ID:6363695

LAB 4.1:On creation of the Web API, execute the application and check if the GET action method result is returned as expected.

StudentController.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using MyFirstWebAPI.Data;

using MyFirstWebAPI.Models;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace MyFirstWebAPI.Controllers

{

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

[ApiController]

public class StudentsController : ControllerBase

{

private readonly AppDbContext \_context;

public StudentsController(AppDbContext context)

{

\_context = context;

}

// GET: api/Students

[HttpGet]

public async Task<ActionResult<IEnumerable<Student>>> GetStudents()

{

return await \_context.Students.ToListAsync();

}

// GET: api/Students/5

[HttpGet("{id}")]

public async Task<ActionResult<Student>> GetStudent(int id)

{

var student = await \_context.Students.FindAsync(id);

if (student == null)

{

return NotFound();

}

return student;

}

// POST: api/Students

[HttpPost]

public async Task<ActionResult<Student>> PostStudent(Student student)

{

\_context.Students.Add(student);

await \_context.SaveChangesAsync();

return CreatedAtAction(nameof(GetStudent), new { id = student.Id }, student);

}

// PUT: api/Students/5

[HttpPut("{id}")]

public async Task<IActionResult> PutStudent(int id, Student student)

{

if (id != student.Id)

{

return BadRequest();

}

\_context.Entry(student).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!\_context.Students.Any(e => e.Id == id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

// DELETE: api/Students/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteStudent(int id)

{

var student = await \_context.Students.FindAsync(id);

if (student == null)

{

return NotFound();

}

\_context.Students.Remove(student);

await \_context.SaveChangesAsync();

return NoContent();

}

}

}

Program.cs

using Microsoft.EntityFrameworkCore;

using Microsoft.OpenApi.Models;

using MyFirstWebAPI.Data;

using MyFirstWebAPI.Data;

var builder = WebApplication.CreateBuilder(args);

// Add services

builder.Services.AddControllers();

builder.Services.AddDbContext<AppDbContext>(options =>

options.UseInMemoryDatabase("StudentDB"));

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

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

});

var app = builder.Build();

// Configure middleware

app.UseSwagger();

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "My API V1");

});

app.UseAuthorization();

app.MapControllers();

app.Run();

//Student.cs

namespace MyFirstWebAPI.Models

{

public class Student

{

public int Id { get; set; }

public string? Name { get; set; } // ✅ nullable fix

public int Age { get; set; }

}

}

//AppdbContext.cs

using Microsoft.EntityFrameworkCore;

using MyFirstWebAPI.Models; // ✅ link to Student model

namespace MyFirstWebAPI.Data

{

public class AppDbContext : DbContext

{

public AppDbContext(DbContextOptions<AppDbContext> options) : base(options)

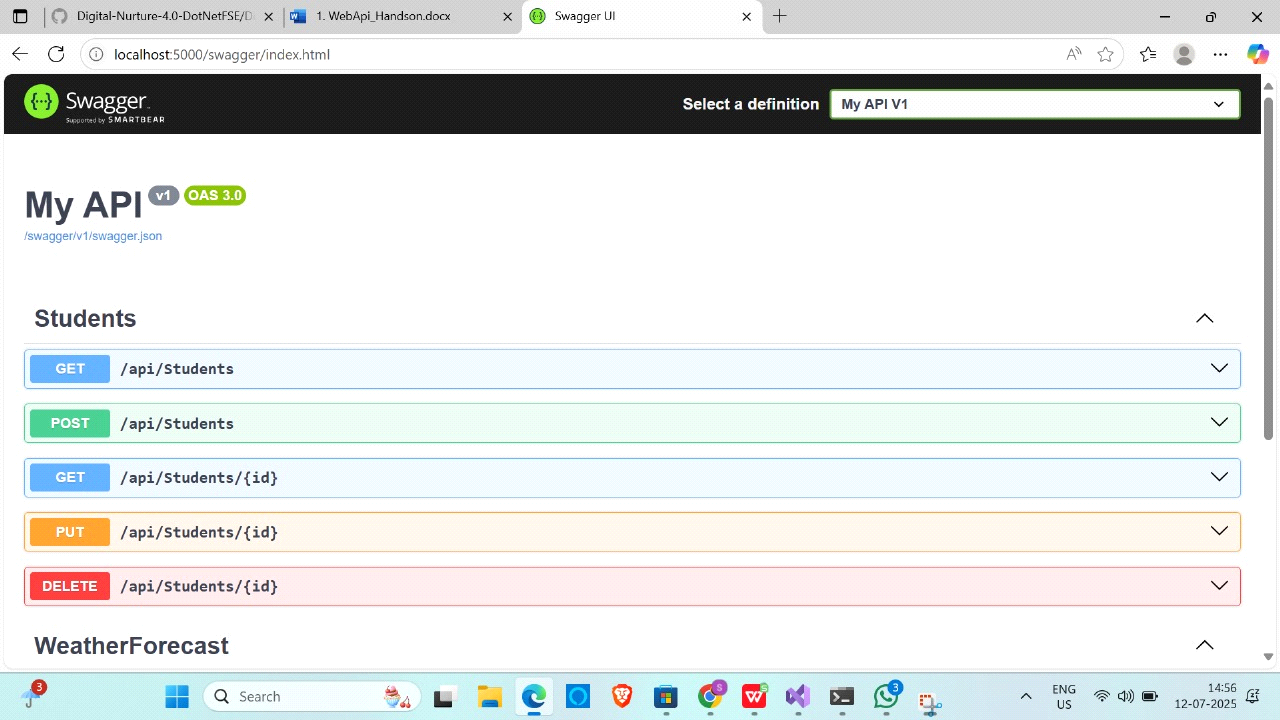
{

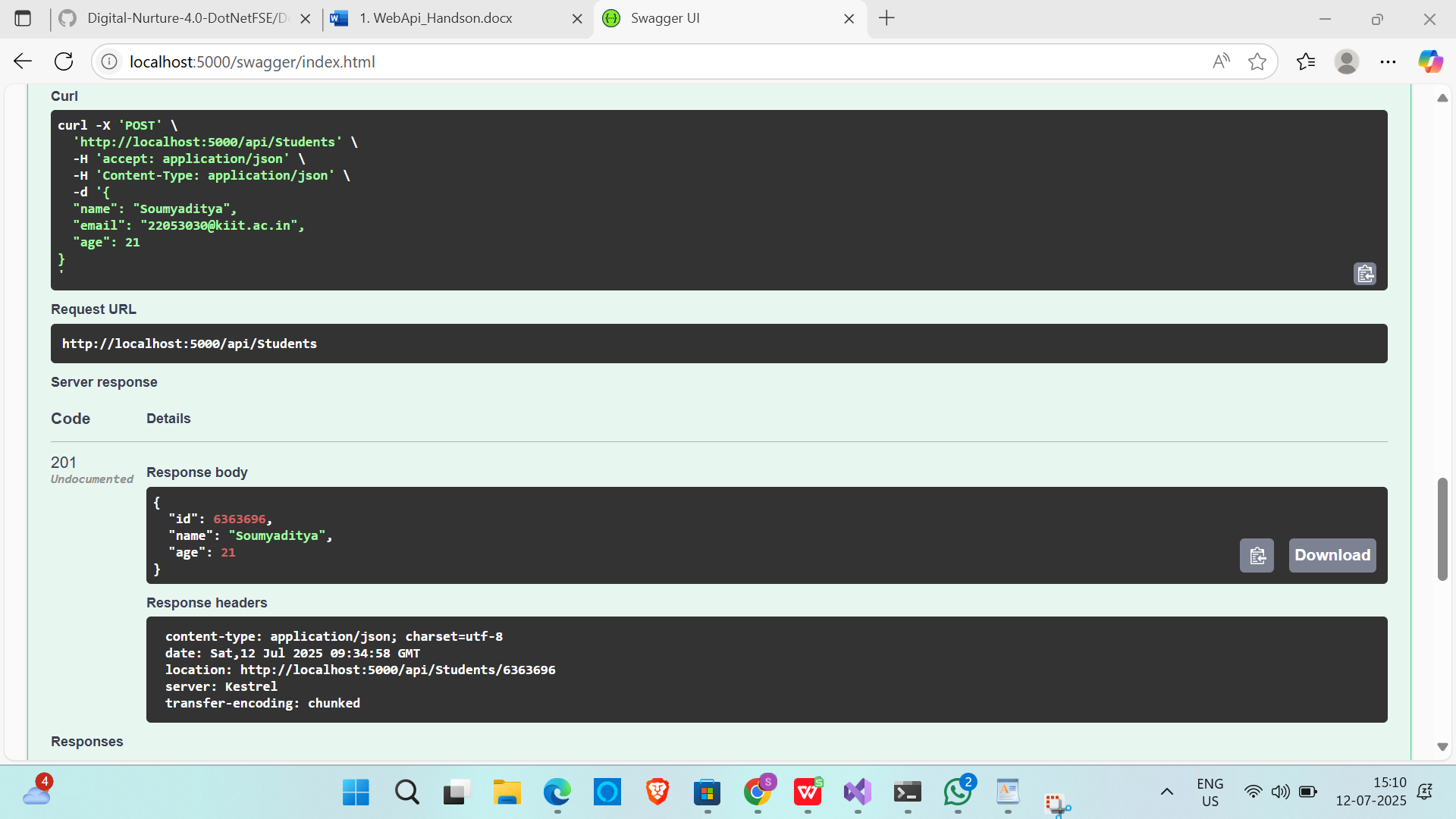
}

public DbSet<Student> Students { get; set; }

}

}





LAB:4.2

Create a .Net core web application with API template. (Use existing application if created). Install Swashbuckle.AspNetCore Nuget package. Post this do the following steps in Startup.cs

//Program.cs

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

// ✅ Add Swagger services

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "Demo API with Swagger",

Contact = new OpenApiContact

{

Name = "John Doe",

Email = "john@xyzmail.com",

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

}

});

});

var app = builder.Build();

// ✅ Enable Swagger middleware

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();

//Startup.cs

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.OpenApi.Models;

using System;

namespace MyFirstWebAPI

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// ✅ This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddControllers();

// ✅ Add Swagger

services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "TBD",

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

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")

}

});

});

}

// ✅ This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseRouting();

// ✅ Enable Swagger

app.UseSwagger();

// ✅ Enable Swagger UI

app.UseSwaggerUI(c =>

{

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

});

app.UseEndpoints(endpoints =>

{

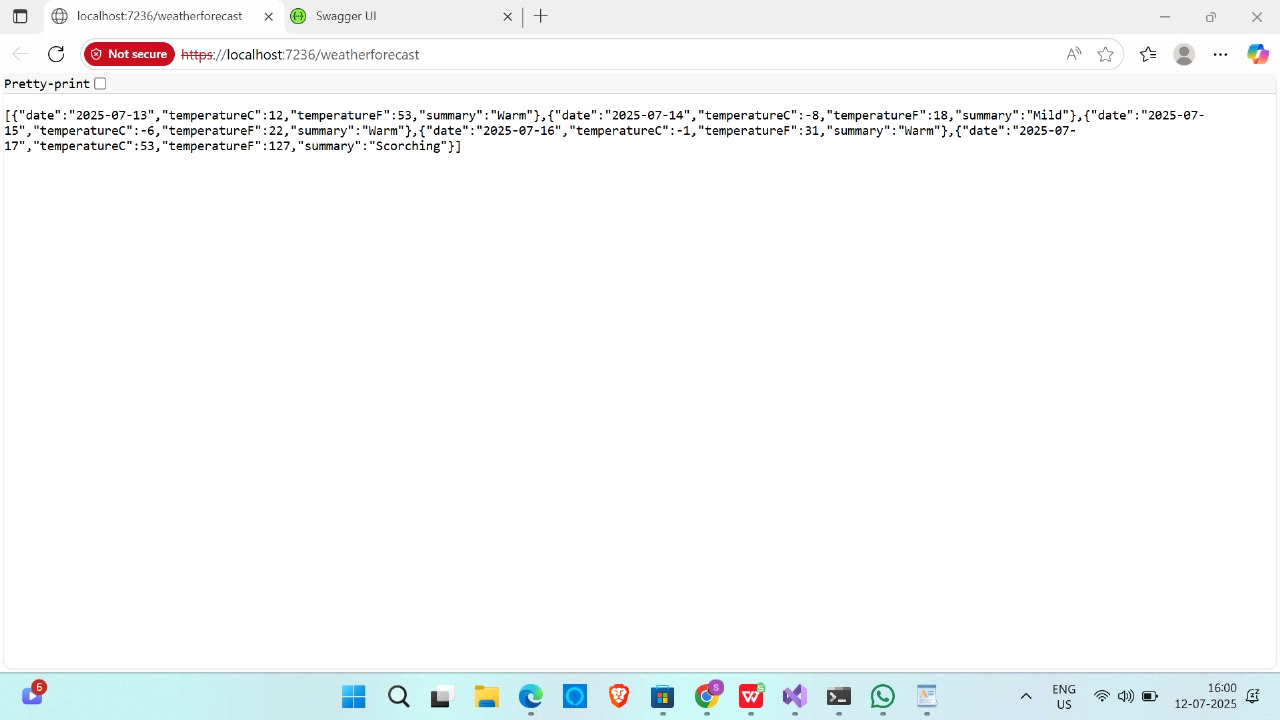
endpoints.MapControllers();

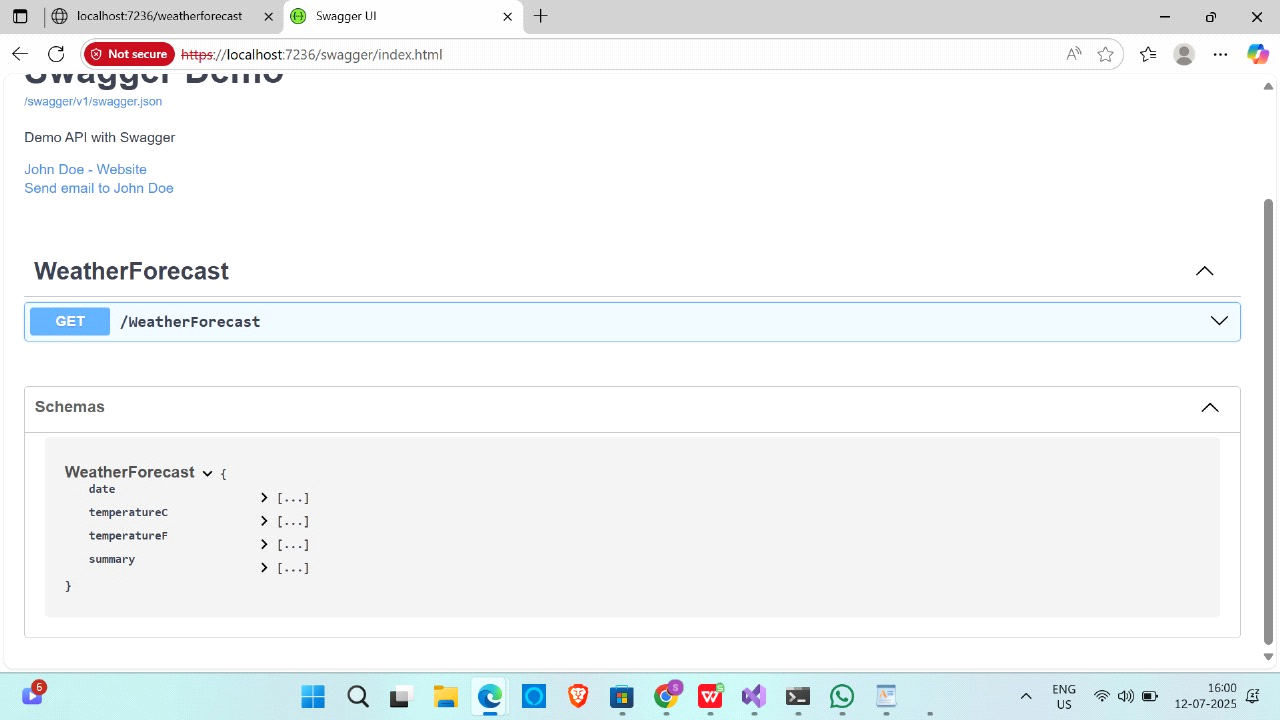
});

}

}

}





LAB 4.3

Demonstrate Custom filter

Usage of ActionFilterAttribute, OnActionExecuting method to intercept the request, Create filter for Custom exception - Need to install Microsoft.AspNetCore.Mvc.WebApiCompatShim package1. Web Api using custom model class

//Department.cs

namespace EmployeeWebAPI.Models

{

public class Department

{

public int Id { get; set; }

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

}

}

//Skill.cs

namespace EmployeeWebAPI.Models

{

public class Skill

{

public int Id { get; set; }

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

}

}

//Employee.cs

using System;

using System.Collections.Generic;

namespace EmployeeWebAPI.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 DateTime DateOfBirth { get; set; }

public Department Department { get; set; } = new Department();

public List<Skill> Skills { get; set; } = new List<Skill>();

}

}

//AppDbContext.cs

using Microsoft.EntityFrameworkCore;

using EmployeeWebAPI.Models;

namespace EmployeeWebAPI.Data

{

public class AppDbContext : DbContext

{

public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }

public DbSet<Employee> Employees { get; set; }

public DbSet<Department> Departments { get; set; }

public DbSet<Skill> Skills { get; set; }

}

}

//Program.cs

using EmployeeWebAPI.Data;

using EmployeeWebAPI.Filters;

using Microsoft.EntityFrameworkCore;

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

// ✅ Add DbContext using In-Memory DB

builder.Services.AddDbContext<AppDbContext>(options =>

options.UseInMemoryDatabase("EmployeeDB"));

// ✅ Add Controllers + Exception Filter

builder.Services.AddControllers(options =>

{

options.Filters.Add<CustomExceptionFilter>();

});

// ✅ Add Swagger support

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Employee Web API",

Version = "v1",

Description = "Demo API with Swagger and Filters",

Contact = new OpenApiContact

{

Name = "Soumyaditya Saha",

Email = "22053030@kiit.ac.in",

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

},

License = new OpenApiLicense

{

Name = "Use under MIT",

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

}

});

});

var app = builder.Build();

// ✅ Enable Swagger middleware

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

//CustomFilter.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace EmployeeWebAPI.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

context.Result = new ObjectResult(new

{

Message = "An error occurred.",

Error = context.Exception.Message

})

{

StatusCode = 500

};

}

}

}

//EmployeeContoller.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using EmployeeWebAPI.Data;

using EmployeeWebAPI.Models;

namespace EmployeeWebAPI.Controllers

{

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

[ApiController]

public class EmployeesController : ControllerBase

{

private readonly AppDbContext \_context;

public EmployeesController(AppDbContext context)

{

\_context = context;

}

// GET: api/Employees

[HttpGet]

public async Task<ActionResult<IEnumerable<Employee>>> GetEmployees()

{

return await \_context.Employees.ToListAsync();

}

// GET: api/Employees/5

[HttpGet("{id}")]

public async Task<ActionResult<Employee>> GetEmployee(int id)

{

var employee = await \_context.Employees.FindAsync(id);

if (employee == null)

{

return NotFound();

}

return employee;

}

// PUT: api/Employees/5

// To protect from overposting attacks, see <https://go.microsoft.com/fwlink/?linkid=2123754>

[HttpPut("{id}")]

public async Task<IActionResult> PutEmployee(int id, Employee employee)

{

if (id != employee.Id)

{

return BadRequest();

}

\_context.Entry(employee).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!EmployeeExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

// POST: api/Employees

// To protect from overposting attacks, see <https://go.microsoft.com/fwlink/?linkid=2123754>

[HttpPost]

public async Task<ActionResult<Employee>> PostEmployee(Employee employee)

{

\_context.Employees.Add(employee);

await \_context.SaveChangesAsync();

return CreatedAtAction("GetEmployee", new { id = employee.Id }, employee);

}

// DELETE: api/Employees/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteEmployee(int id)

{

var employee = await \_context.Employees.FindAsync(id);

if (employee == null)

{

return NotFound();

}

\_context.Employees.Remove(employee);

await \_context.SaveChangesAsync();

return NoContent();

}

private bool EmployeeExists(int id)

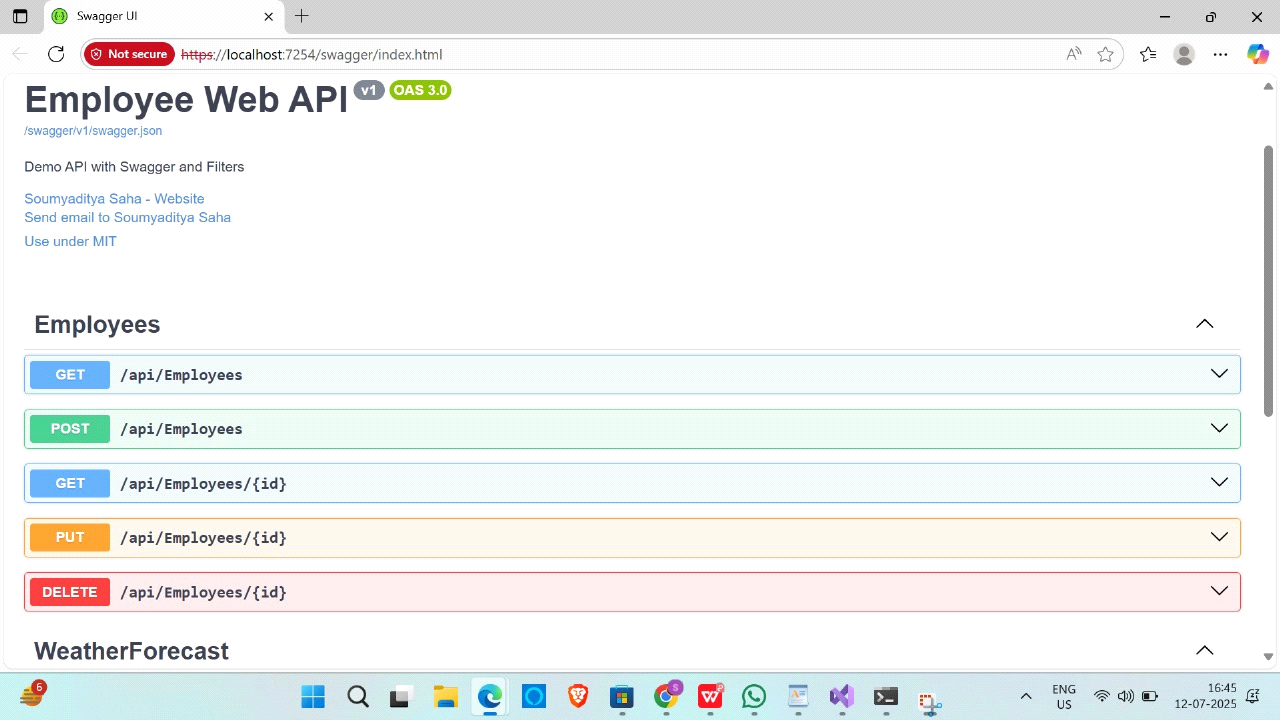
{

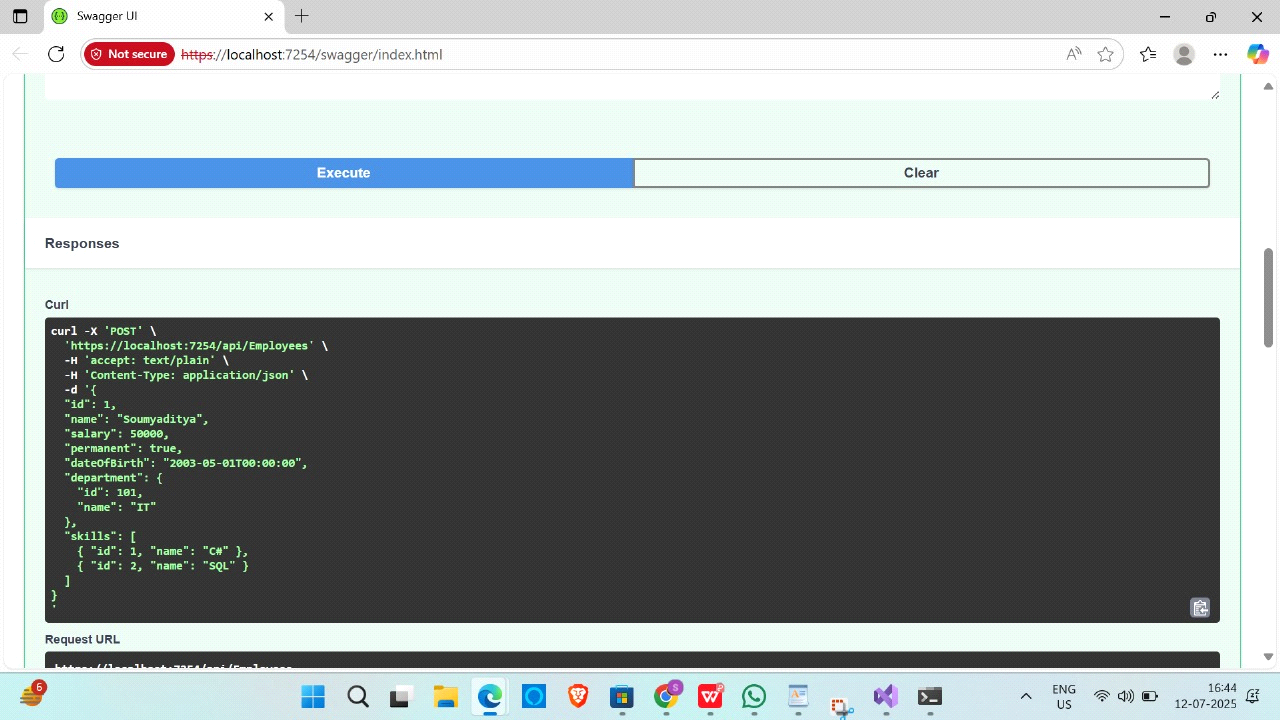
return \_context.Employees.Any(e => e.Id == id);

}

}

}





//LAB4.4:

//Employee.cs

namespace EmployeeCrudAPI.Models

{

public class Employee

{

public int Id { get; set; }

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

public int Salary { get; set; }

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

}

}

//EmployeeController.cs

using EmployeeCrudAPI.Models;

using Microsoft.AspNetCore.Mvc;

namespace EmployeeCrudAPI.Controllers

{

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

[ApiController]

public class EmployeeController : ControllerBase

{

private static List<Employee> employees = new List<Employee>

{

new Employee { Id = 1, Name = "Soumyaditya", Salary = 50000, Department = "IT" },

new Employee { Id = 2, Name = "Ravi", Salary = 40000, Department = "HR" }

};

[HttpGet]

public ActionResult<IEnumerable<Employee>> GetEmployees()

{

return Ok(employees);

}

[HttpPost]

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

{

employee.Id = employees.Max(e => e.Id) + 1;

employees.Add(employee);

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

}

[HttpPut("{id}")]

public ActionResult<Employee> PutEmployee(int id, [FromBody] Employee updatedEmployee)

{

if (id <= 0)

return BadRequest("Invalid employee id");

var existing = employees.FirstOrDefault(e => e.Id == id);

if (existing == null)

return BadRequest("Invalid employee id");

existing.Name = updatedEmployee.Name;

existing.Salary = updatedEmployee.Salary;

existing.Department = updatedEmployee.Department;

return Ok(existing);

}

[HttpDelete("{id}")]

public ActionResult DeleteEmployee(int id)

{

var employee = employees.FirstOrDefault(e => e.Id == id);

if (employee == null)

return NotFound("Employee not found");

employees.Remove(employee);

return Ok($"Employee with ID {id} deleted.");

}

}

}

//Program.cs

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

// ✅ Enable Swagger

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Demo API with Swagger and Filters",

Version = "v1",

Description = "Employee Web API",

Contact = new OpenApiContact

{

Name = "Soumyaditya Saha",

Email = "22053030@kiit.ac.in",

Url = new Uri("https://yourwebsite.com") // You can update this

},

License = new OpenApiLicense

{

Name = "Use under MIT",

Url = new Uri("https://opensource.org/licenses/MIT")

}

});

});

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

}

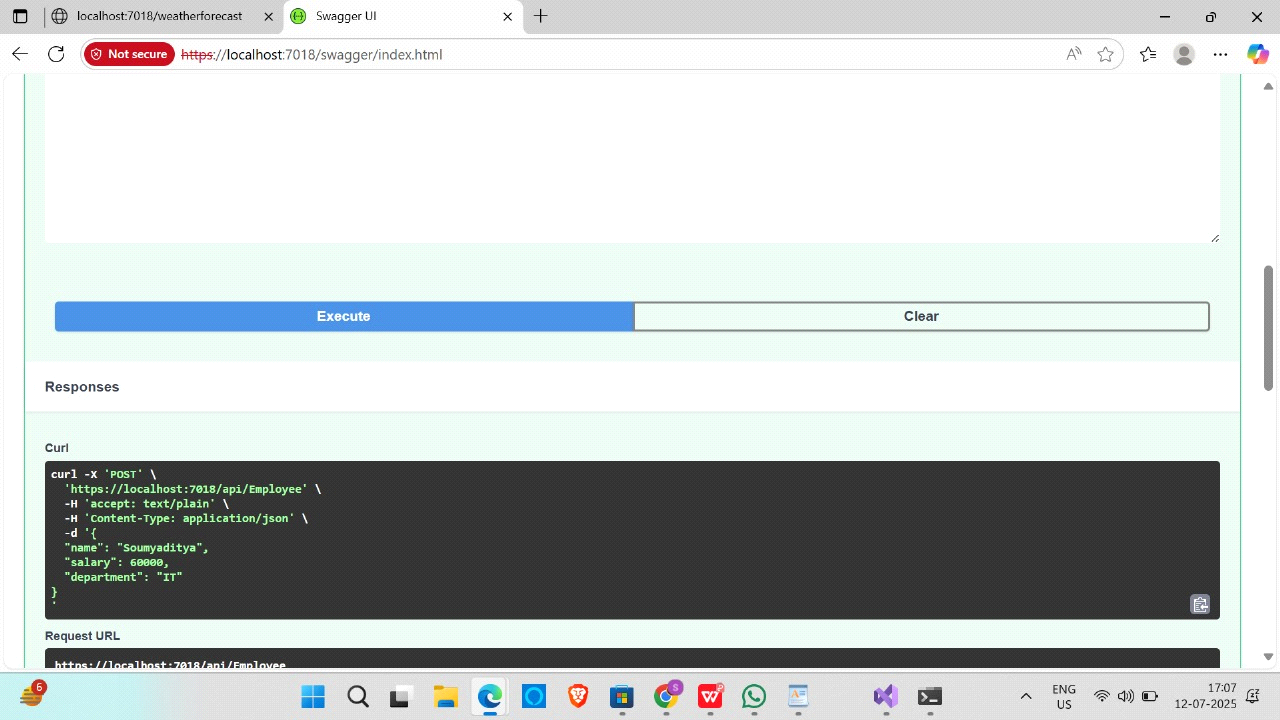
app.UseHttpsRedirection();

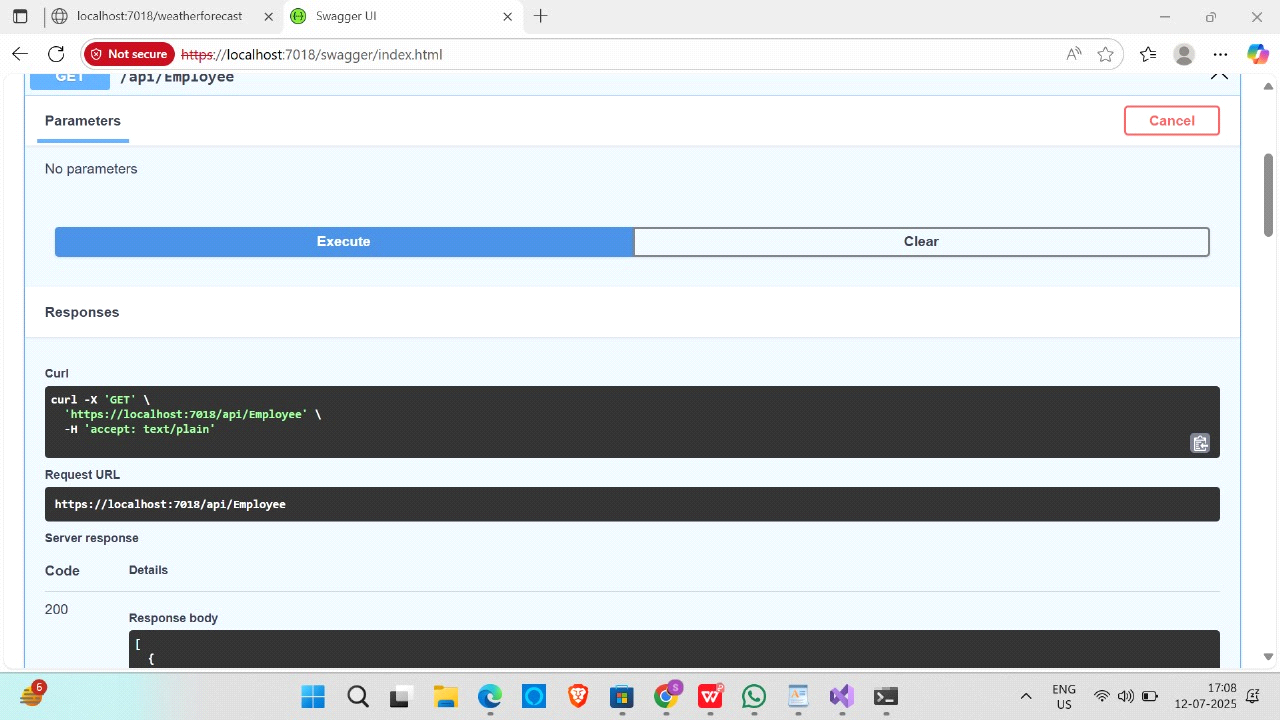
app.UseAuthorization();

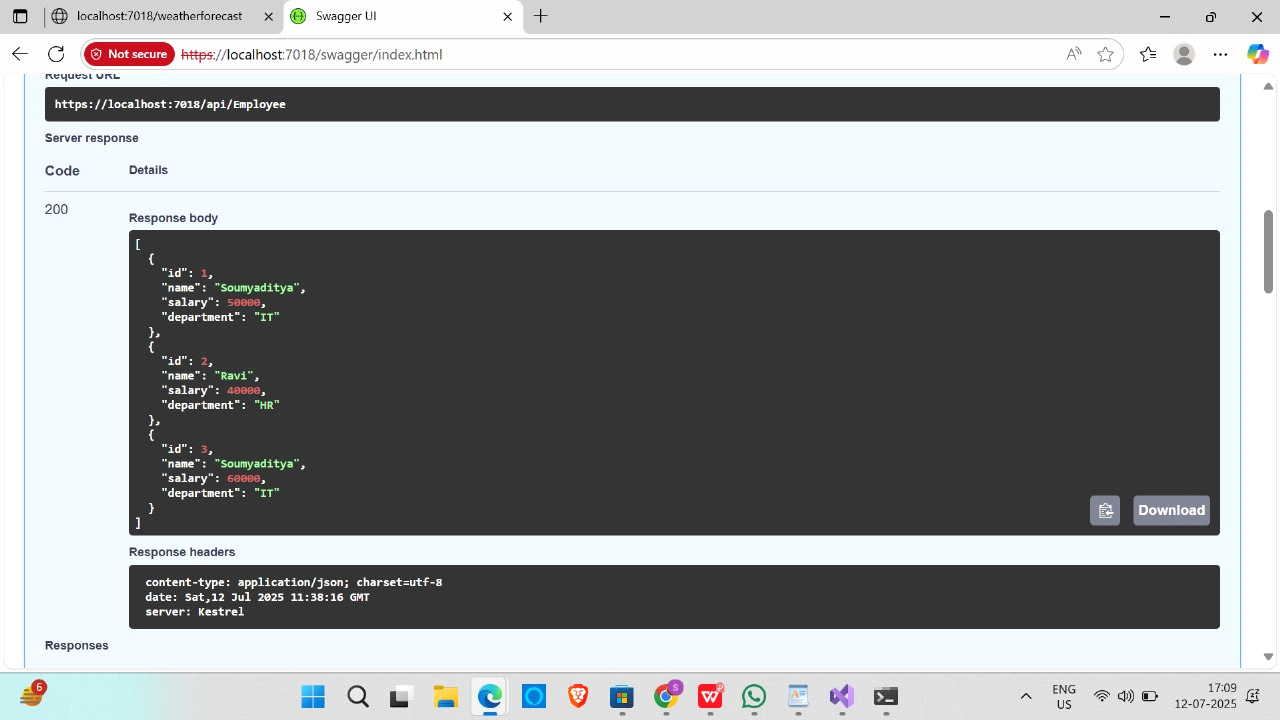
app.MapControllers(); // ✅ This is essential to map controller endpoints

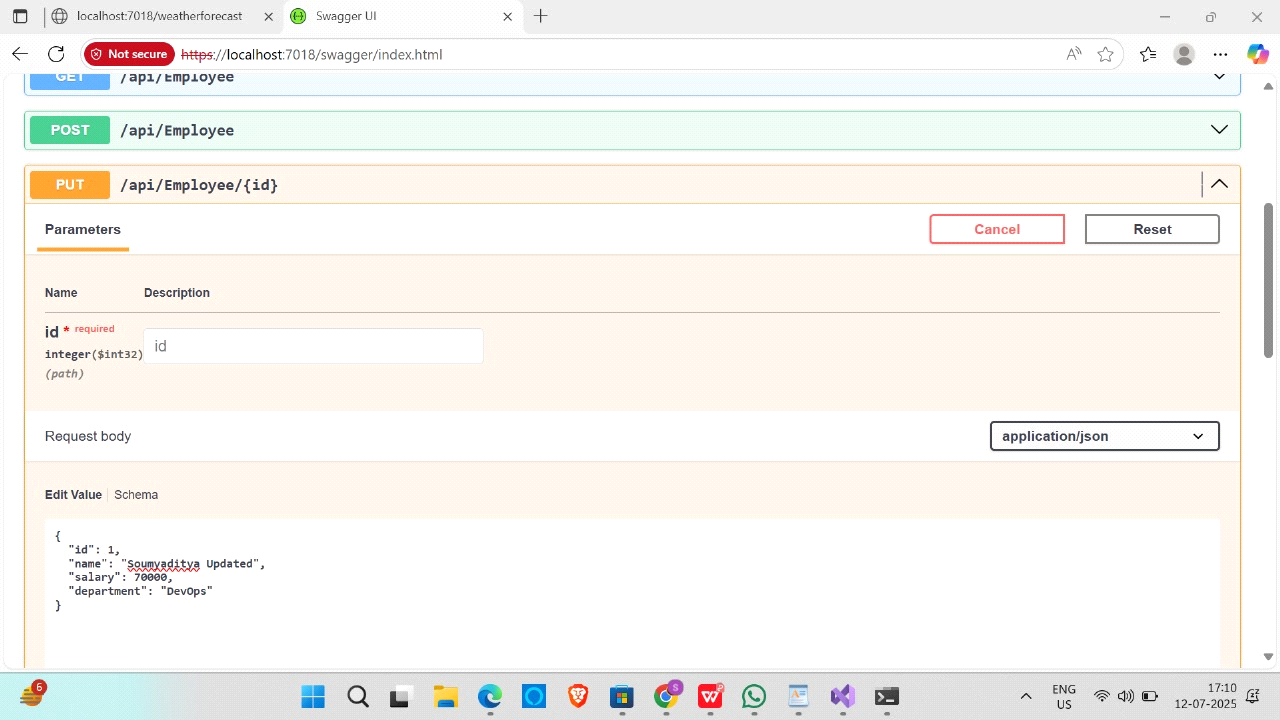
app.Run();

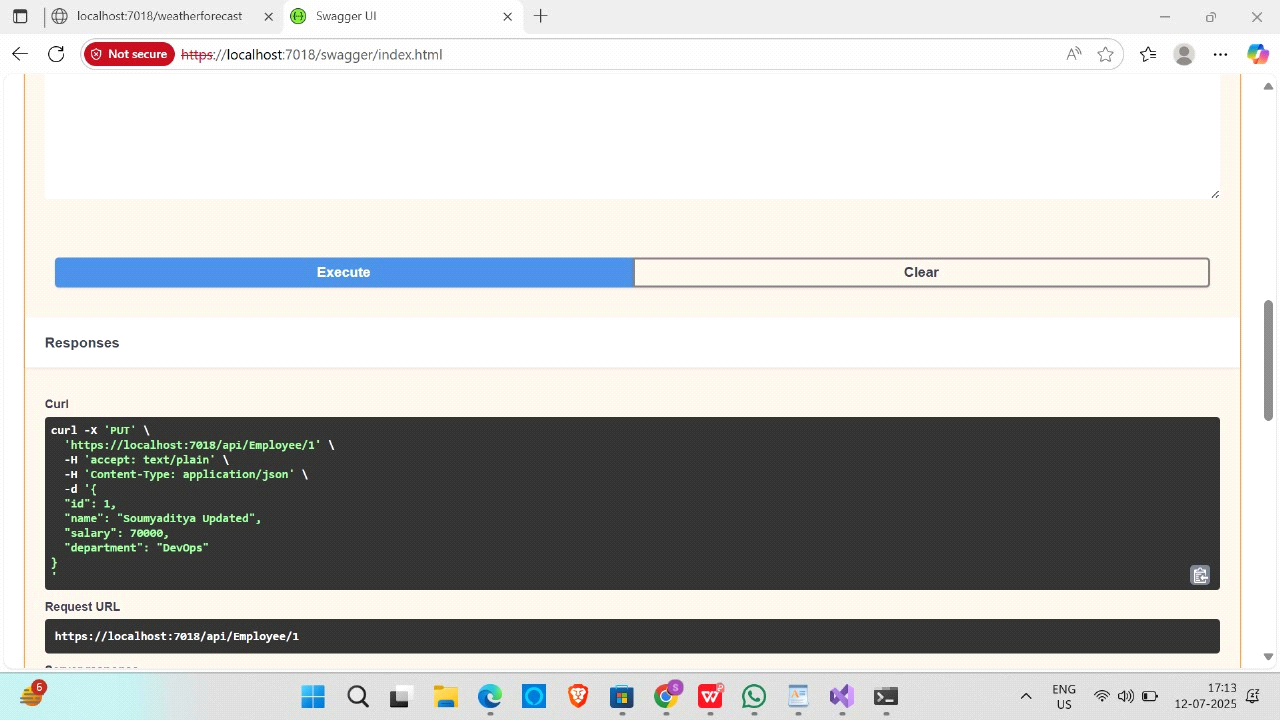
OUTPUT:

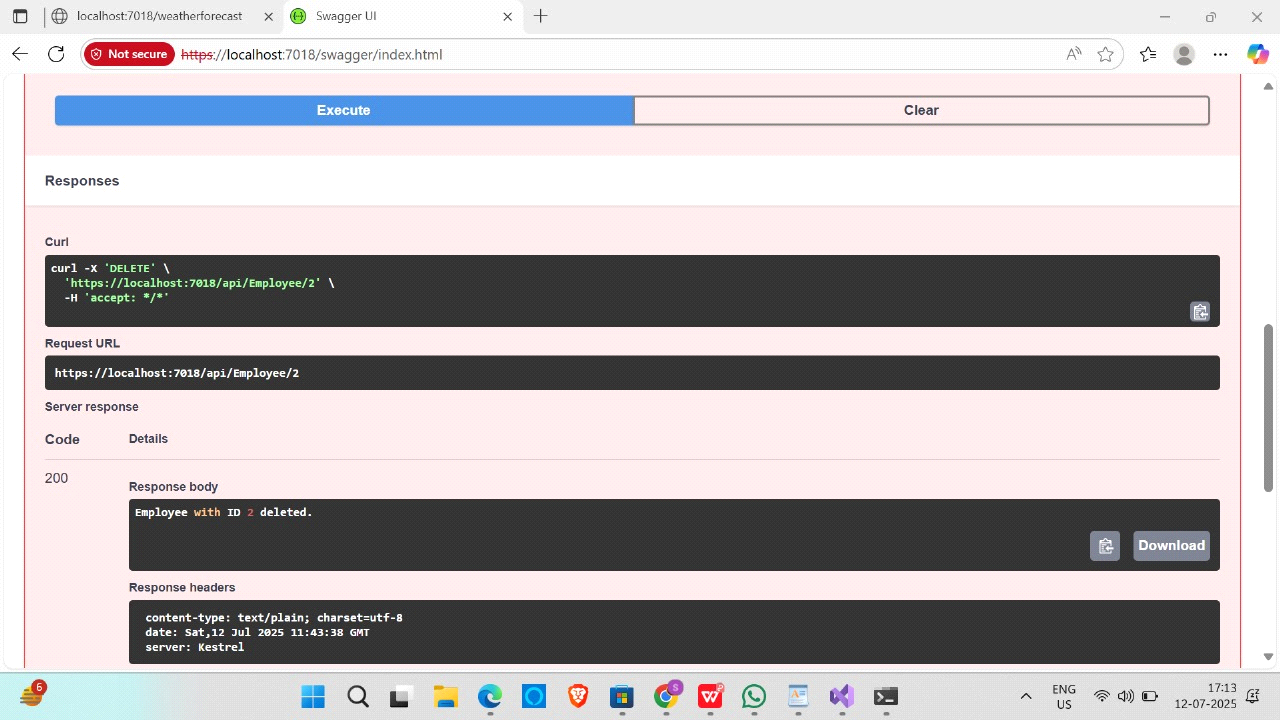


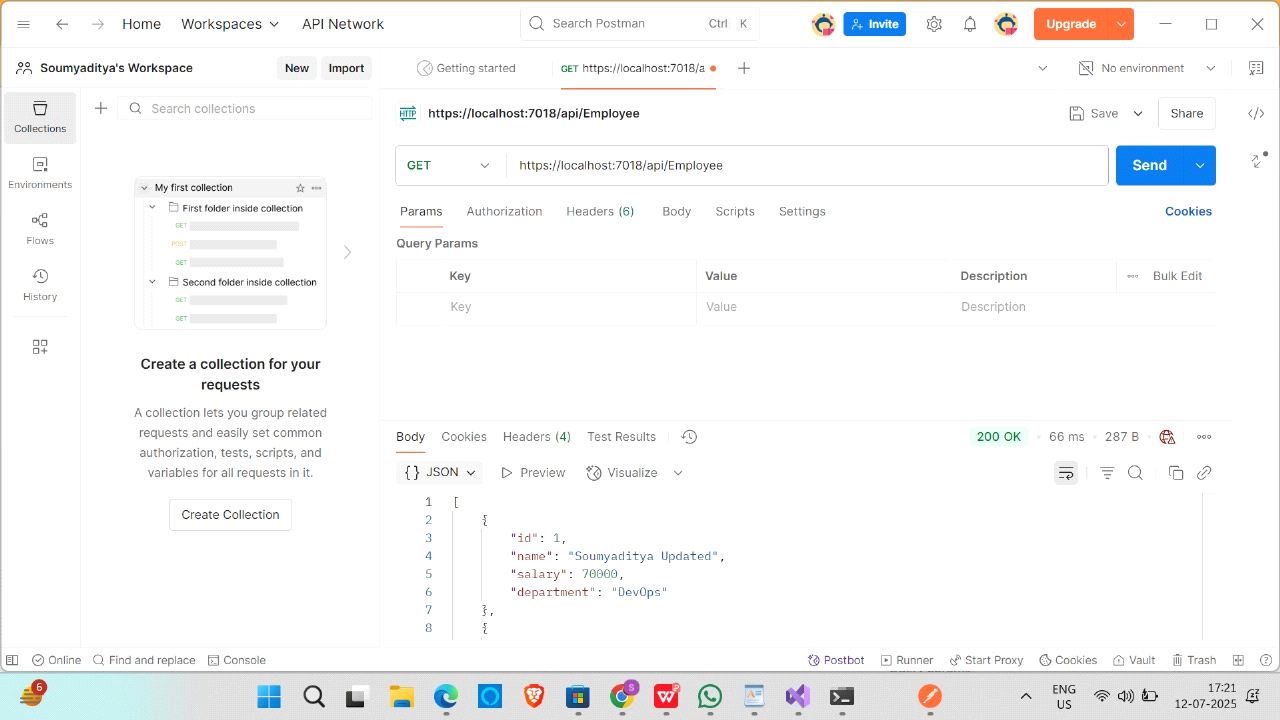












LAB:4.5

//JWT Auth Web API

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace JwtAuthWebAPI.Controllers

{

[ApiController]

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

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

public class EmployeeController : ControllerBase

{

[HttpGet("GetAll")]

public IActionResult GetAll()

{

var employees = new List<string> { "Soumya", "Rahul", "Priya" };

return Ok(employees);

}

}

}

//Program.cs

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

// Set up long enough security key

var securityKey = "mysuperdupersecretkey12345678901234"; // Minimum 32 characters

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

// Add Authentication

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 = symmetricSecurityKey

};

});

// Add CORS

builder.Services.AddCors(options =>

{

options.AddPolicy("AllowAll", policy =>

{

policy.AllowAnyOrigin()

.AllowAnyHeader()

.AllowAnyMethod();

});

});

// Add Controllers

builder.Services.AddControllers();

// Add Swagger

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

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

// Enable JWT in Swagger

c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

In = ParameterLocation.Header,

Description = "Enter 'Bearer' [space] and then your 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();

// Middleware

app.UseCors("AllowAll");

app.UseAuthentication();

app.UseAuthorization();

app.UseSwagger();

app.UseSwaggerUI();

app.MapControllers();

app.Run();

//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 JwtAuthWebAPI.Controllers

{

[ApiController]

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

public class AuthController : ControllerBase

{

private readonly string \_securityKey = "mysuperdupersecretkey12345678901234";

[HttpGet("login")]

[AllowAnonymous]

public IActionResult Login()

{

var token = GenerateJSONWebToken(123, "Admin");

return Ok(new { Token = token });

}

private string GenerateJSONWebToken(int userId, string userRole)

{

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

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);

}

}

}

OUTPUT:

