**Skill:-ASP.NET Core 8.0 Web API**

**Filename:-1. WebApi\_Handson**

**Name:- Document-1**

**Solution:-**

1. **Explain the concept of RESTful web service, Web API & Microservice ?**

**Answer:- **REST**:** Rules for designing APIs using ****HTTP****.

It follows principles like:

**Stateless**: Server doesn’t remember anything about the user between requests.

**Resources**: Everything is a resource (users, products, etc.)

**Uses HTTP methods** like GET, POST, PUT, DELETE.

**Web API**: A type of RESTful service. We build it using .NET so that other apps or frontend (like React/Angular) can talk to our server.

**WebService vs WebAPI**:

| **Web Service** | **Web API** |
| --- | --- |
| SOAP-based | REST-based |
| Returns XML | Returns XML/JSON |
| Heavy | Lightweight |

**Microservice**: Breaking big applications into small independent services. Each does one job and talks to others via APIs.

1. Explain what is Http Request & Http Response

**Answer:-**

**HttpRequest**: What your browser/app sends to the server (e.g., "Hey, give me user list").

**HttpResponse**: What server sends back (e.g., "Here are the users").

List the types of Action Verbs

### Action Verbs

Used to tell what kind of operation you are doing:

| **Verb** | **Purpose** |
| --- | --- |
| HttpGet | Read or fetch data |
| HttpPost | Create new data |
| HttpPut | Update existing data |
| HttpDelete | Delete data |

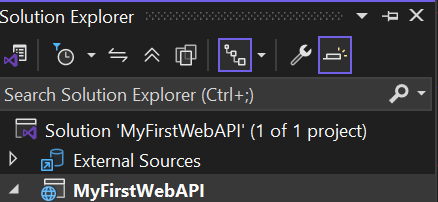
Demonstrate creation of a simple WebAPI - With Read, Write actions

### HTTP Status Codes

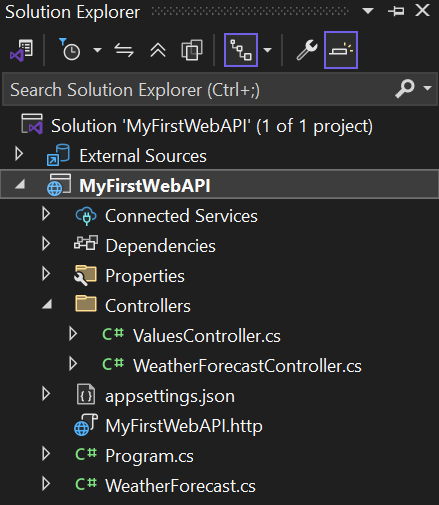
These are response codes returned by the server:

| **Code** | **Meaning** |
| --- | --- |
| 200 OK | Success |
| 400 BadRequest | Invalid input |
| 401 Unauthorized | Login needed |
| 500 InternalServerError | Server failed |

***Created MyFirstWebAPI using ASP.NET Core Web API and with* ***.NET 8.0*****



*****Created*** *Controller with Read/Write with name of Values Controller***



***Code For ValuesController.cs in MyFirstWebAPI***

using Microsoft.AspNetCore.Mvc;

namespace MyFirstWebAPI.Controllers

{

[ApiController]

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

public class ValuesController : ControllerBase

{

[HttpGet]

public ActionResult<IEnumerable<string>> Get()

{

return new string[] { "value1", "value2" };

}

[HttpGet("{id}")]

public ActionResult<string> Get(int id)

{

return $"You requested value {id}";

}

[HttpPost]

public IActionResult Post([FromBody] string value)

{

return Ok($"You posted: {value}");

}

[HttpPut("{id}")]

public IActionResult Put(int id, [FromBody] string value)

{

return Ok($"You updated ID {id} with value {value}");

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

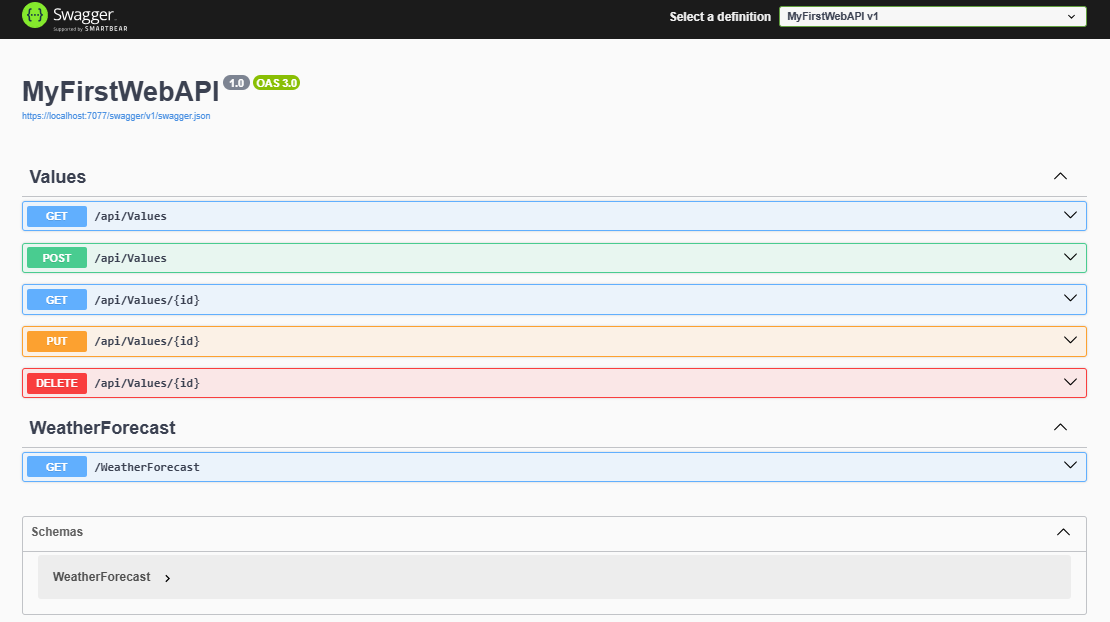
return Ok($"You deleted value with ID {id}");

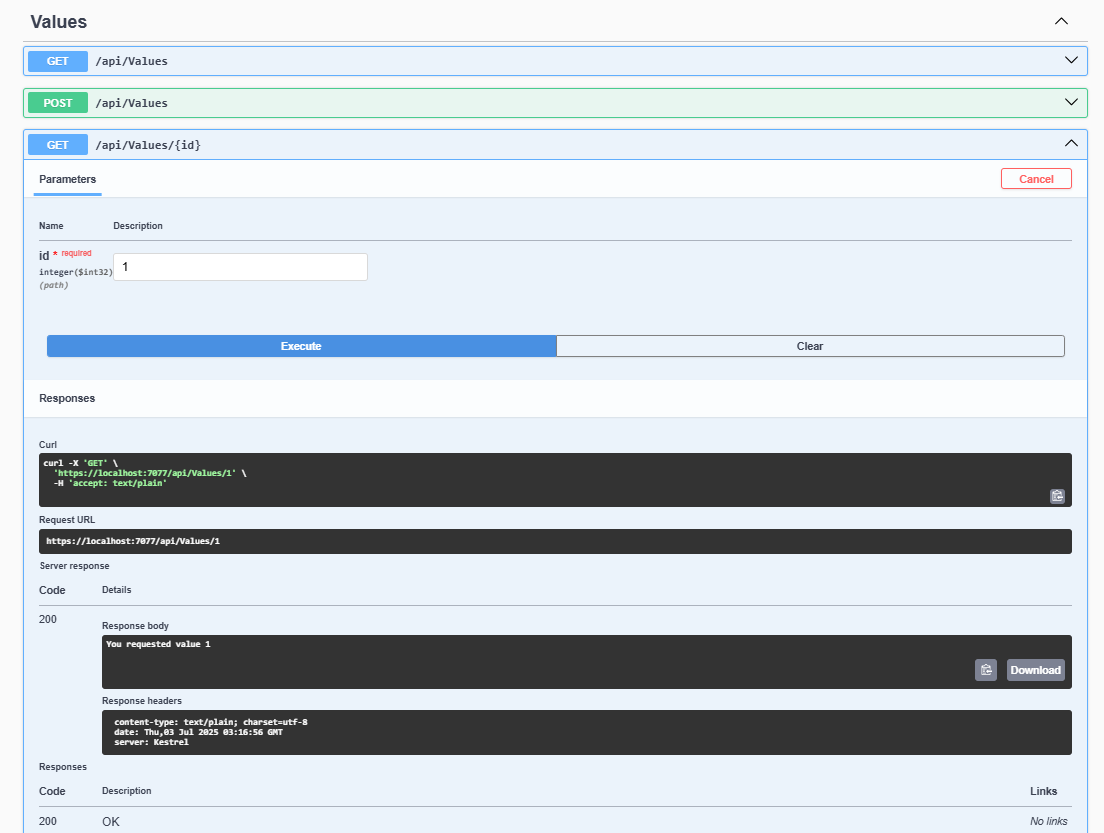
}

}

}

***Run and Test***

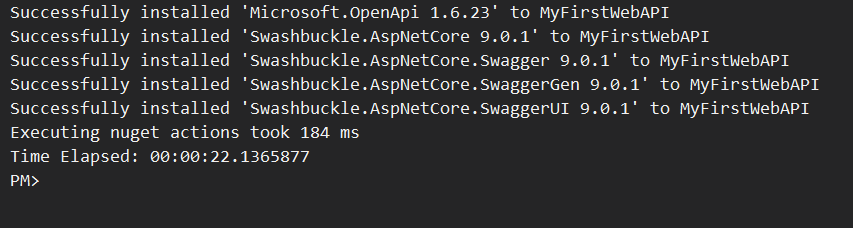




**Name:- Document-2**

**Solution:-**

***Installed Swashbuckle.AspNetCore Nuget package:***



***Code for Program.cs in MyFirstWebApi to Enable Swagger:***

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 = "1.0.0",

Description = "API for demonstration",

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",

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

}

});

});

var app = builder.Build();

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

app.MapGet("/", () => Results.Redirect("/swagger"));

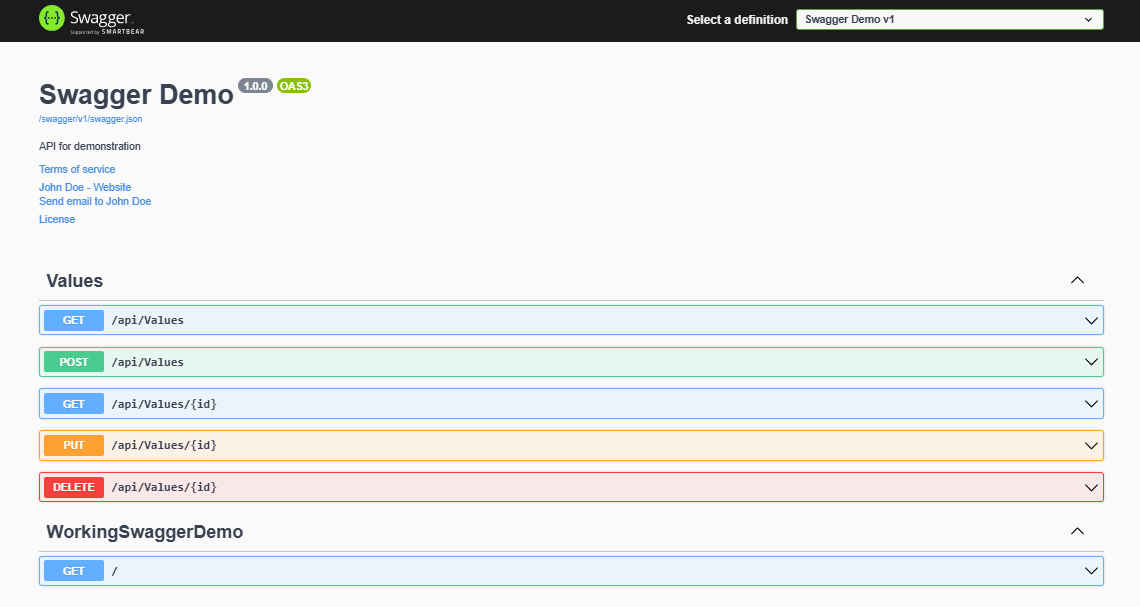
app.UseHttpsRedirection();

app.UseAuthorization();

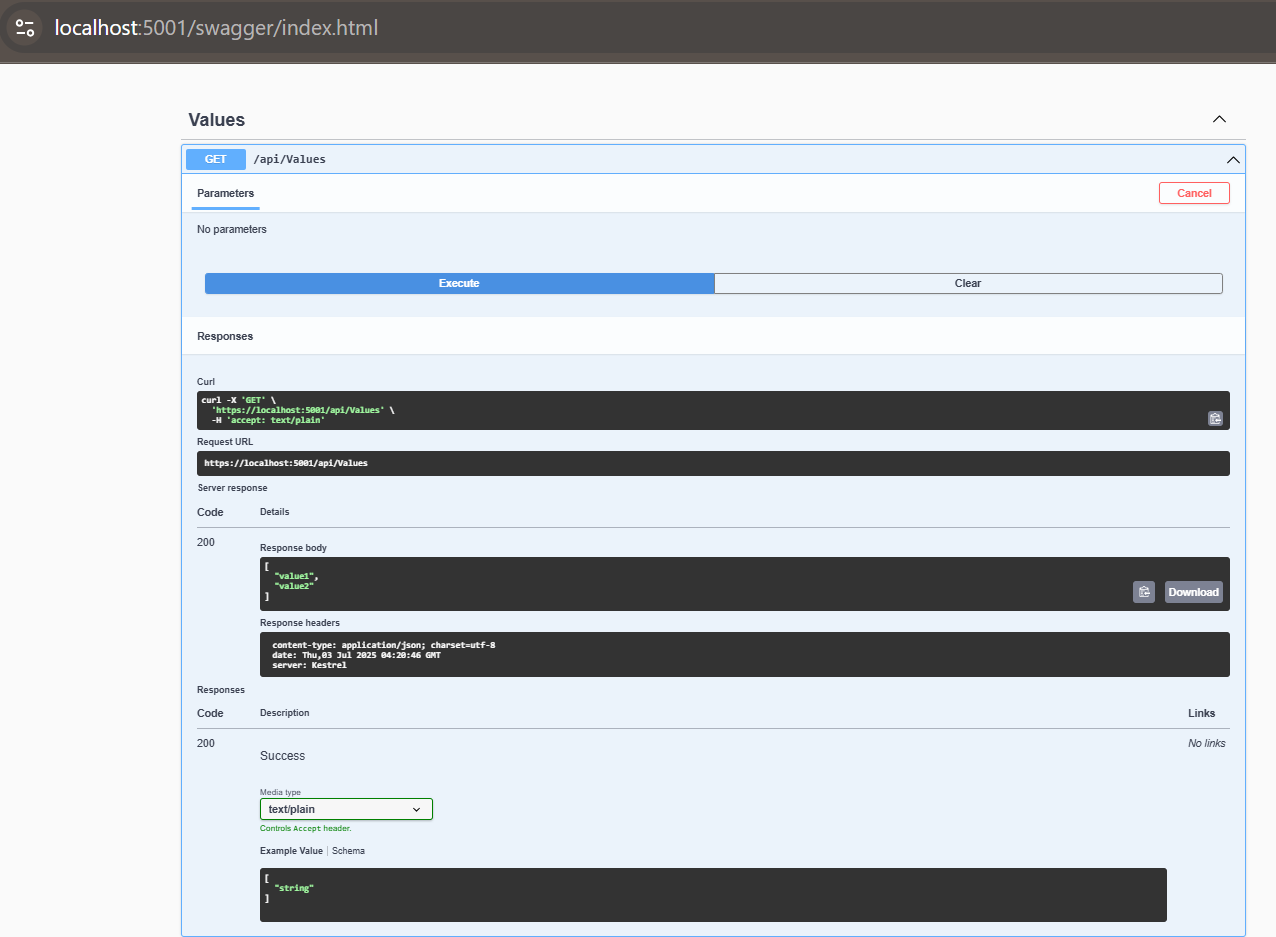
app.MapControllers();

app.Run();

***Run & Verify Swagger UI:***



***It opens a panel which has ‘Try it out’ button. Click that and Click ‘Execute’ button.***



**2.Use POSTMAN tool, to point to the local Web API that was created with Employee controller. Test the GET action method using POSTMAN.Verify the output if the List of employees are listed in the ‘Body’ part of the GET window on POSTMAN tool.**

**Verify the Status on the right side of the output pane on POSTMAN tool.**

***Code for EmployeeController.cs in Controllers to Get in Postman:***

using Microsoft.AspNetCore.Mvc;

using WorkingSwaggerDemo.Models;

namespace WorkingSwaggerDemo.Controllers

{

[ApiController]

[Route("api/Emp")]

public class EmployeeController : ControllerBase

{

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

{

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

new Employee { Id = 2, Name = "Rahul", Department = "HR" }

};

[HttpGet]

public ActionResult<IEnumerable<Employee>> Get()

{

return Ok(employees);

}

[HttpPost]

public IActionResult Post([FromBody] Employee emp)

{

if (emp == null || string.IsNullOrWhiteSpace(emp.Name))

return BadRequest("Invalid employee data.");

employees.Add(emp);

return Ok("Employee added");

}

}

}

***Code for ValuesController.cs in Controllers to Get in Postman***

using Microsoft.AspNetCore.Mvc;

namespace WorkingSwaggerDemo.Controllers

{

[ApiController]

[Route("api/Values")]

public class ValuesController : ControllerBase

{

[HttpGet]

public ActionResult<IEnumerable<string>> Get()

{

return Ok(new string[] { "value1", "value2" });

}

[HttpGet("{id}")]

public ActionResult<string> Get(int id)

{

if (id <= 0)

return NotFound("Invalid ID");

return Ok($"You requested value {id}");

}

[HttpPost]

public IActionResult Post([FromBody] string value)

{

if (string.IsNullOrWhiteSpace(value))

return BadRequest("Value cannot be empty");

return Ok($"You posted: {value}");

}

[HttpPut("{id}")]

public IActionResult Put(int id, [FromBody] string value)

{

if (id <= 0 || string.IsNullOrWhiteSpace(value))

return BadRequest("Invalid input");

return Ok($"You updated ID {id} with value {value}");

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

if (id <= 0)

return NotFound("Invalid ID");

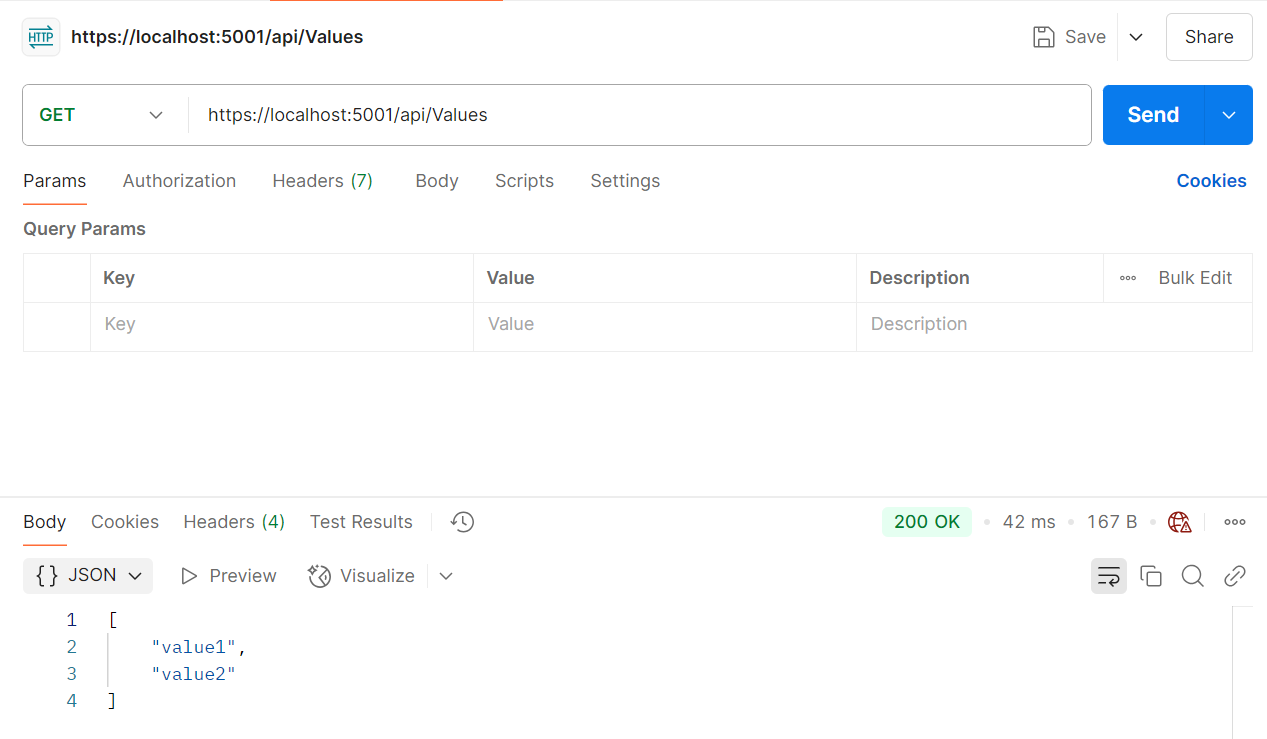
return Ok($"You deleted value with ID {id}");

}

}

}

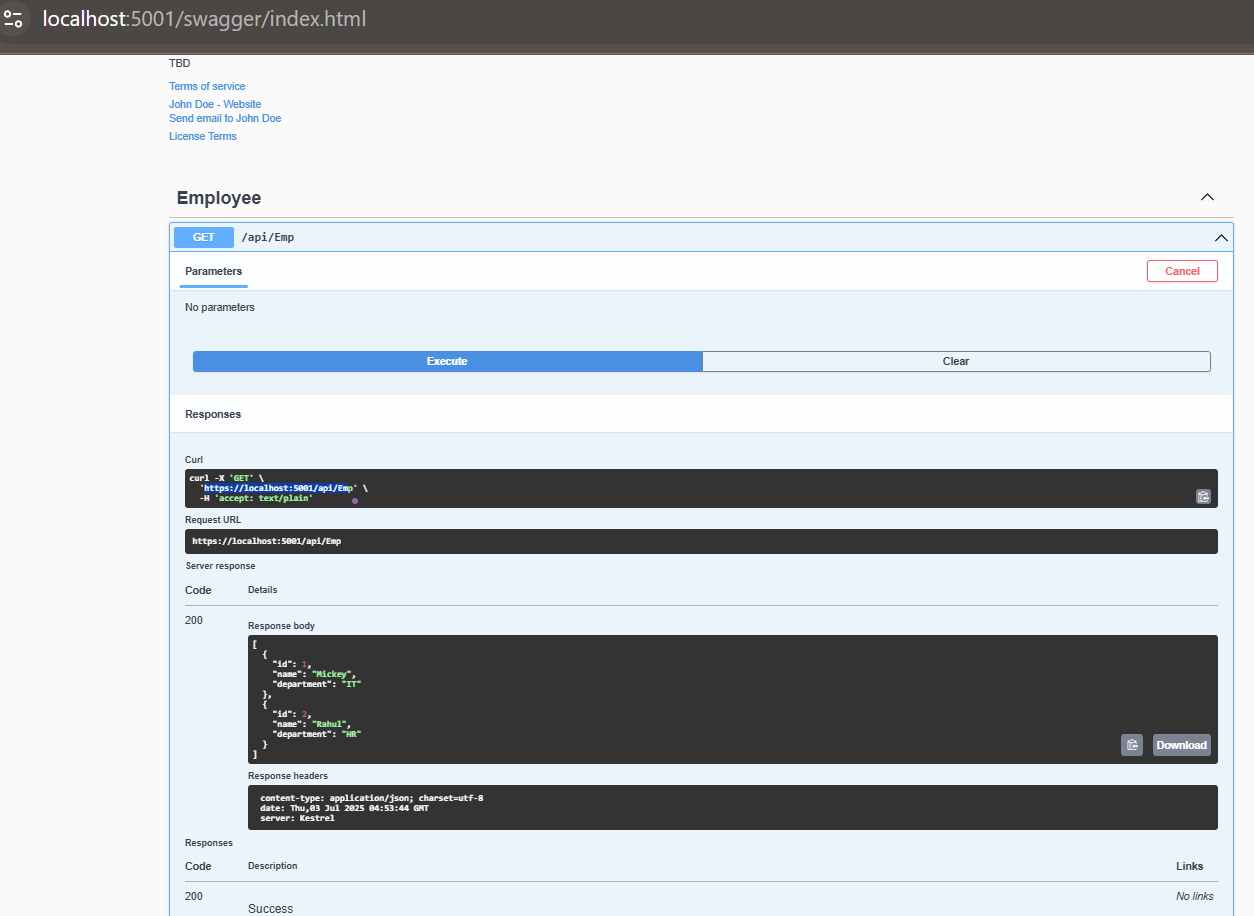
***Verify the Status on the right side of the output pane on POSTMAN tool.***

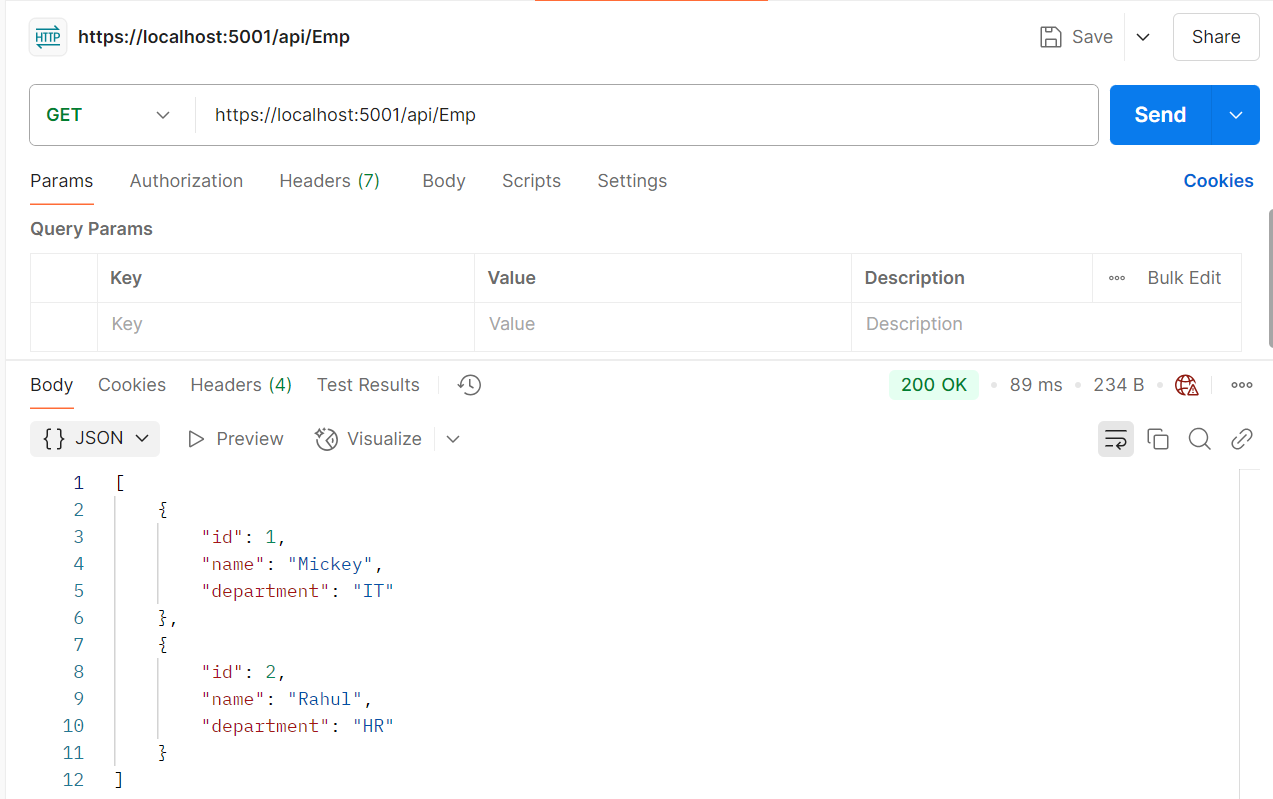


1. **Modify the Controller name in the Route attribute of the Employee controller to ‘Emp’ and check its access thru POSTMAN ?**

## *Open POSTMAN and Test:*

### A. ****GET Request****





**B.** **POST Request**

***Code for EmployeeController.cs in Controllers to POST in Postman:***

using Microsoft.AspNetCore.Mvc;

namespace WorkingSwaggerDemo.Controllers

{

[ApiController]

[Route("api/Emp")]

public class EmployeeController : ControllerBase

{

[HttpPost]

public IActionResult Post([FromBody] string name)

{

if (string.IsNullOrWhiteSpace(name))

return BadRequest("Name cannot be empty");

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

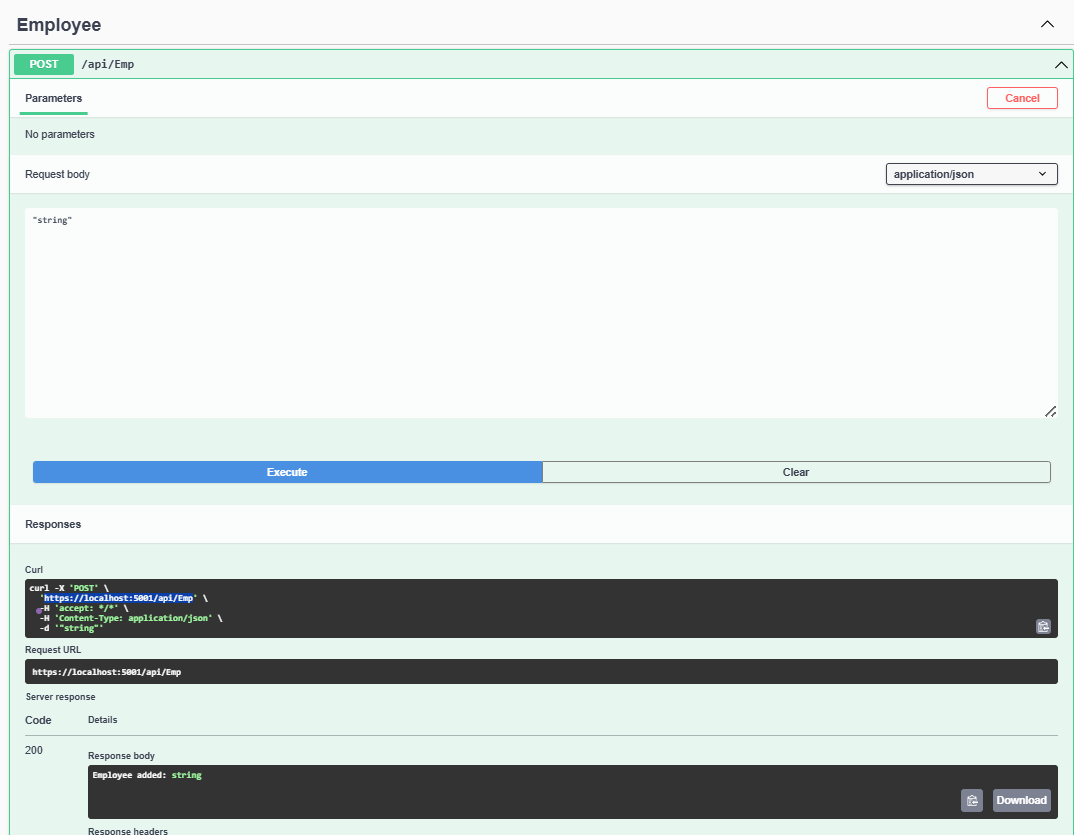
}

}

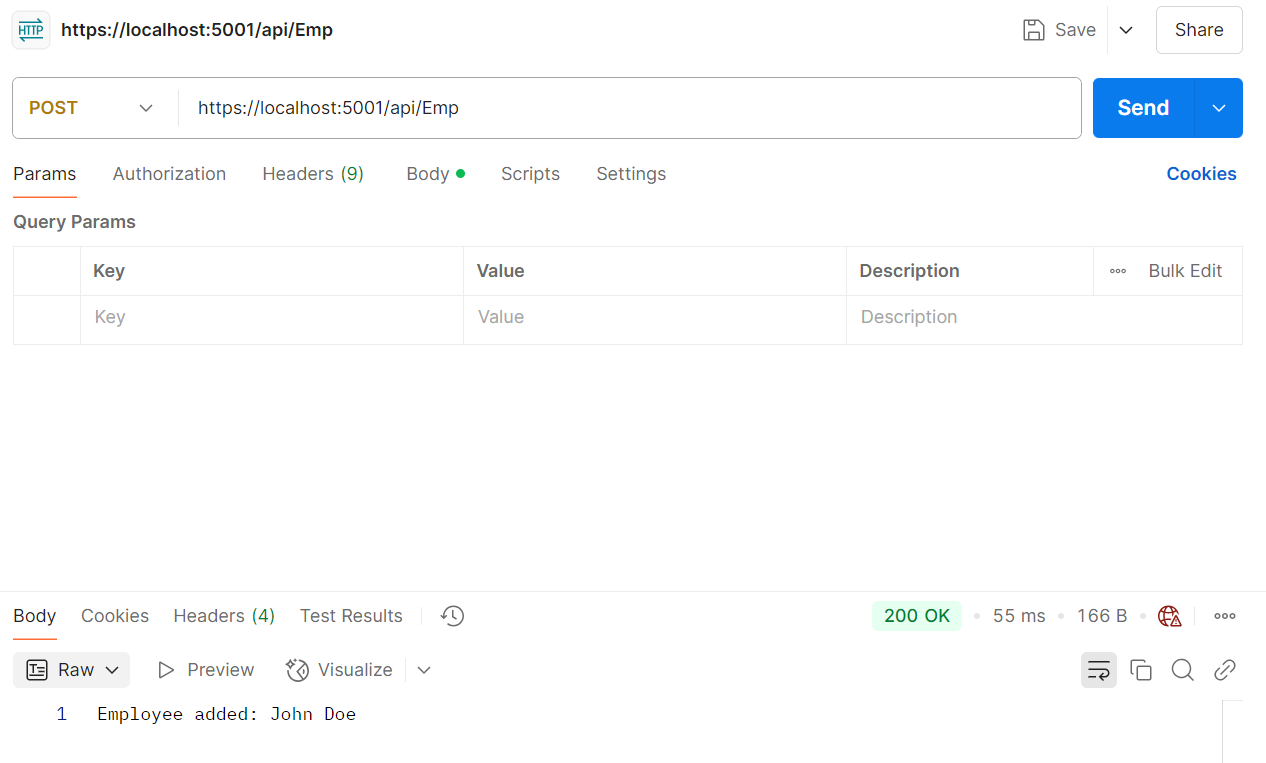
}

## *Open Local Server and POSTMAN to Verify the Tests:*

***Checking on Local Server:***



***Checking on Postman to Verifying the details:***

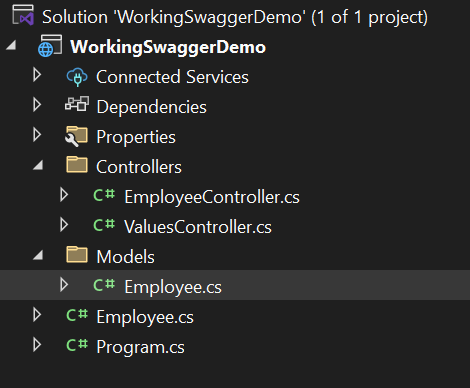


**Name:- Document-3**

**Solution:-**

***Web Api using custom model class***

***Create Employee Model Folder and Employee.cs class in Model Folder:***



***Code for Employee.cs in Models***

using System;

using System.Collections.Generic;

namespace WorkingSwaggerDemo.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; } = new();

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

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; }

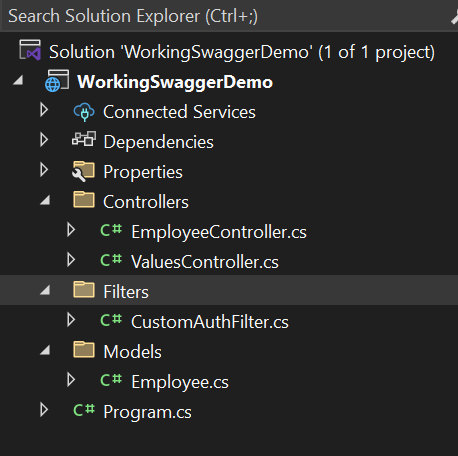
public string Name { get; set; } = "";

}

}

***Update EmployeeController.cs in Controllers with this***

***Create Filters Folder and CustomAuthFilter.cs class in Filter Folder:***



***Code for CustomAuthFilter.cs in Filters***

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace WorkingSwaggerDemo.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

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

{

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");

}

}

}

}

***Updated Code For EmployeeController.cs in Controllers***

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using WorkingSwaggerDemo.Filters;

using WorkingSwaggerDemo.Models;

namespace WorkingSwaggerDemo.Controllers

{

[ApiController]

[Route("api/Emp")]

[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 = "Hrishav Ranjan",

Salary = 50000,

Permanent = true,

DateOfBirth = new DateTime(2003, 11, 5),

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

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "C#" },

new Skill { Id = 2, Name = "ASP.NET Core" }

}

}

};

}

[HttpGet]

[AllowAnonymous]

[ProducesResponseType(StatusCodes.Status200OK)]

public ActionResult<List<Employee>> Get()

{

return Ok(\_employees);

}

[HttpPost]

[ProducesResponseType(StatusCodes.Status200OK)]

public IActionResult Post([FromBody] Employee emp)

{

return Ok($"Employee added: {emp.Name}");

}

}

}

***Updated Code for Program.cs***

using Microsoft.OpenApi.Models;

using WorkingSwaggerDemo.Filters;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddScoped<CustomAuthFilter>();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "Custom WebAPI with Filters",

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();

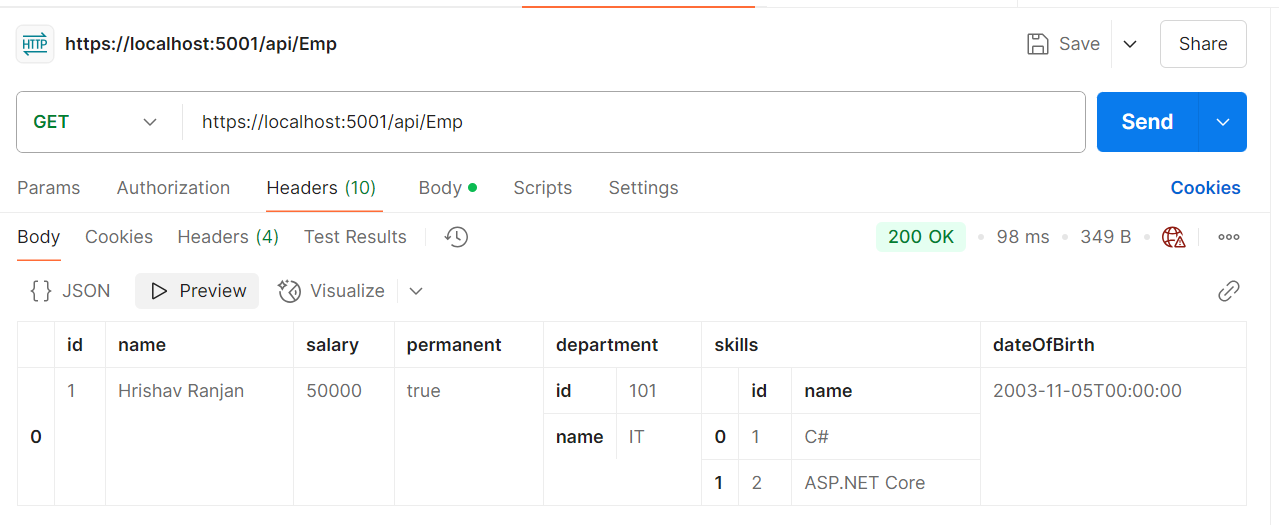
app.UseAuthorization();

app.MapControllers();

app.Run();

***After Getig Connected now in Postman Add Authorization Header , Key-Authorization , Value-Bearer test-token after that we have to verify the below details:***

***Test GET Request***



***TEST POST REQUEST***

***In BODY Tab choose RAW and then give data is:***

{

  "id": 2,

  "name": "Mickey",

  "salary": 45000,

  "permanent": **true**,

  "dateOfBirth": "2004-10-05T00:00:00",

  "department": {

    "id": 102,

    "name": "HR"

  },

  "skills": [

    {

      "id": 1,

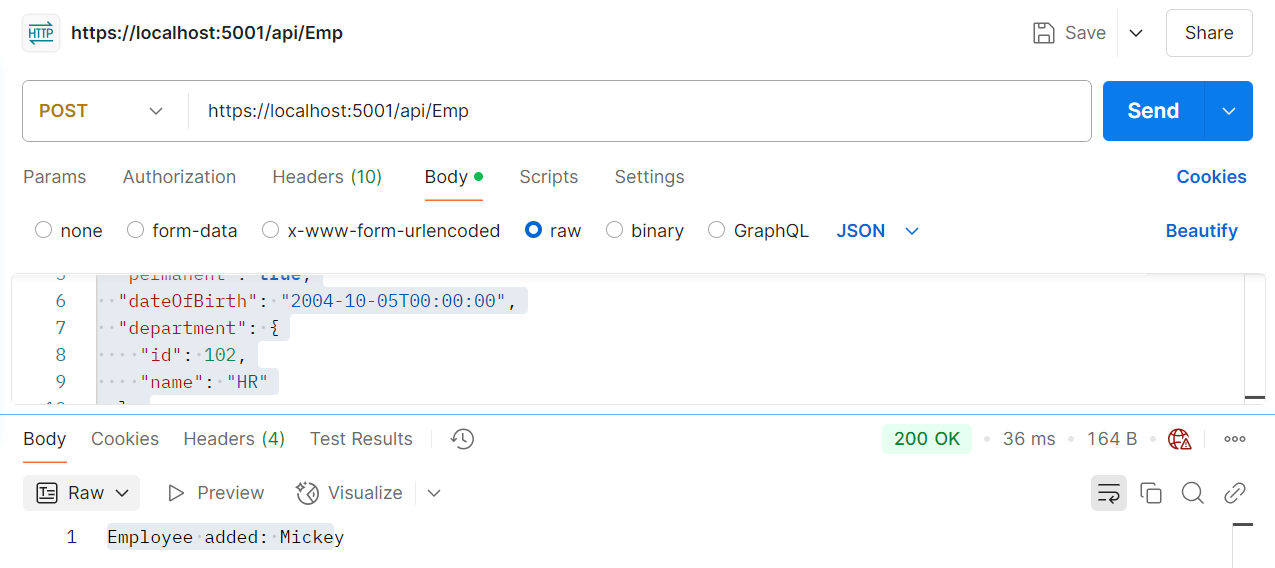
      "name": "Java"

    }

  ]

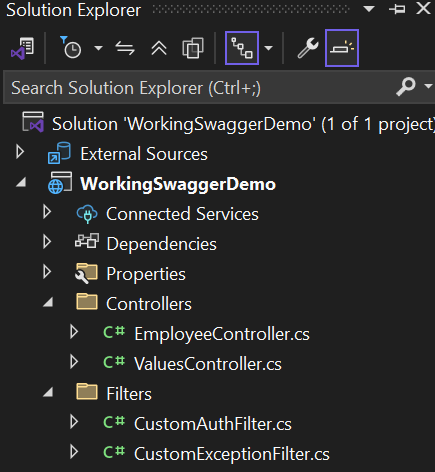
}

***After adding and clicking on Send the data is been added and below attached Picture Clearly Shows that:***



***Create a Custom action filter for Authorization.***

***Create CustomExceptionFilter.cs in Filters***



***Code CustomExceptionFilter.cs in Filters***

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System.IO;

namespace WorkingSwaggerDemo.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

var exception = context.Exception;

string logFilePath = "error\_log.txt";

File.AppendAllText(logFilePath, $"[{DateTime.Now}] {exception.Message}\n");

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

{

StatusCode = 500

};

context.ExceptionHandled = true;

}

}

}

***Updated Code for Program.cs***

using Microsoft.OpenApi.Models;

using WorkingSwaggerDemo.Filters;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddScoped<CustomAuthFilter>();

builder.Services.AddScoped<CustomExceptionFilter>();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "Custom WebAPI with Filters",

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();

***Updted Code for EmployeeController.cs***

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using WorkingSwaggerDemo.Filters;

using WorkingSwaggerDemo.Models;

namespace WorkingSwaggerDemo.Controllers

{

[ApiController]

[Route("api/Emp")]

[ServiceFilter(typeof(CustomAuthFilter))]

[ServiceFilter(typeof(CustomExceptionFilter))]

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 = "Hrishav Ranjan",

Salary = 50000,

Permanent = true,

DateOfBirth = new DateTime(2003, 11, 5),

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

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "C#" },

new Skill { Id = 2, Name = "ASP.NET Core" }

}

}

};

}

[HttpGet]

[AllowAnonymous]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public ActionResult<List<Employee>> Get()

{

throw new Exception("This is a test exception from GET.");

}

[HttpPost]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

public IActionResult Post([FromBody] Employee emp)

{

if (emp == null || string.IsNullOrEmpty(emp.Name))

{

return BadRequest("Invalid Employee data");

}

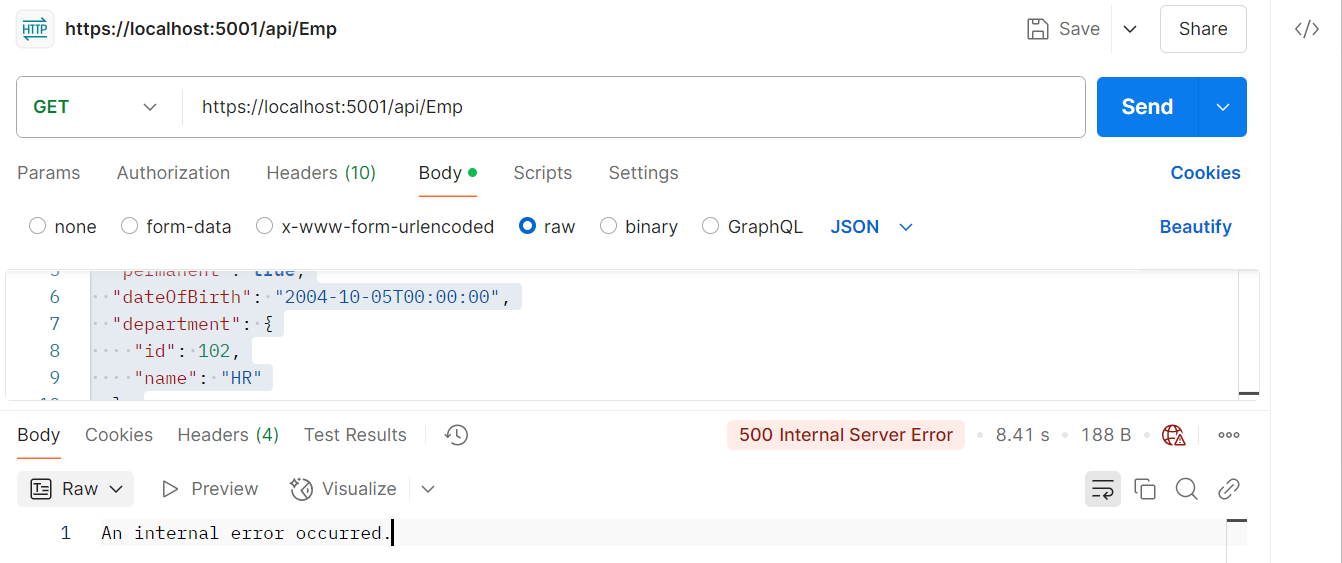
return Ok($"Employee added: {emp.Name}");

}

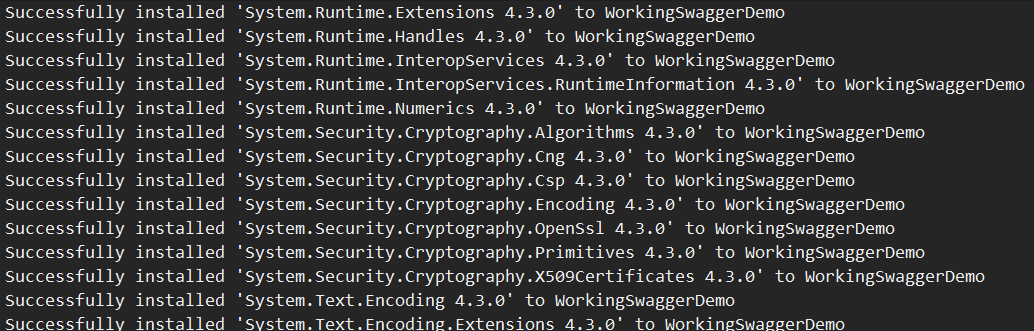
}

}

***It shows the error while doing which proves in below the given Screenshot:***



***Installing Microsoft.AspNetCore.Mvc.WebApiCompatShim***



**Name:- Document-4**

**Solution:-**

***Updated Code for EmployeeController.cs***

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using WorkingSwaggerDemo.Filters;

using WorkingSwaggerDemo.Models;

namespace WorkingSwaggerDemo.Controllers

{

[ApiController]

[Route("api/Emp")]

[ServiceFilter(typeof(CustomAuthFilter))]

[ServiceFilter(typeof(CustomExceptionFilter))]

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 = "Hrishav Ranjan",

Salary = 50000,

Permanent = true,

DateOfBirth = new DateTime(2003, 11, 5),

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

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "C#" },

new Skill { Id = 2, Name = "ASP.NET Core" }

}

}

};

}

[HttpPut("{id}")]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

public ActionResult<Employee> Put(int id, [FromBody] Employee updatedEmp)

{

if (id <= 0)

return BadRequest("Invalid employee id");

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

if (existingEmp == null)

return BadRequest("Invalid employee id");

existingEmp.Name = updatedEmp.Name;

existingEmp.Salary = updatedEmp.Salary;

existingEmp.Permanent = updatedEmp.Permanent;

existingEmp.DateOfBirth = updatedEmp.DateOfBirth;

existingEmp.Department = updatedEmp.Department;

existingEmp.Skills = updatedEmp.Skills;

return Ok(existingEmp);

}

[HttpGet]

[AllowAnonymous]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public ActionResult<List<Employee>> Get()

{

throw new Exception("This is a test exception from GET.");

// return Ok(\_employees); // Use this after testing exception filter

}

[HttpPost]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

public IActionResult Post([FromBody] Employee emp)

{

if (emp == null || string.IsNullOrEmpty(emp.Name))

{

return BadRequest("Invalid Employee data");

}

return Ok($"Employee added: {emp.Name}");

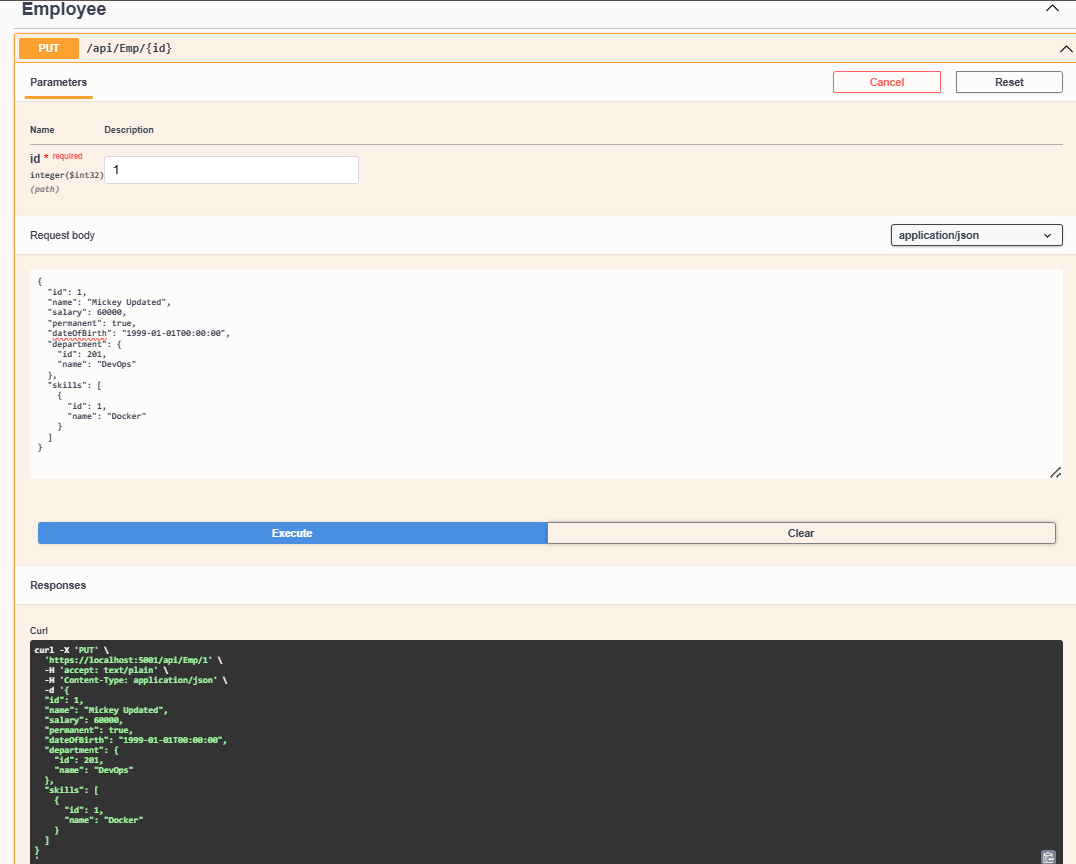
}

}

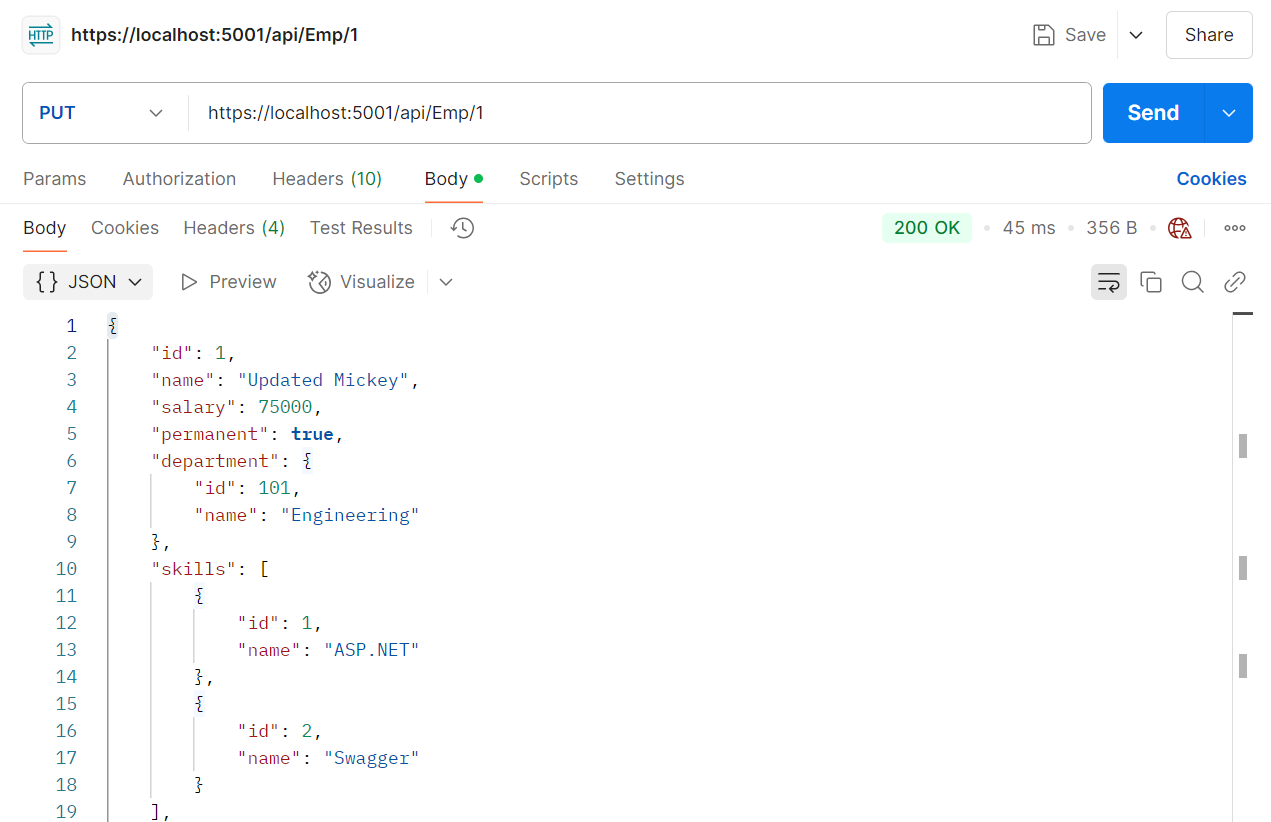
}

***TESTING WITH SWAGGER***

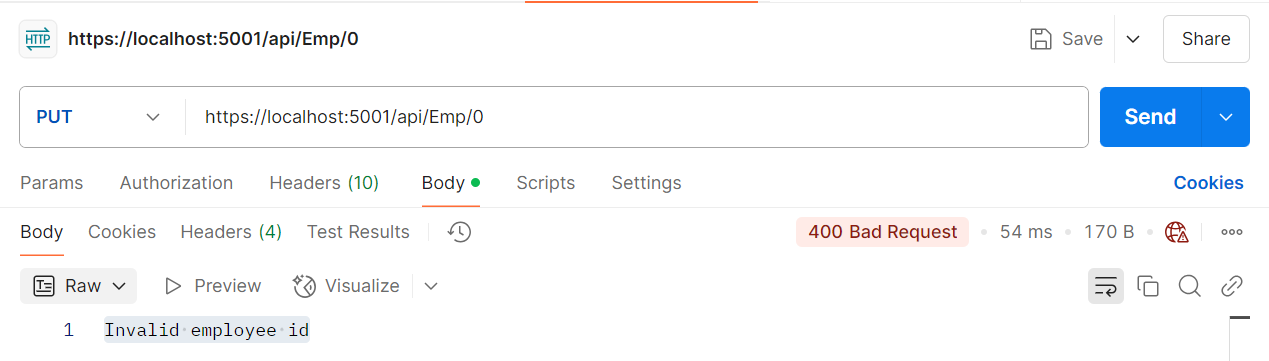
***JSON in the request body***



***After Checking in the Local Host now checking in the Postman and hence below are the results:***



***Check if the id value is lesser than or equal to 0. If true, throw BadRequest action result with the message ‘Invalid employee id’***



***It is showing invalid.***

**Name:- Document-5**

**Solution:-**

## What is CORS?

**Answer:-**

**CORS (Cross-Origin Resource Sharing)** is a security feature in browsers that blocks requests from other origins (like frontend running on http://localhost:3000 accessing API on https://localhost:5001).  
To **enable** access, you need to **enable CORS in your API.**

***Updated Program.cs***

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddCors(options =>

{

options.AddPolicy("AllowAll", policy =>

{

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

});

});

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

string securityKey = "mysuperdupersecretkeyforJWTtoken@123";

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

builder.Services.AddAuthentication(x =>

{

x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

}).AddJwtBearer(options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = "mySystem",

ValidAudience = "myUsers",

IssuerSigningKey = symmetricSecurityKey

};

});

var app = builder.Build();

app.UseSwagger();

app.UseSwaggerUI();

app.UseHttpsRedirection();

app.UseCors("AllowAll");

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

***Create AuthController.cs in Controolers folder***

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 WorkingSwaggerDemo.Controllers

{

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

[ApiController]

[AllowAnonymous]

public class AuthController : ControllerBase

{

[HttpGet]

public IActionResult GetToken()

{

var token = GenerateJSONWebToken(1, "Admin");

return Ok(new { token });

}

private string GenerateJSONWebToken(int userId, string userRole)

{

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretkeyforJWTtoken@123"));

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);

}

}

}

***Updated Code for EmployeeController.cs in Controllers Folder for Authorization:***

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using WorkingSwaggerDemo.Models;

namespace WorkingSwaggerDemo.Controllers

{

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

[ApiController]

[Authorize(Roles = "Admin")]

public class EmployeeController : ControllerBase

{

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

{

new Employee

{

Id = 1,

Name = "Mickey",

Salary = 60000,

Permanent = true,

DateOfBirth = new DateTime(1999, 10, 10),

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

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "ASP.NET" },

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

}

}

};

[HttpGet]

public ActionResult<IEnumerable<Employee>> GetAll()

{

return Ok(employees);

}

[HttpGet("{id}")]

public ActionResult<Employee> GetById(int id)

{

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

if (emp == null) return NotFound("Employee not found");

return Ok(emp);

}

[HttpPut("{id}")]

public ActionResult<Employee> UpdateEmployee(int id, [FromBody] Employee updated)

{

if (id <= 0) return BadRequest("Invalid employee id");

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

if (emp == null) return BadRequest("Invalid employee id");

emp.Name = updated.Name;

emp.Salary = updated.Salary;

emp.Permanent = updated.Permanent;

emp.Department = updated.Department;

emp.Skills = updated.Skills;

emp.DateOfBirth = updated.DateOfBirth;

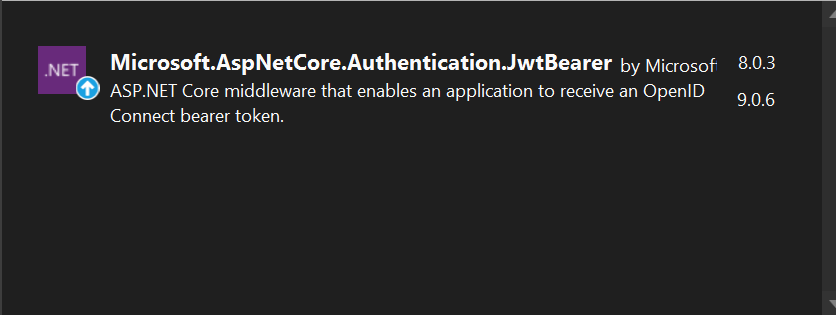
return Ok(emp);

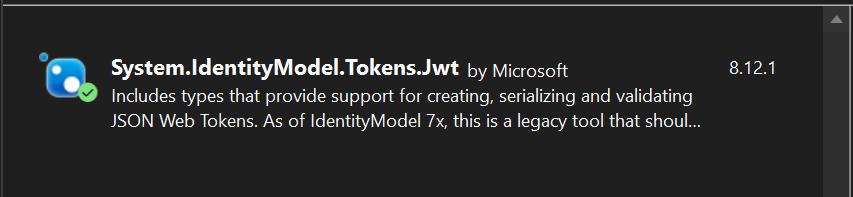
}

}

}

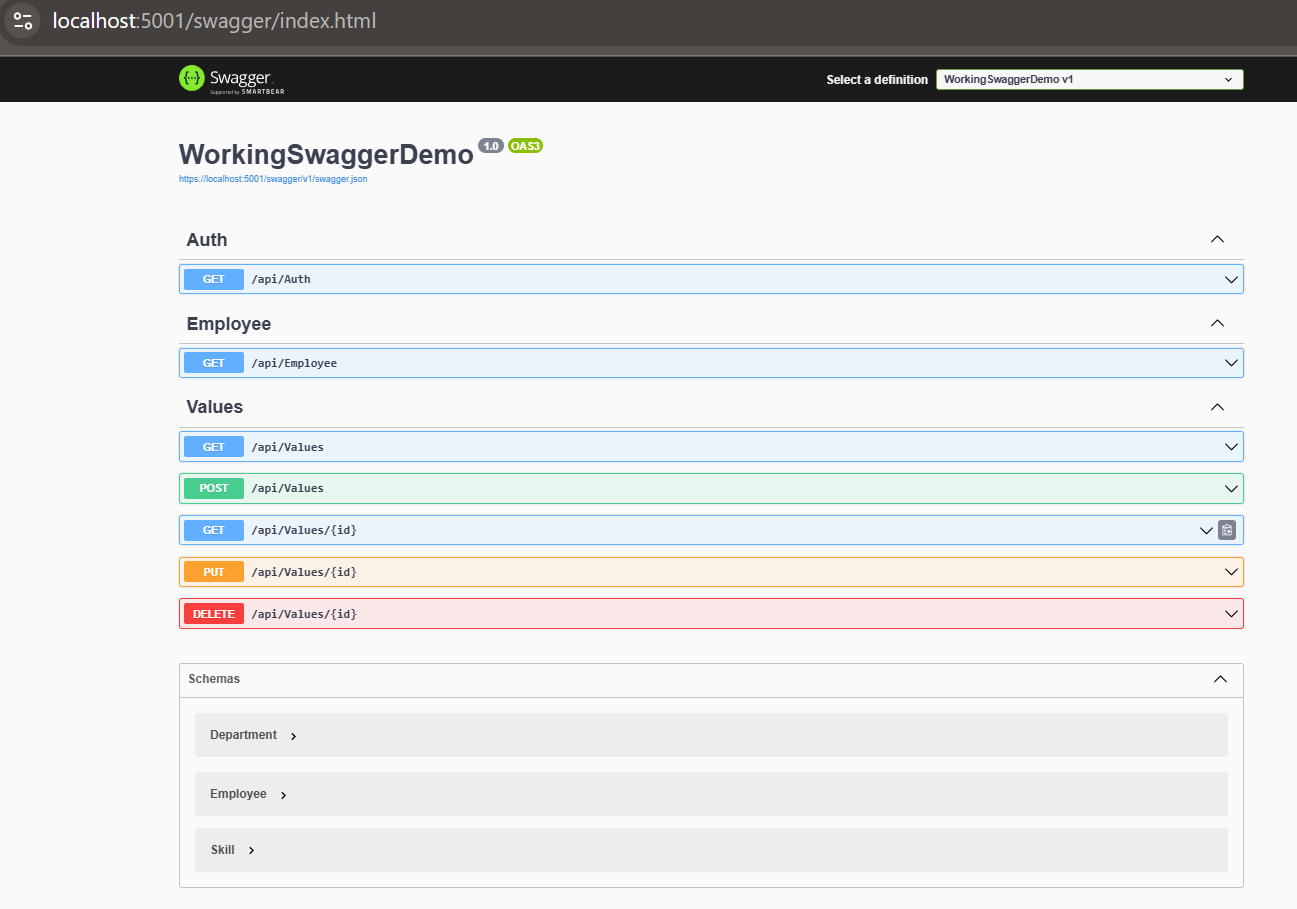
***Added Microsoft.AspNetCore.Authentication.JwtBearer and System.IdentityModel.Tokens.Jwt***





***After doing this all the final view and Testing is Below:***

***View Of Local Host***

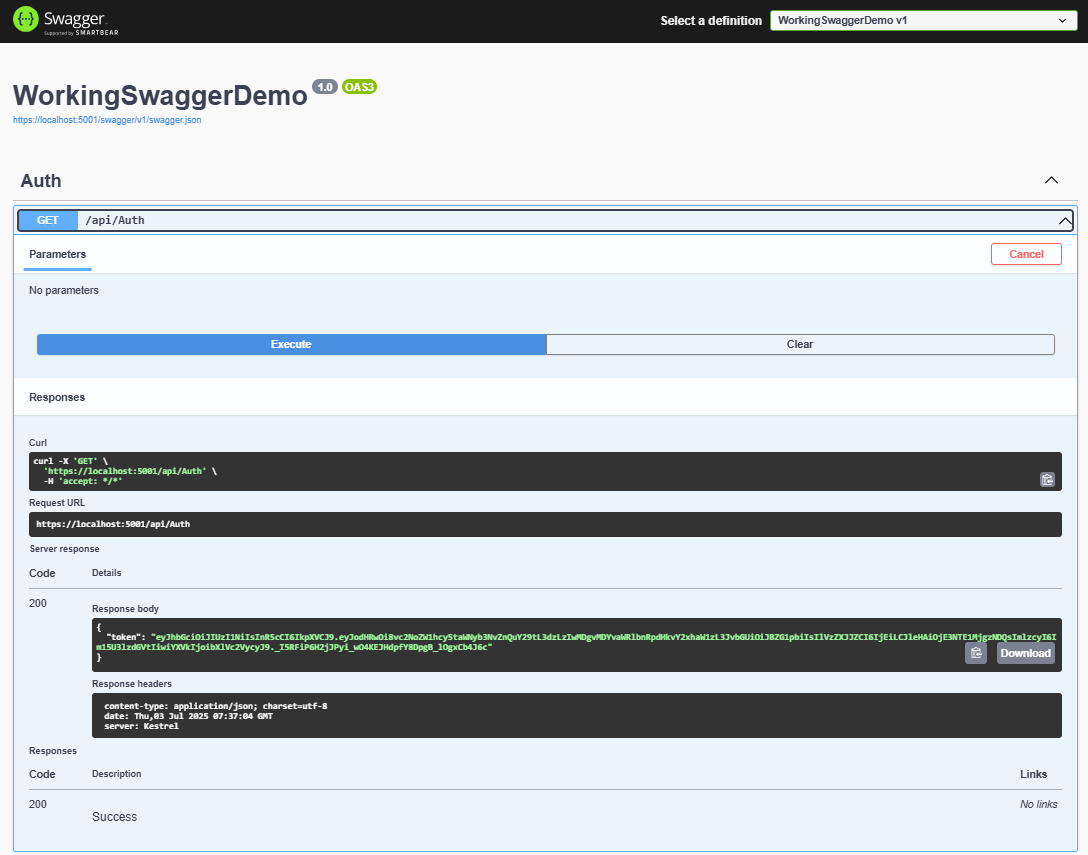


***Use Postman to Test***

***Generated Token for Test***

**eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy5taWNyb3NvZnQuY29tL3dzLzIwMDgvMDYvaWRlbnRpdHkvY2xhaW1zL3JvbGUiOiJBZG1pbiIsIlVzZXJJZCI6IjEiLCJleHAiOjE3NTE1MjgzNDQsImlzcyI6Im15U3lzdGVtIiwiYXVkIjoibXlVc2VycyJ9.\_I5RFiP6H2jJPyi\_wO4KEJHdpfY8DpgB\_lOgxCb4J6c**

***The View of local host***



***Now Final Checking in Postman and result is below:***

