**3. WebApi\_Handson**

**EmployeeController.cs**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using YourNamespace.Filters;

using YourNamespace.Models;

namespace YourNamespace.Controllers

{

[ApiController]

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

[AllowAnonymous]

[ServiceFilter(typeof(CustomAuthFilter))]

public class EmployeeController : ControllerBase

{

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "Sumiran",

Salary = 50000,

Permanent = true,

Department = new Department { DeptId = 1, DeptName = "HR" },

Skills = new List<Skill>

{

new Skill { SkillId = 1, SkillName = "C#" },

new Skill { SkillId = 2, SkillName = "SQL" }

},

DateOfBirth = new DateTime(1998, 8, 15)

}

};

}

[HttpGet]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public ActionResult<List<Employee>> GetStandard()

{

return Ok(GetStandardEmployeeList());

}

[HttpPost]

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

{

employee.Id = 99;

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

}

[HttpPut]

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

{

return Ok($"Employee with ID {employee.Id} updated.");

}

}

}

**CustomAuthFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace YourNamespace.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

var headers = context.HttpContext.Request.Headers;

if (!headers.ContainsKey("Authorization"))

{

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

return;

}

var token = headers["Authorization"].ToString();

if (!token.StartsWith("Bearer"))

{

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

return;

}

}

}

}

**Program.cs**

using YourNamespace.Filters;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddScoped<CustomAuthFilter>();

builder.Services.AddControllers(options =>

{

options.Filters.Add<CustomExceptionFilter>();

});

builder.Services.AddEndpointsApiExplorer();

builder.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")

}

});

c.AddSecurityDefinition("Bearer", new Microsoft.OpenApi.Models.OpenApiSecurityScheme

{

Description = "Enter 'Bearer' [space] and then your token in the text input below.\r\n\r\nExample: \"Bearer abc123\"",

Name = "Authorization",

In = Microsoft.OpenApi.Models.ParameterLocation.Header,

Type = Microsoft.OpenApi.Models.SecuritySchemeType.ApiKey,

Scheme = "Bearer"

});

c.AddSecurityRequirement(new Microsoft.OpenApi.Models.OpenApiSecurityRequirement

{

{

new Microsoft.OpenApi.Models.OpenApiSecurityScheme

{

Reference = new Microsoft.OpenApi.Models.OpenApiReference

{

Type = Microsoft.OpenApi.Models.ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

Array.Empty<string>()

}

});

});

var app = builder.Build();

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

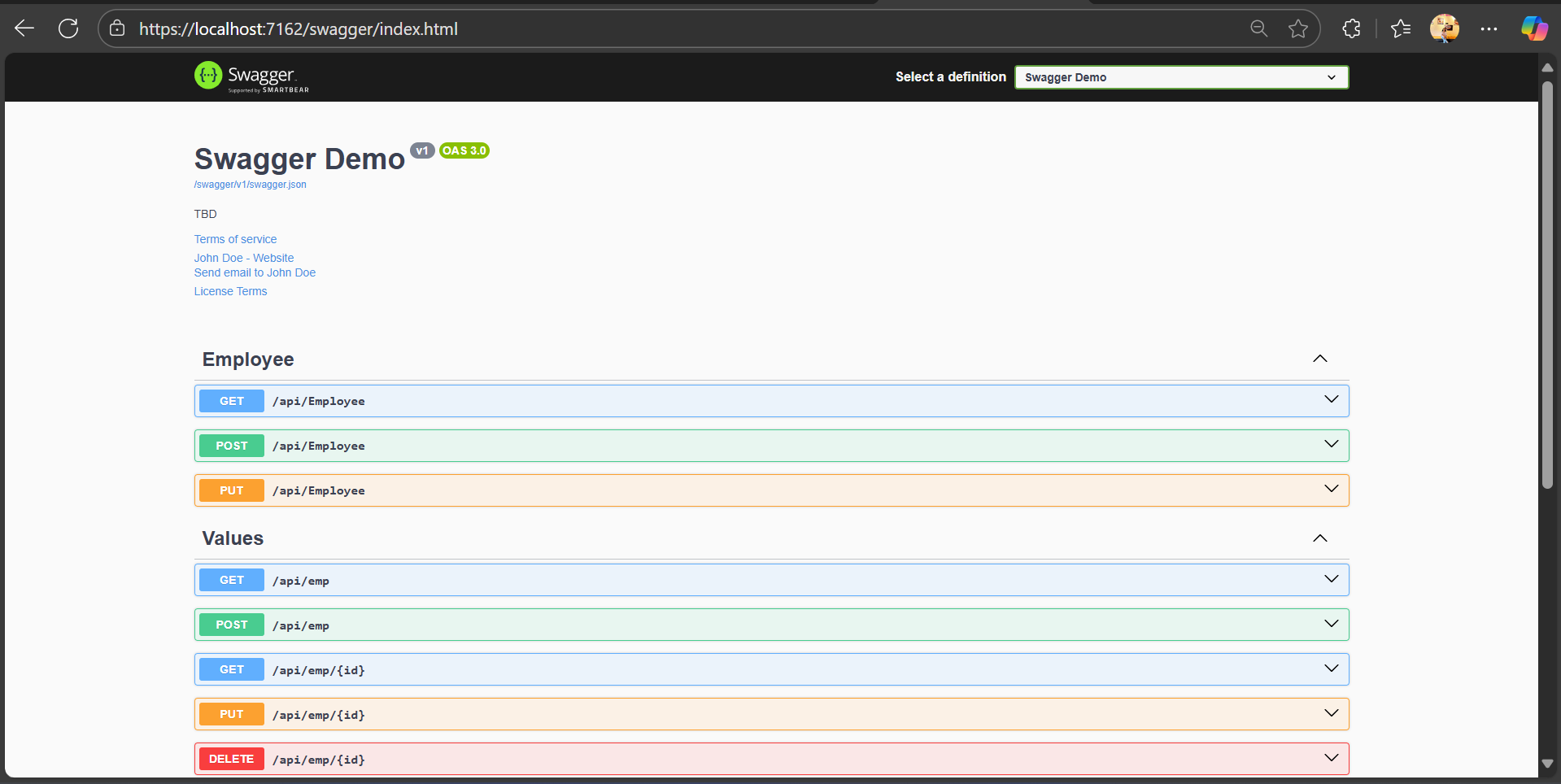
});

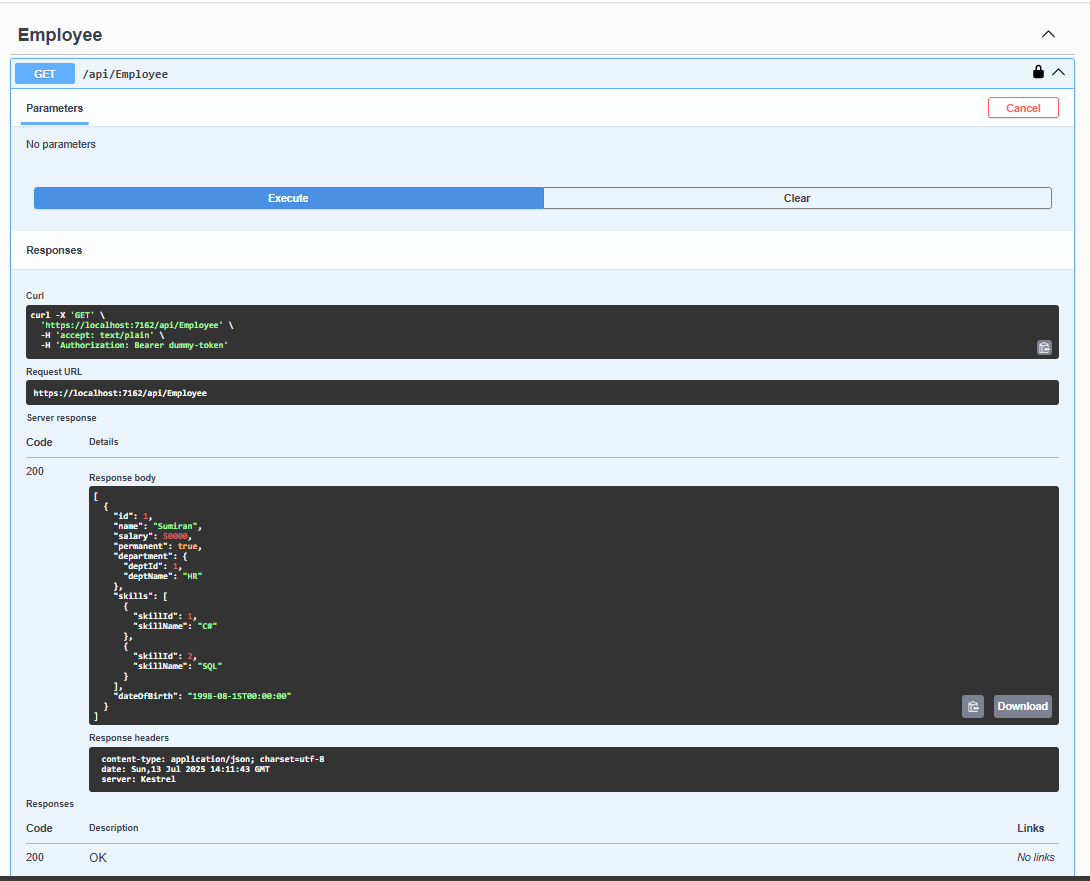
app.UseAuthorization();

app.MapControllers();

app.Run();

1. Swagger UI



1. GET (Swagger)
2. GET (Postman)