastController> \_logger;

public WeatherForecastController(ILogger<WeatherForecastController> logger)

{

\_logger = logger;

}

[HttpGet]

public IEnumerable<WeatherForecast> Get()

{

return Enumerable.Range(1, 5).Select(index => new WeatherForecast

{

Date = DateOnly.FromDateTime(DateTime.Now.AddDays(index)),

TemperatureC = Random.Shared.Next(-20, 55),

Summary = Summaries[Random.Shared.Next(Summaries.Length)]

})

.ToArray();

}

}

}

WeatherForecast.cs

namespace MyFirstWebApi

{

public class WeatherForecast

{

public DateOnly Date { get; set; }

public int TemperatureC { get; set; }

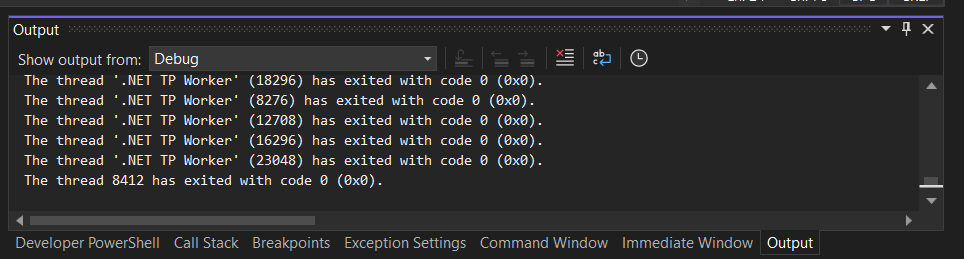
public int TemperatureF => 32 + (int)(TemperatureC / 0.5556);

public string? Summary { get; set; }

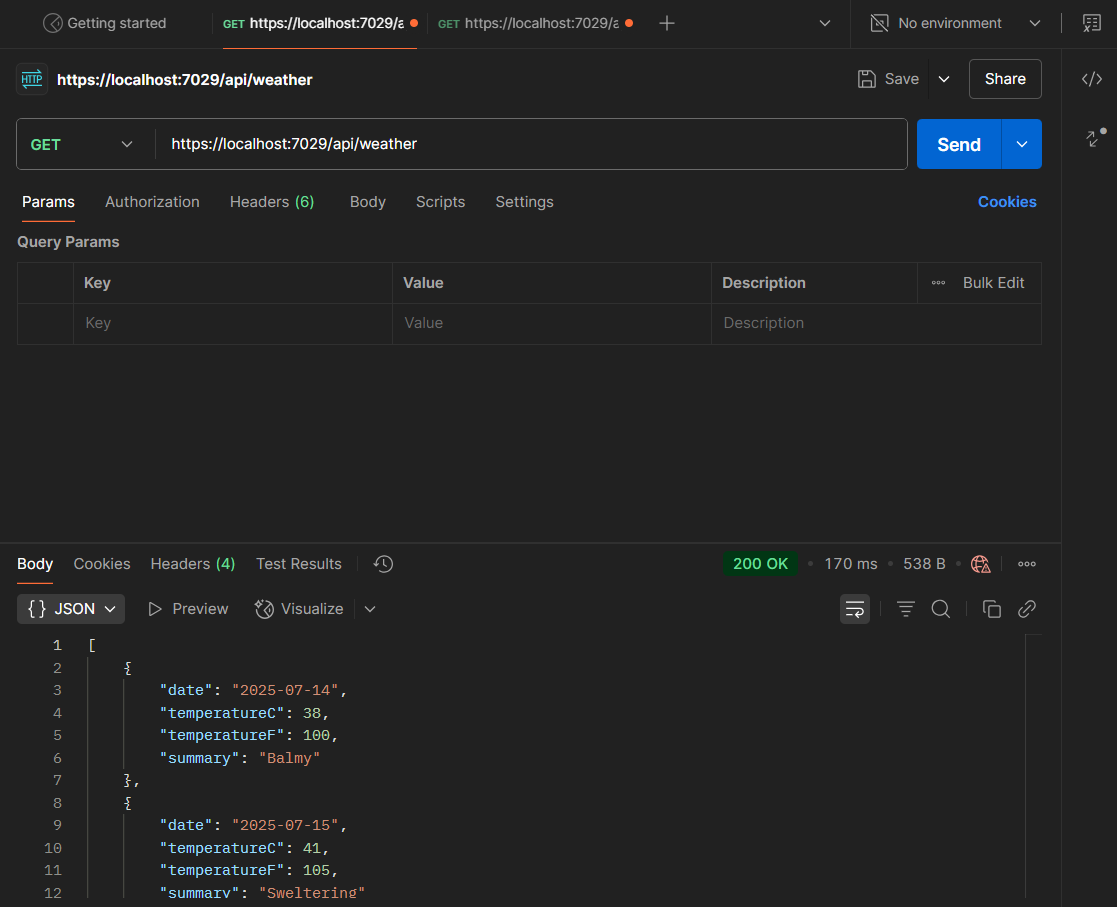
}

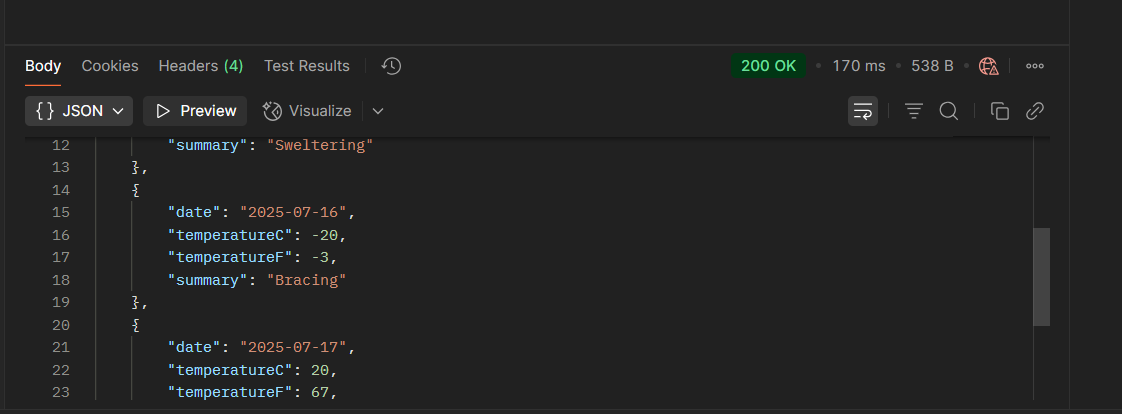
}

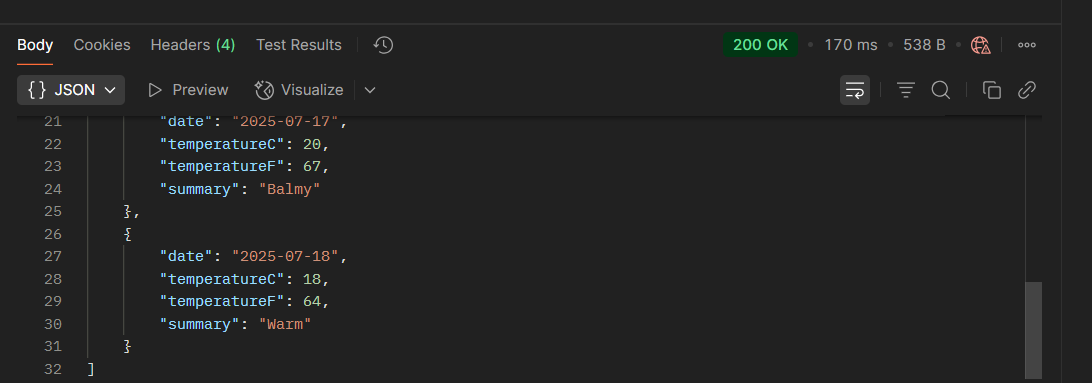
OUTPUT



(Screenshots from Postman App)







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

Employee.cs

using System;

using System.Collections.Generic;

namespace MyFirstWebApi.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; }

}

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; }

}

}

EmployeeController.cs

using Microsoft.AspNetCore.Mvc;

using MyFirstWebApi.Models;

using MyFirstWebApi.Filters;

namespace MyFirstWebApi.Controllers

{

[ApiController]

[Route("[controller]")]

[ServiceFilter(typeof(CustomAuthFilter))] // Custom Authorization Filter

public class EmployeeController : ControllerBase

{

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

public EmployeeController()

{

if (!employees.Any())

{

employees = GetStandardEmployeeList();

}

}

[HttpGet]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)] // <== For exception

public ActionResult<List<Employee>> GetStandard()

{

// Simulate an exception to trigger the CustomExceptionFilter

throw new Exception("This is a test exception to trigger the custom exception filter.");

}

[HttpPost]

[ProducesResponseType(StatusCodes.Status201Created)]

public ActionResult<Employee> AddEmployee([FromBody] Employee employee)

{

employees.Add(employee);

return CreatedAtAction(nameof(GetStandard), new { id = employee.Id }, employee);

}

private List<Employee> GetStandardEmployeeList()

{

return 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 = "Communication" },

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

},

DateOfBirth = new DateTime(1990, 5, 15)

},

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 = 3, Name = "Accounting" },

new Skill { Id = 4, Name = "Taxation" }

},

DateOfBirth = new DateTime(1992, 3, 20)

}

};

}

}

}

CustomAuthFilter.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System.Linq;

namespace MyFirstWebApi.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);

}

}

}

HomeController.cs

using Microsoft.AspNetCore.Mvc;

namespace MyFirstWebApi.Controllers

{

public class HomeController : Controller

{

public IActionResult Index()

{

return View();

}

}

}

WeatherForecastController.cs

using Microsoft.AspNetCore.Mvc;

namespace MyFirstWebApi.Controllers

{

[ApiController]

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

public class WeatherForecastController : ControllerBase

{

private static readonly string[] Summaries = new[]

{

"Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching"

};

private readonly ILogger<WeatherForecastController> \_logger;

public WeatherForecastController(ILogger<WeatherForecastController> logger)

{

\_logger = logger;

}

[HttpGet]

public IEnumerable<WeatherForecast> Get()

{

return Enumerable.Range(1, 5).Select(index => new WeatherForecast

{

Date = DateOnly.FromDateTime(DateTime.Now.AddDays(index)),

TemperatureC = Random.Shared.Next(-20, 55),

Summary = Summaries[Random.Shared.Next(Summaries.Length)]

})

.ToArray();

}

}

}

Program.cs

using Microsoft.AspNetCore.Mvc;

using MyFirstWebApi.Filters;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container

builder.Services.AddControllers();

// Register Custom Filters

builder.Services.AddScoped<CustomAuthFilter>();

builder.Services.AddScoped<CustomExceptionFilter>();

// Add Swagger

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Use Swagger

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

// Add exception filter globally

app.Use(async (context, next) =>

{

try

{

await next.Invoke();

}

catch (Exception ex)

{

var logger = context.RequestServices.GetRequiredService<ILogger<Program>>();

logger.LogError(ex, "Unhandled exception occurred.");

context.Response.StatusCode = 500;

await context.Response.WriteAsync("An unexpected error occurred.");

}

});

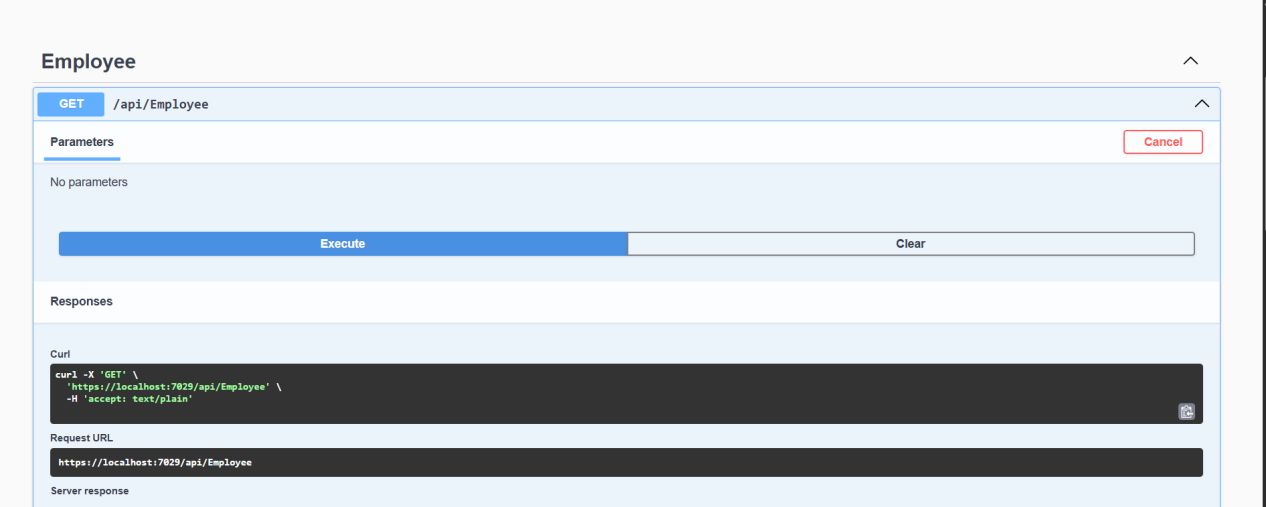
app.UseHttpsRedirection();

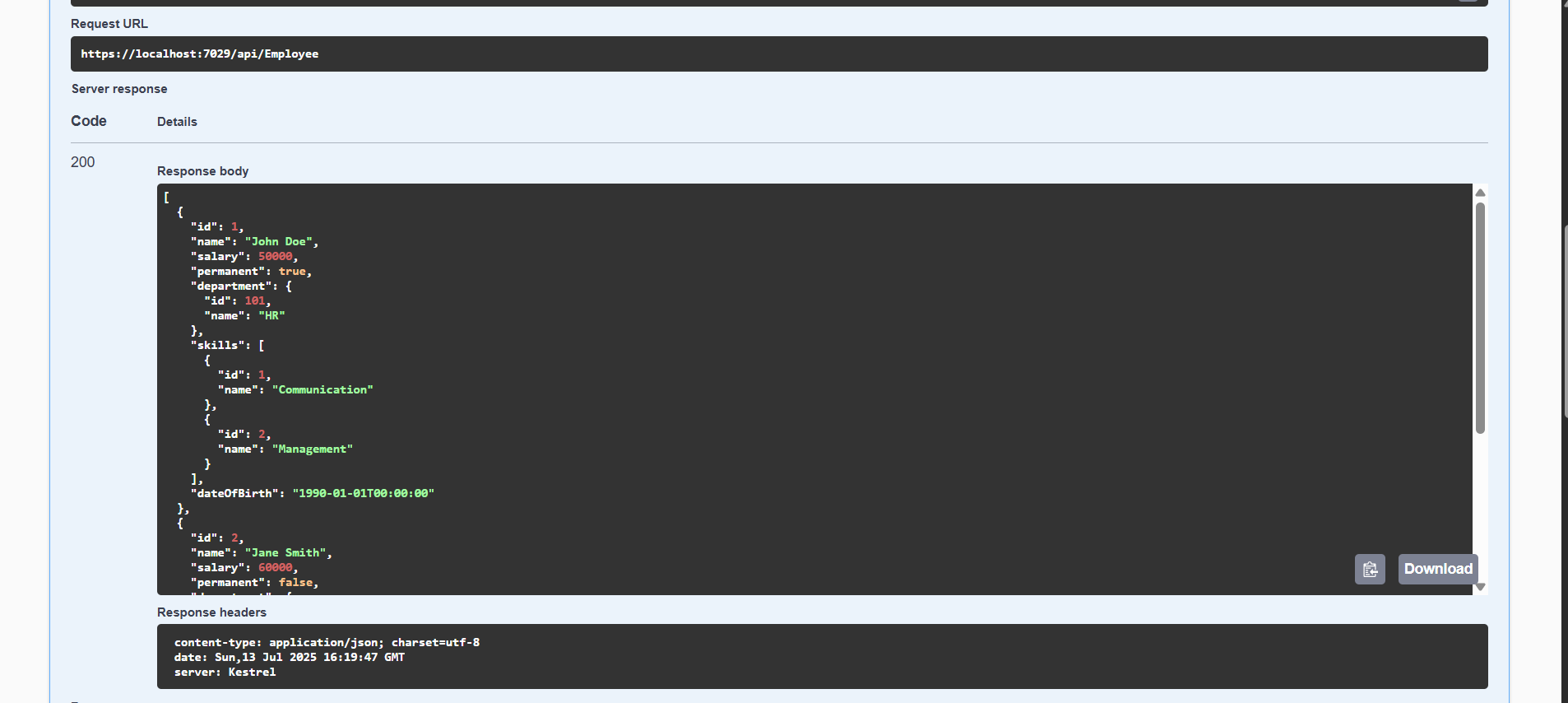
app.UseAuthorization();

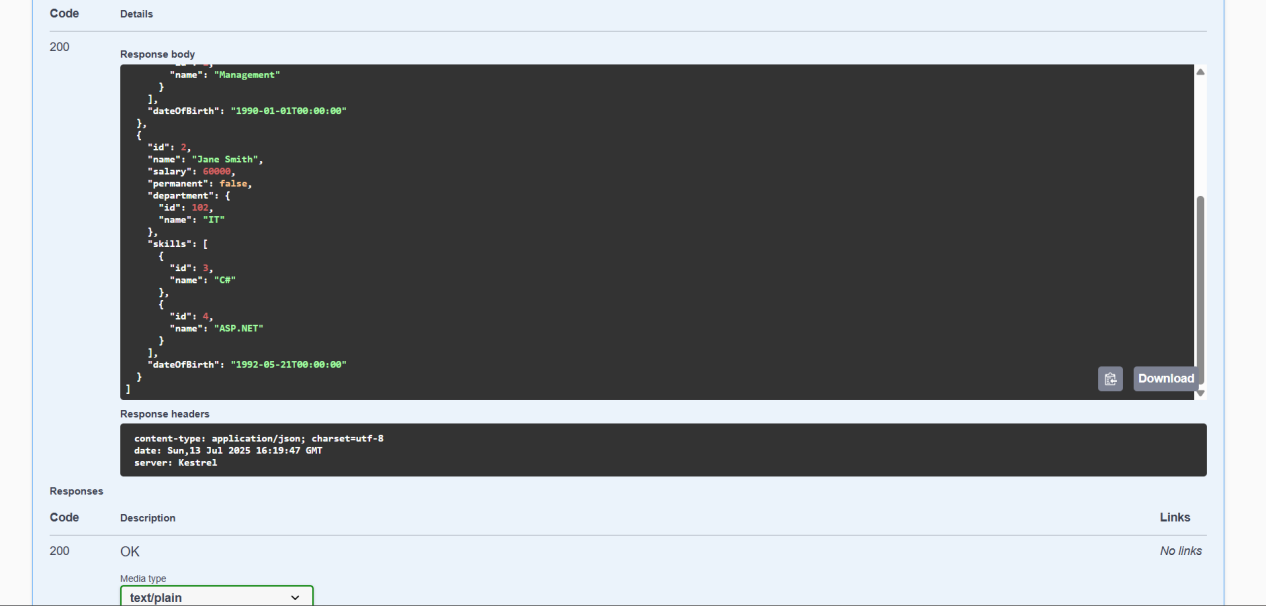
app.MapControllers();

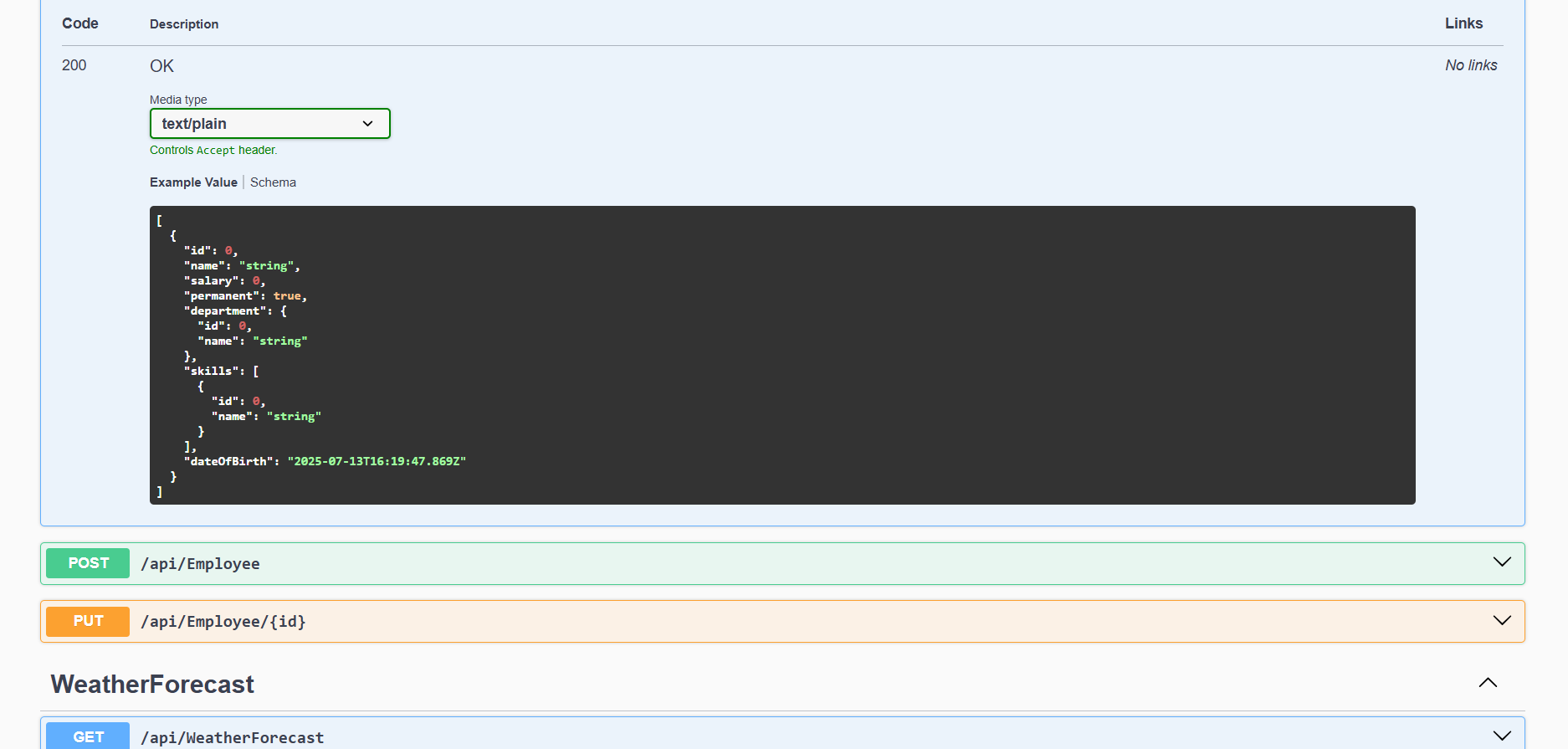
app.Run();

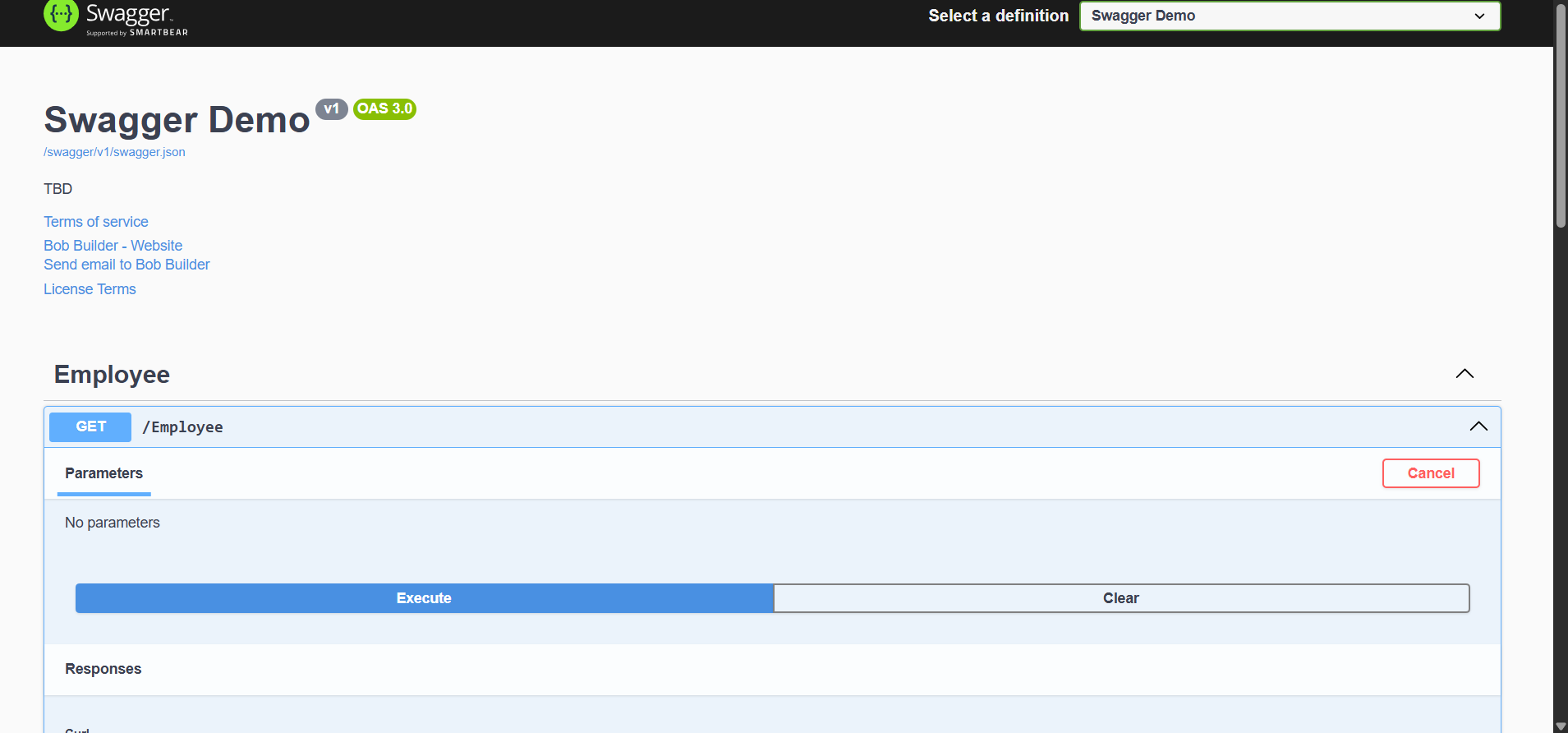
OUTPUT

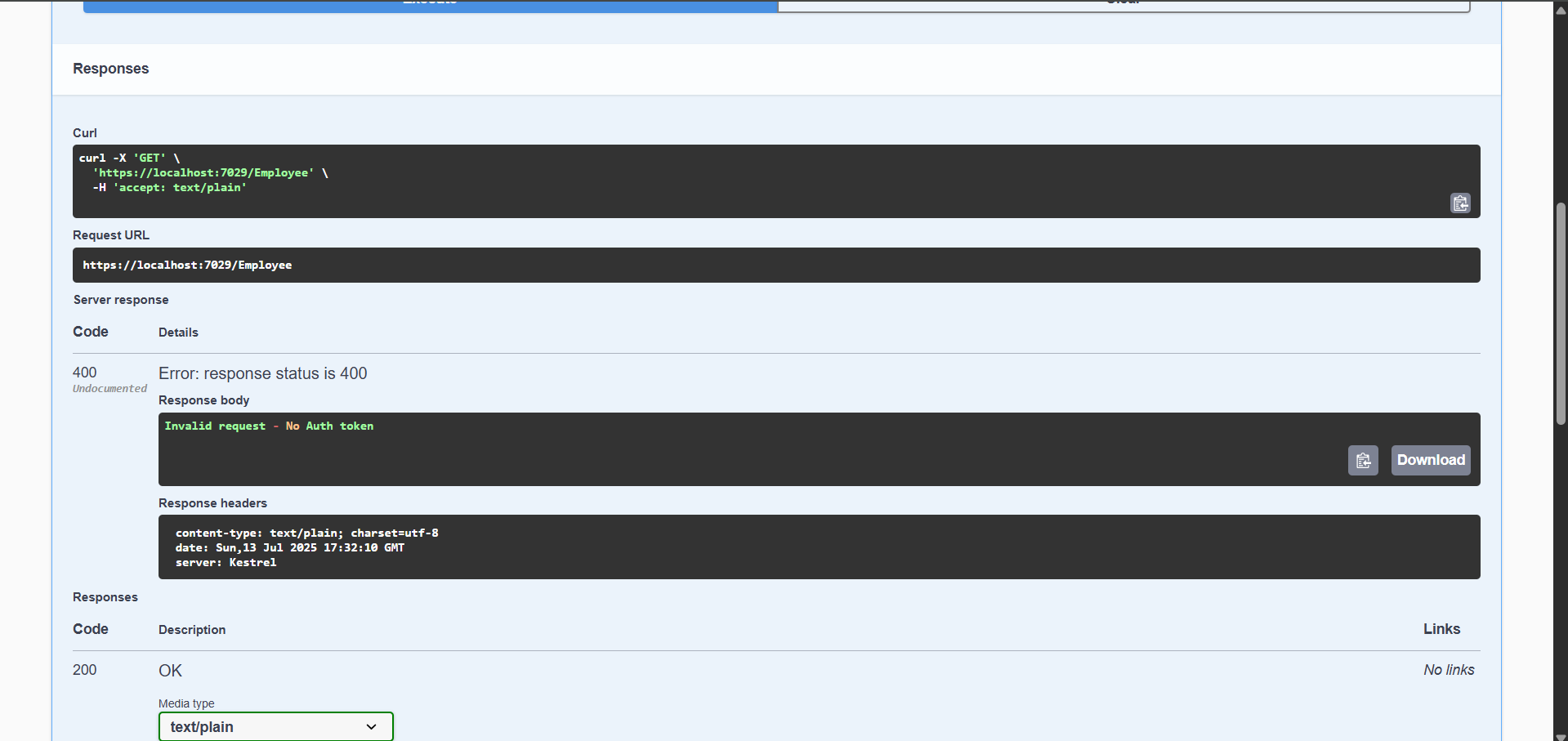


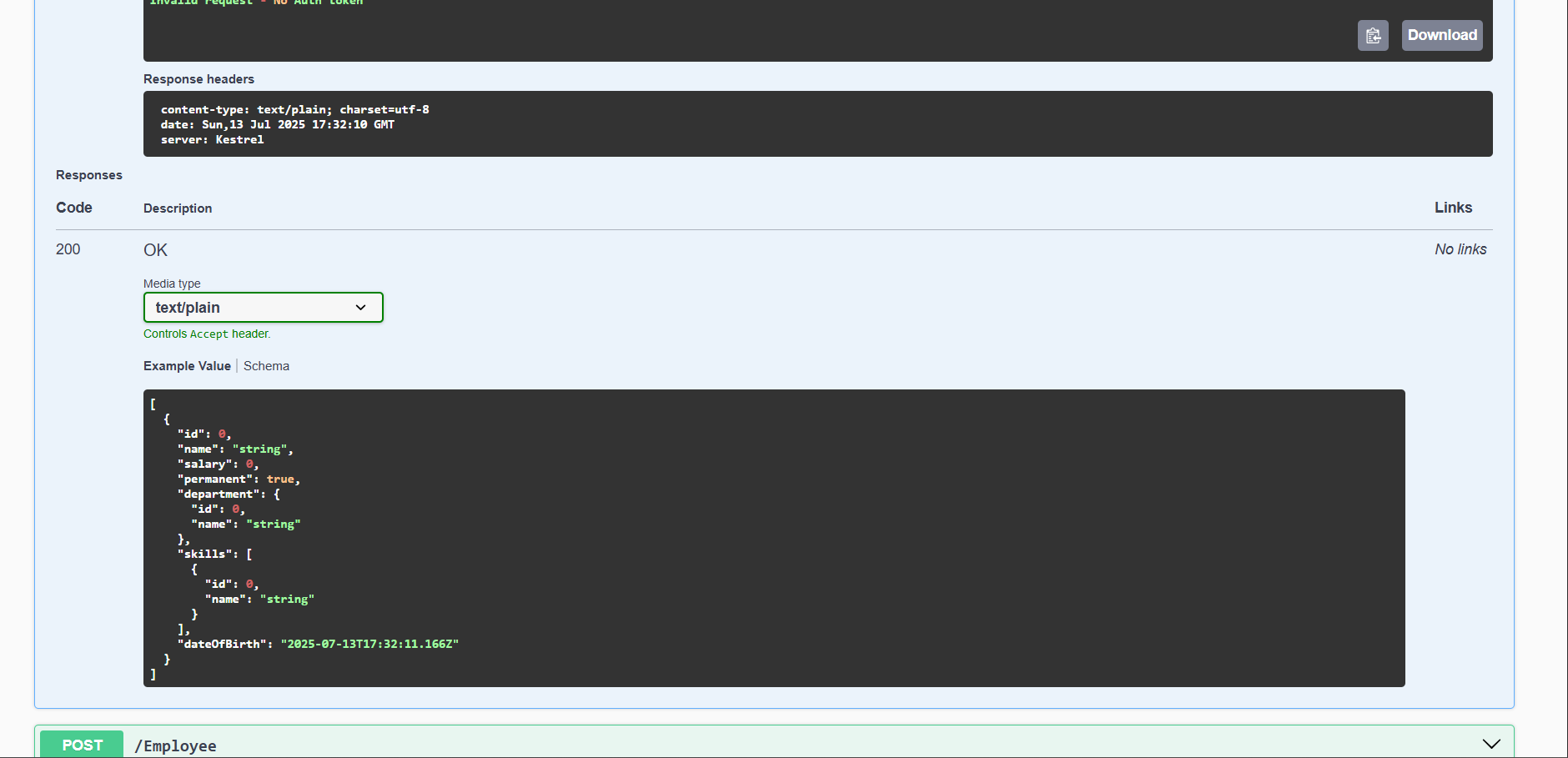


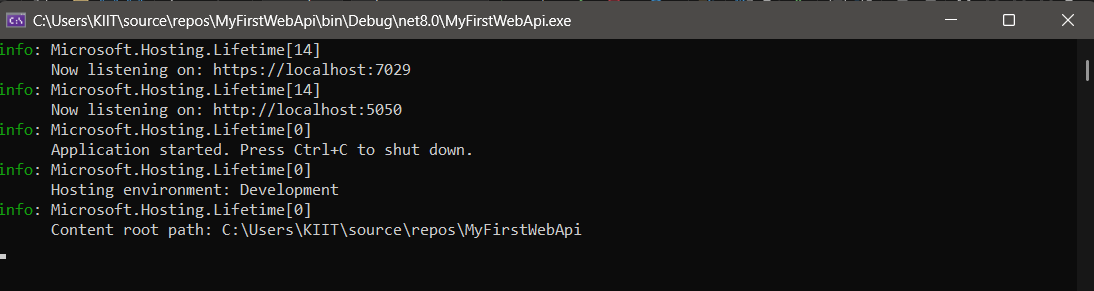












**4.Web Api CRUD operation**

EmployeeController.cs

using Microsoft.AspNetCore.Mvc;

using MyFirstWebApi.Models;

using MyFirstWebApi.Filters;

namespace MyFirstWebApi.Controllers

{

[ApiController]

[Route("[controller]")]

[ServiceFilter(typeof(CustomAuthFilter))]

public class EmployeeController : ControllerBase

{

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

public EmployeeController()

{

if (!employees.Any())

{

employees = GetStandardEmployeeList();

}

}

[HttpGet]

[ProducesResponseType(StatusCodes.Status200OK)]

public ActionResult<List<Employee>> GetStandard()

{

return Ok(employees);

}

[HttpPost]

[ProducesResponseType(StatusCodes.Status201Created)]

public ActionResult<Employee> AddEmployee([FromBody] Employee employee)

{

employees.Add(employee);

return CreatedAtAction(nameof(GetStandard), new { id = employee.Id }, employee);

}

[HttpPut]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

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

{

if (updatedEmp.Id <= 0)

{

return BadRequest("Invalid employee id");

}

var existingEmp = employees.FirstOrDefault(e => e.Id == updatedEmp.Id);

if (existingEmp == null)

{

return BadRequest("Invalid employee id");

}

existingEmp.Name = updatedEmp.Name;

existingEmp.Salary = updatedEmp.Salary;

existingEmp.Permanent = updatedEmp.Permanent;

existingEmp.Department = updatedEmp.Department;

existingEmp.Skills = updatedEmp.Skills;

existingEmp.DateOfBirth = updatedEmp.DateOfBirth;

return Ok(existingEmp);

}

private List<Employee> GetStandardEmployeeList()

{

return 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 = "Communication" },

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

},

DateOfBirth = new DateTime(1990, 5, 15)

},

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 = 3, Name = "Accounting" },

new Skill { Id = 4, Name = "Taxation" }

},

DateOfBirth = new DateTime(1992, 3, 20)

}

};

}

}

}

Program.cs

using Microsoft.OpenApi.Models;

using MyFirstWebApi.Filters;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container

builder.Services.AddControllers();

// Register the custom authorization filter

builder.Services.AddScoped<CustomAuthFilter>();

// Add Swagger generation

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "Web API with Custom Filters",

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

Contact = new OpenApiContact

{

Name = "Bob Builder",

Email = "bob@example.com",

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

},

License = new OpenApiLicense

{

Name = "Use under LICX",

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

}

});

});

var app = builder.Build();

// Configure middleware pipeline

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

Employee.cs

using System;

using System.Collections.Generic;

namespace MyFirstWebApi.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; }

}

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; }

}

}

CustomAuthFilter.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System.Linq;

namespace MyFirstWebApi.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;

}

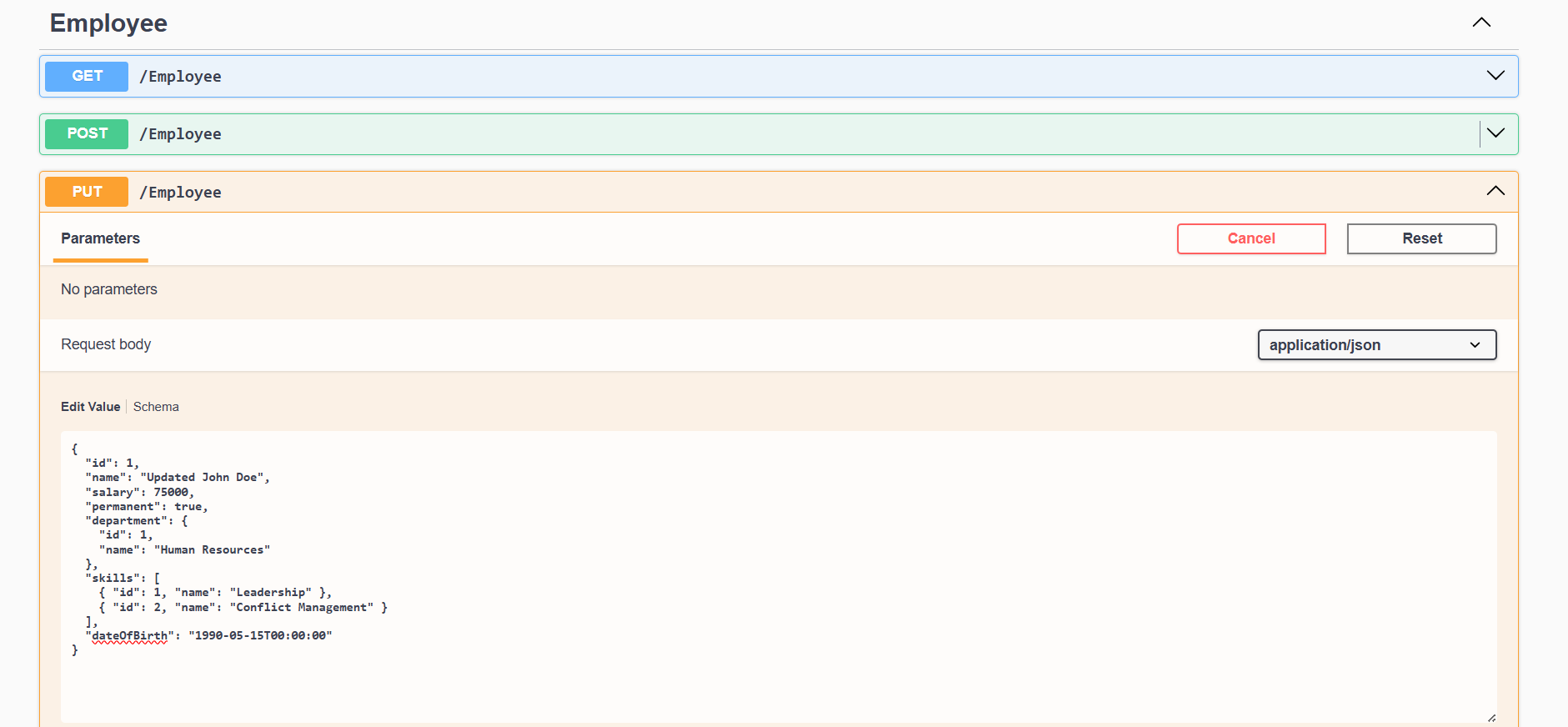
base.OnActionExecuting(context);

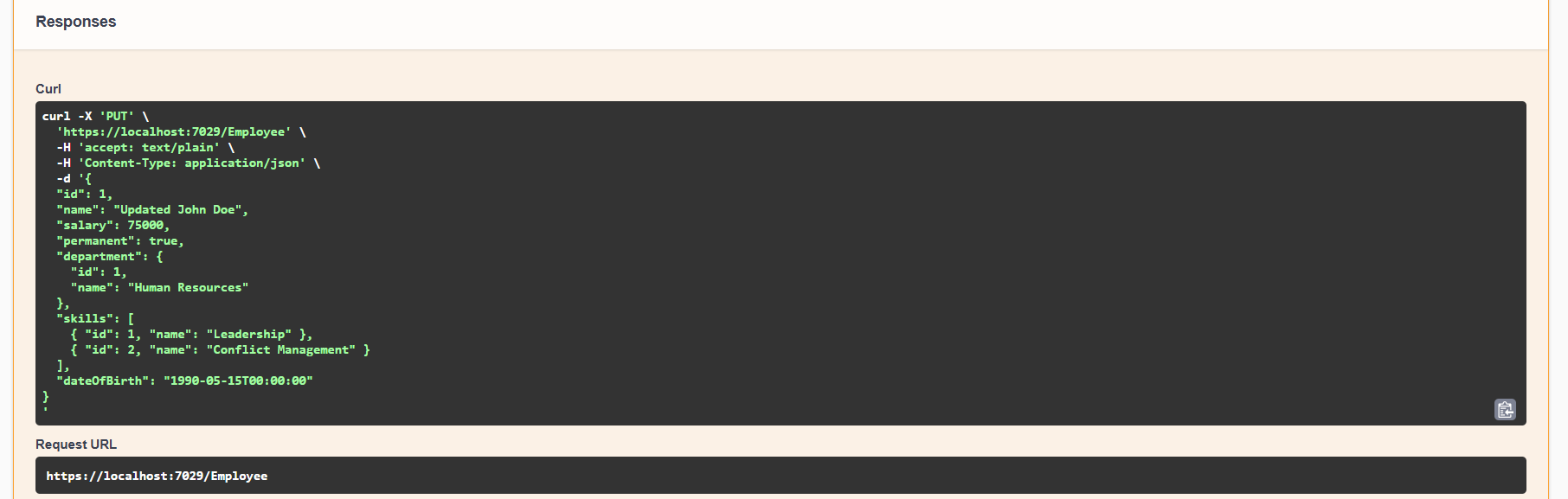
}

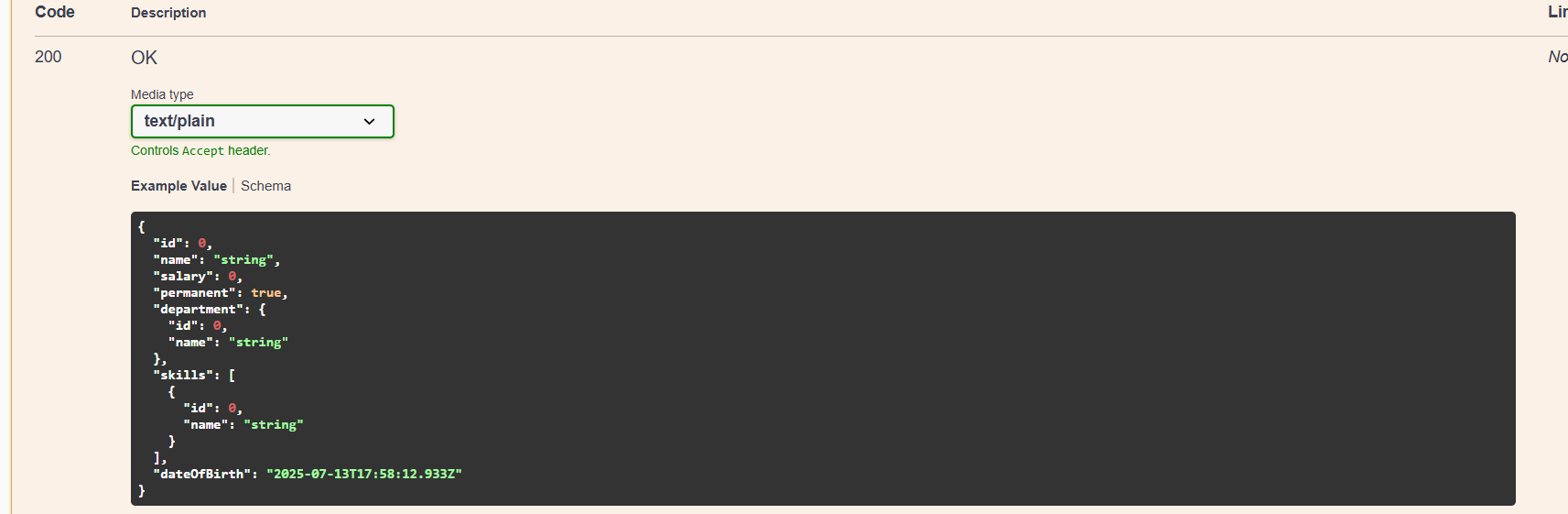
}

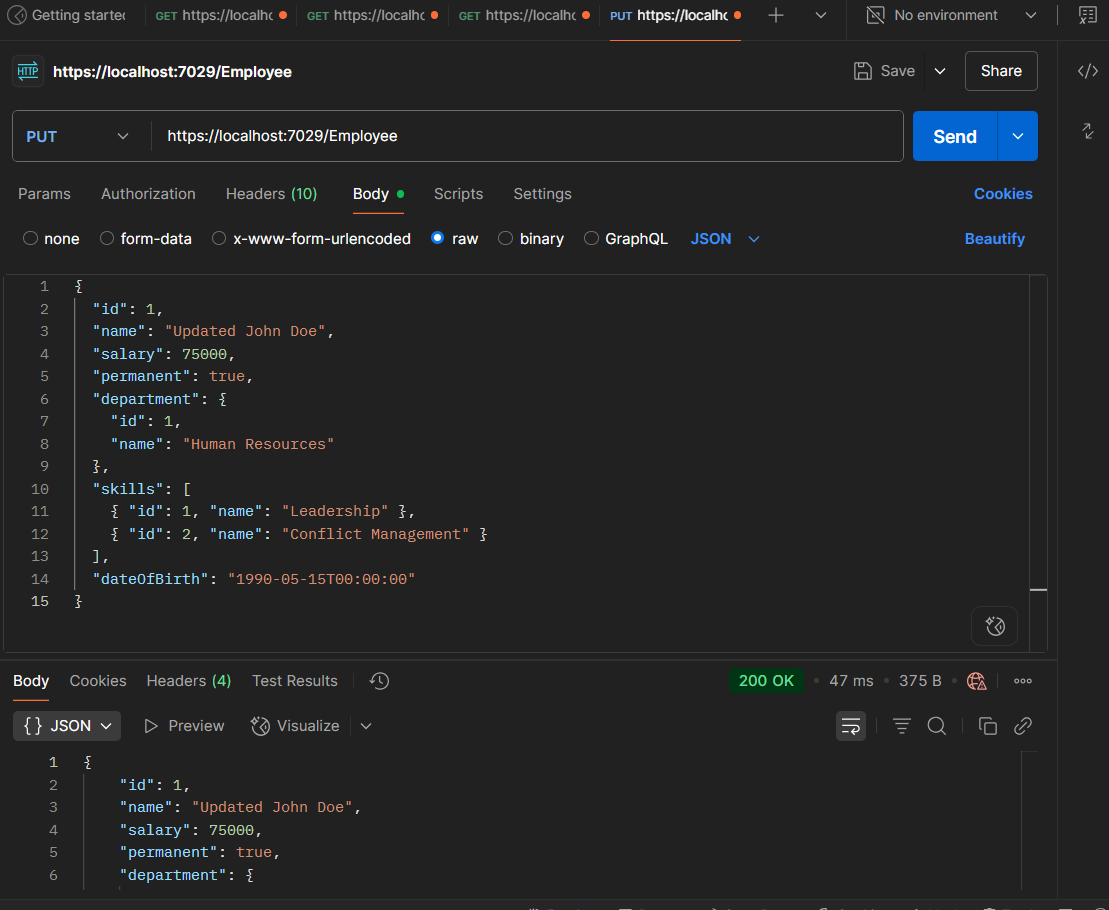
}

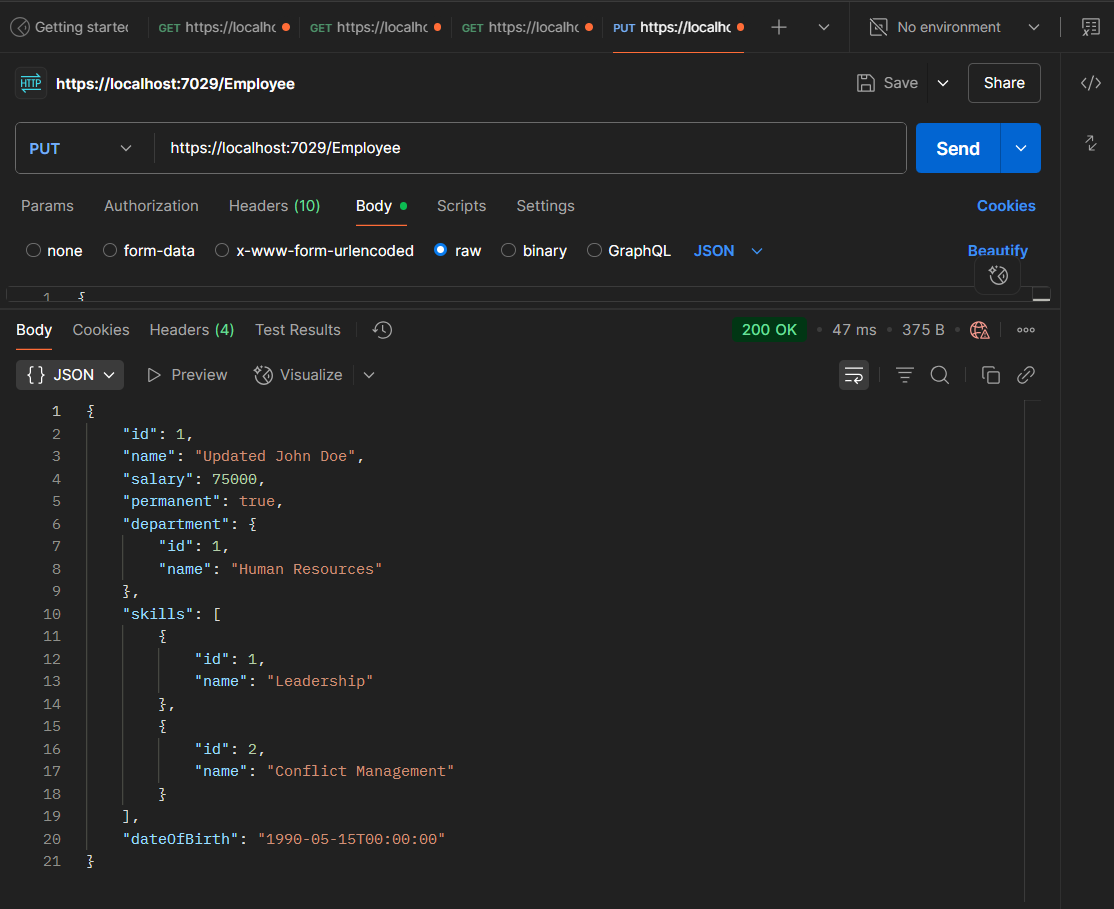
OUTPUT











**5.JsonWebToken**

Program.cs

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

using Microsoft.OpenApi.Models;

using MyFirstWebApi.Filters;

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 = "Swagger Demo",

Version = "v1",

Description = "JWT Auth API",

Contact = new OpenApiContact { Name = "Bob Builder", Email = "bob@example.com", Url = new Uri("https://example.com") },

License = new OpenApiLicense { Name = "License Terms", Url = new Uri("https://example.com") }

});

});

// JWT Authentication configuration

string securityKey = "mysuperdupersecret";

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

builder.Services.AddAuthentication(x =>

{

x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

x.DefaultSignInScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(JwtBearerDefaults.AuthenticationScheme, x =>

{

x.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = "mySystem",

ValidAudience = "myUsers",

IssuerSigningKey = symmetricSecurityKey

};

});

// Register any custom filters if needed

builder.Services.AddScoped<CustomExceptionFilter>();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

// Enable Authentication and Authorization

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

EmployeeController.cs

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using MyFirstWebApi.Models;

namespace MyFirstWebApi.Controllers

{

[ApiController]

[Route("[controller]")]

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

public class EmployeeController : ControllerBase

{

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

public EmployeeController()

{

if (!employees.Any())

{

employees = GetStandardEmployeeList();

}

}

[HttpGet]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public ActionResult<List<Employee>> GetStandard()

{

try

{

return Ok(employees);

}

catch (Exception ex)

{

return StatusCode(500, $"Internal server error: {ex.Message}");

}

}

private List<Employee> GetStandardEmployeeList()

{

return 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 = "Communication" },

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

},

DateOfBirth = new DateTime(1990, 5, 15)

}

};

}

}

}

AuthController.cs

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace MyFirstWebApi.Controllers

{

[AllowAnonymous]

[ApiController]

[Route("[controller]")]

public class AuthController : ControllerBase

{

private string GenerateJSONWebToken(int userId, string userRole)

{

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretkey1234567890123456"));

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.UtcNow.AddMinutes(5),

signingCredentials: credentials

);

return new JwtSecurityTokenHandler().WriteToken(token);

}

[HttpGet]

public IActionResult GetToken()

{

try

{

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

return Ok(new { token });

}

catch (Exception ex)

{

return StatusCode(500, $"Token generation failed: {ex.Message}");

}

}

}

}

CustomAuthFilter.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System.Linq;

namespace MyFirstWebApi.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);

}

}

}

Employee.cs

using System;

using System.Collections.Generic;

namespace MyFirstWebApi.Models

{

public class Employee

{

public int Id { get; set; }

public string Name { get; set; } = string.Empty;

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; }

}

}

Department.cs

namespace MyFirstWebApi.Models

{

public class Department

{

public int Id { get; set; }

public string Name { get; set; } = string.Empty;

}

}

Skill.cs

namespace MyFirstWebApi.Models

{

public class Skill

{

public int Id { get; set; }

public string Name { get; set; } = string.Empty;

}

}

OUTPUT

