**Cognizant - DN 4.0 I Deep Skilling**

**WEEK-3**

**ASP.NET Core 8.0 Web API**

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

**Question:**

Create a .Net core web application with API template. Use the option to create controller with Read Write permissions. Notice the ValuesController creation with Action methods corresponding to the Action verbs.

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

****Lab 2 –****Demonstrate Swagger installation, usage of Postman tool and usage of Route in Http requests

**Question:**

* Demonstrate Swagger installation to WebAPI and WebAPI listing on browser
  + Nuget package to download Swashbuckle.AspNetCore, Usage of ProducesResponseType to Web API method, AddSwaggerGen, UseSwaggerUI
* Demonstrate the usage of Postman tool to hit WebAPI methods
  + Structure in Postman tool, Headers with Authorization, Body as JSON, Option to choose the type of request, Request collection and how to add a new request in the collection, Tabs in the center pane that corresponds to the request
* Demonstrate the usage of Route and Explain Name attribute in Http requests
  + Importance of user friendly name to action method, Explain the usage of ActionName to have more than 1 method with the same Action verbDemonstrate creation of a simple WebAPI - With Read, Write actions

**SOLUTION :**

**CODE -:**

1. **Program.cs**

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

// ? Replace AddOpenApi with SwaggerGen

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(options =>

{

options.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")

}

});

});

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

// ? Enable Swagger in Development mode

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

1. **Models/Products.cs**

namespace ASPCoreRestAPI.Models

{

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

public string Description { get; set; }

public string Price { get; set; }

}

}

1. **Controllers/ProductsController.cs**

using ASPCoreRestAPI.Models;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

using System.Linq;

namespace ASPCoreRestAPI.Controllers

{

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

[ApiController]

public class ProductsController : ControllerBase

{

// In-memory static list

private static List<Product> products = new List<Product>

{

new Product { Id = 1, Name = "Burger", Description = "Beef patty with cheese", Price = "150" },

new Product { Id = 2, Name = "Pizza", Description = "Cheesy pepperoni", Price = "250" },

new Product { Id = 3, Name = "Pasta", Description = "White sauce pasta", Price = "180" }

};

// GET: api/products

[HttpGet]

public ActionResult<IEnumerable<Product>> Get()

{

return Ok(products);

}

// GET: api/products/2

[HttpGet("{id}")]

public ActionResult<Product> Get(int id)

{

var product = products.FirstOrDefault(p => p.Id == id);

if (product == null)

return NotFound("Product not found.");

return Ok(product);

}

// POST: api/products

[HttpPost]

public ActionResult<Product> Post([FromBody] Product product)

{

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

return BadRequest("Invalid product data.");

product.Id = products.Max(p => p.Id) + 1;

products.Add(product);

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

}

// PUT: api/products/2

[HttpPut("{id}")]

public ActionResult<Product> Put(int id, [FromBody] Product updatedProduct)

{

var product = products.FirstOrDefault(p => p.Id == id);

if (product == null)

return NotFound("Product not found.");

product.Name = updatedProduct.Name;

product.Description = updatedProduct.Description;

product.Price = updatedProduct.Price;

return Ok(product);

}

// DELETE: api/products/2

[HttpDelete("{id}")]

public ActionResult Delete(int id)

{

var product = products.FirstOrDefault(p => p.Id == id);

if (product == null)

return NotFound("Product not found.");

products.Remove(product);

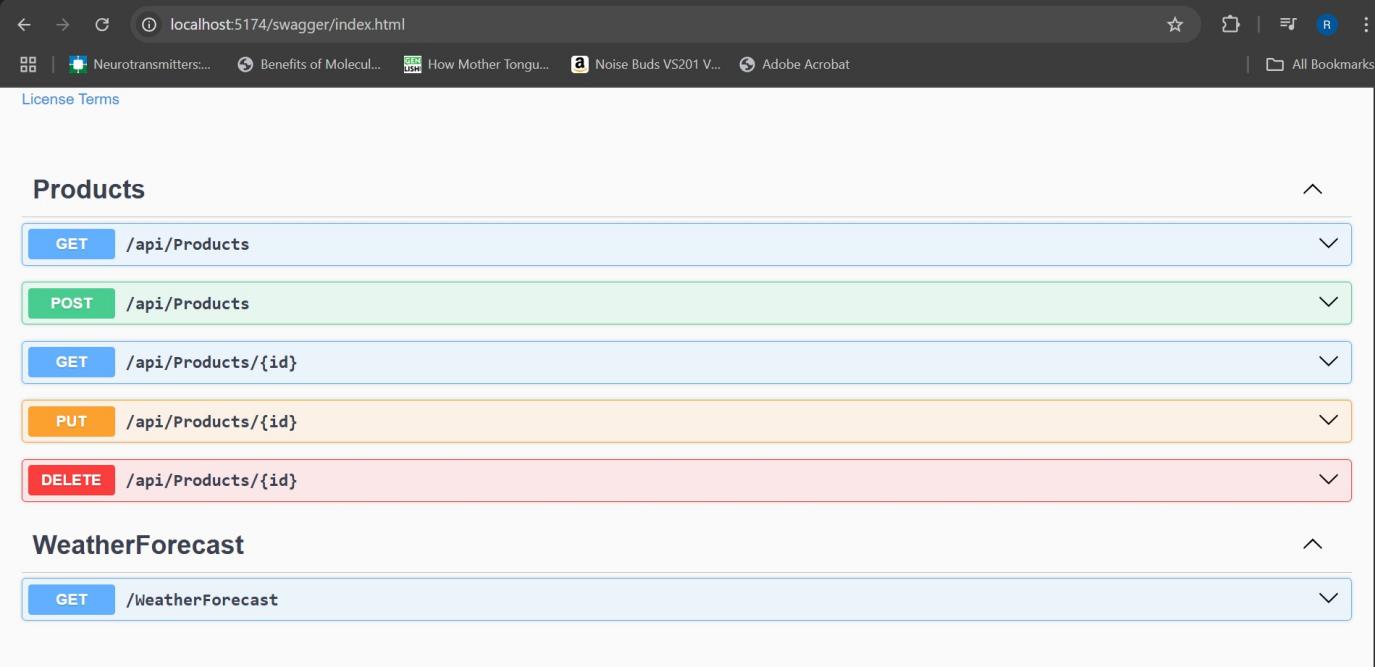
return Ok("Product deleted successfully.");

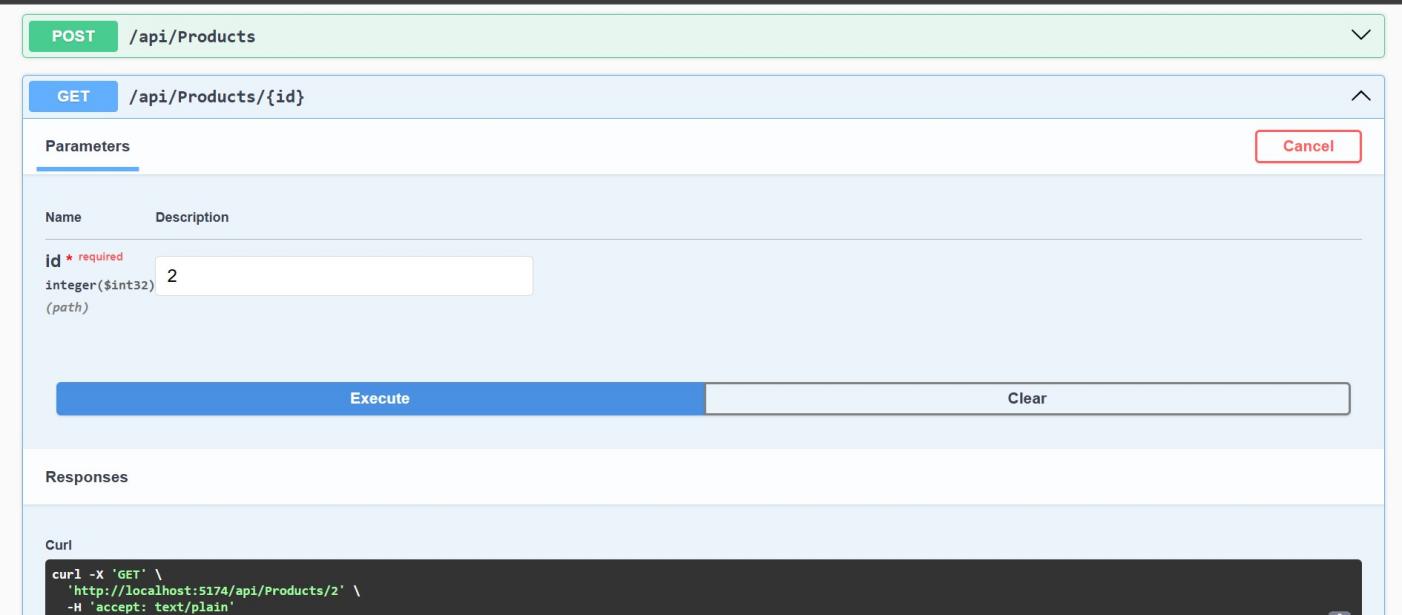
}

}

}

**OUTPUT -:**

****

****

****Lab 3 -**** Demonstrate Action method, usage of FromBody Attribute , Custom Filter

* Demonstrate creation of an Action method to return list of custom class entity
  + Model class creation, Use AllowAnonymous attribute, Use HttpGet action method
* Explain the usage of FromBody attribute
  + Read the model object from request, other than the query string parameter
* Demonstrate Custom filter
  + Usage of ActionFilterAttribute, OnActionExecuting method to intercept the request, Create filter for Custom exception - Need to install Microsoft.AspNetCore.Mvc.WebApiCompatShim package

**SOLUTION :**

**CODE -:**

1. **Models/Department.cs**

namespace ASPCoreRestAPI.Models

{

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

}

}

1. **Models/Skill.cs**

namespace ASPCoreRestAPI.Models

{

public class Skill

{

public int Id { get; set; }

public string Name { get; set; }

}

}

1. **Models/Employee.cs**

namespace ASPCoreRestAPI.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; }

}

}

1. **Controllers/EmployeeController.cs**

using ASPCoreRestAPI.Models;

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

namespace ASPCoreRestAPI.Controllers

{

[ApiController]

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

[ServiceFilter(typeof(Filters.CustomAuthFilter))] // Apply Authorization Filter

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 = "Riya",

Salary = 50000,

Permanent = true,

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

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "Communication" },

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

},

DateOfBirth = new DateTime(1999, 7, 10)

}

};

}

[HttpGet("GetStandard")]

[ProducesResponseType(StatusCodes.Status200OK)]

public ActionResult<List<Employee>> GetStandard()

{

return Ok(employees);

}

[HttpGet]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public ActionResult Get()

{

throw new Exception("Test exception from GET");

}

[HttpPost]

public ActionResult Post([FromBody] Employee employee)

{

if (employee == null)

return BadRequest("Invalid data");

return Ok($"Employee {employee.Name} received.");

}

}

}

1. **Filters/CustomAuthFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System.Linq;

namespace ASPCoreRestAPI.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

var hasHeader = context.HttpContext.Request.Headers.TryGetValue("Authorization", out var token);

if (!hasHeader)

{

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");

return;

}

base.OnActionExecuting(context);

}

}

}

1. **Filters/CustomExceptionFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System;

using System.IO;

namespace ASPCoreRestAPI.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

var exceptionMessage = $"[{DateTime.Now}] Exception: {context.Exception.Message}";

File.AppendAllText("ExceptionLog.txt", exceptionMessage + Environment.NewLine);

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

{

StatusCode = 500

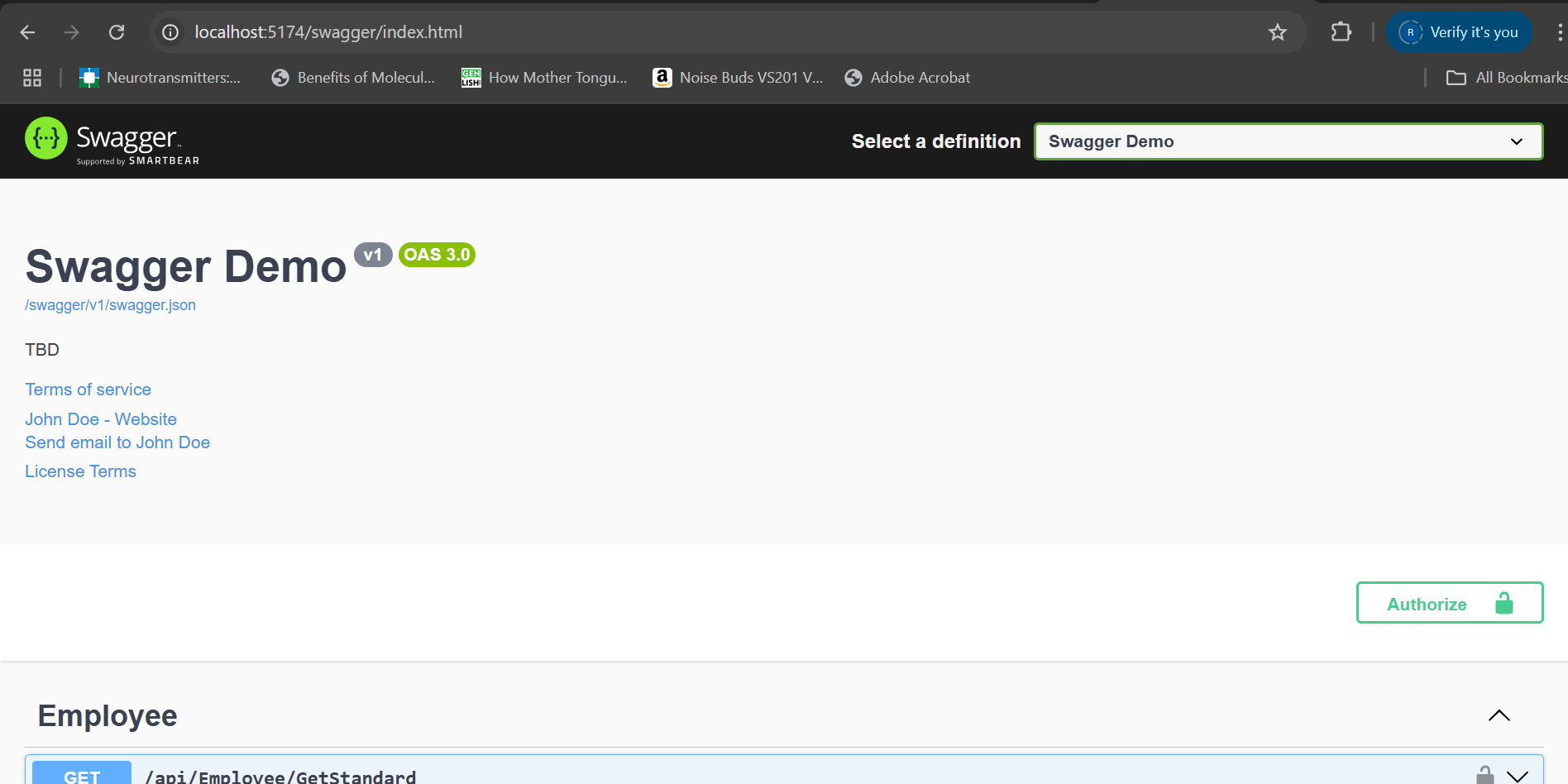
};

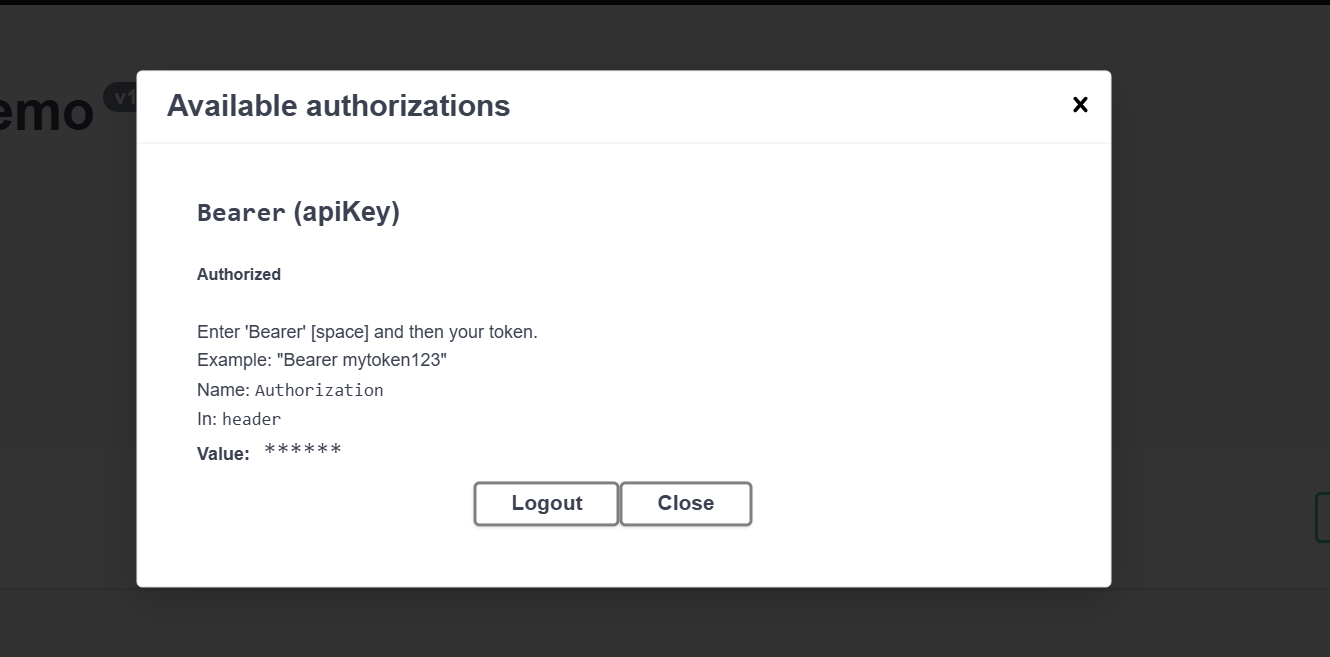
}

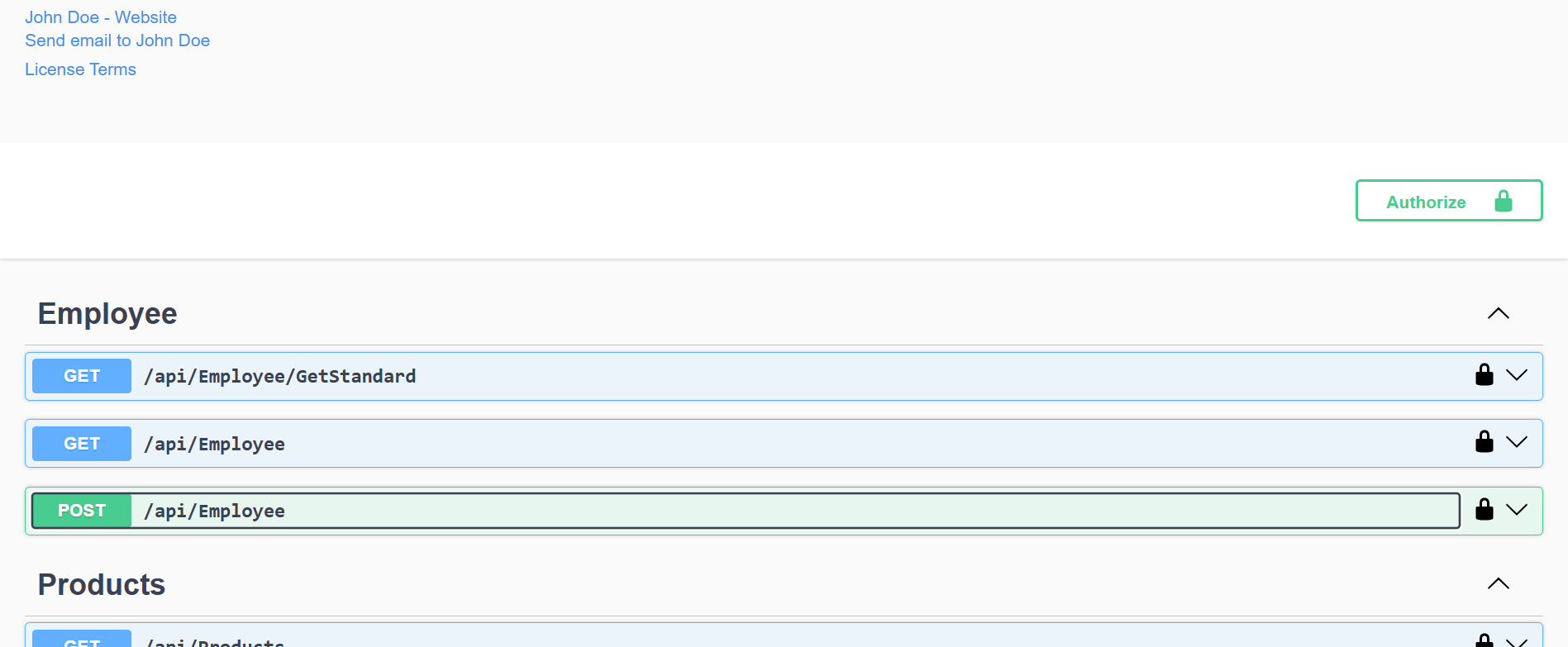
}

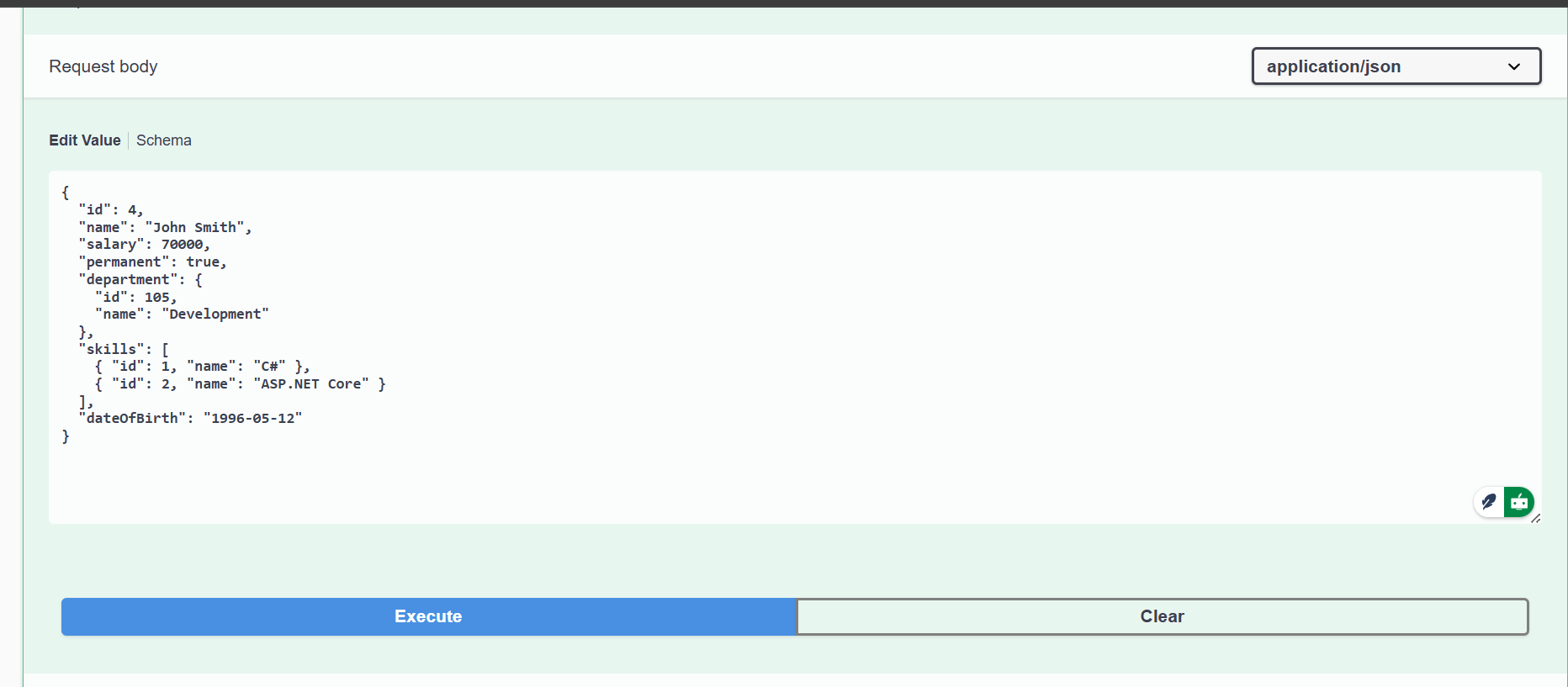
}

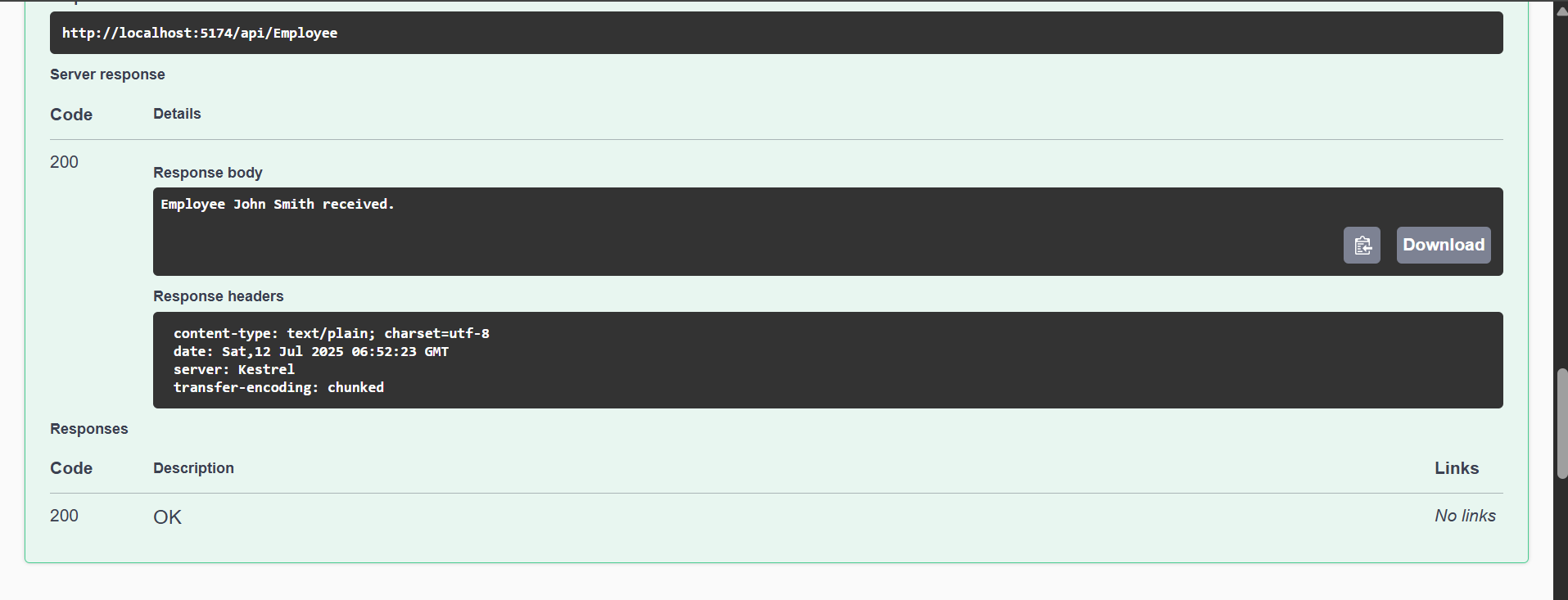
**OUTPUT -:**











****Lab 4 –**** Demonstrate creation of an Action method to perform data create, update & delete operation

1. **Web Api CRUD operation**

Update Employee data as per the input thru Web API PUT action method call

Employee information has to be updated based on the user input. Use Swagger tool to invoke the action method mapped with Http PUT action verb to update an employee data.

Modify the action method to return Employee data thru ActionResult.

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

If the value is greater than 0 but not available in the list of employee ids that is there in the hardcoded list of employees, throw BadRequest action result with the same message as stated above.

If the id value is valid, use the JSON data from the input body and update the hardcoded list. Filter the employee list data for the input id and return that as the output.

**SOLUTION :**

**CODE -:**

**Controllers/EmployeeController.cs**

using ASPCoreRestAPI.Models;

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

namespace ASPCoreRestAPI.Controllers

{

[ApiController]

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

[ServiceFilter(typeof(Filters.CustomAuthFilter))] // Apply Authorization Filter

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 = "Riya",

Salary = 50000,

Permanent = true,

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

Skills = new List<Skill>

{

new Skill { Id = 1, Name = "Communication" },

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

},

DateOfBirth = new DateTime(1999, 7, 10)

}

};

}

[HttpGet("GetStandard")]

[ProducesResponseType(StatusCodes.Status200OK)]

public ActionResult<List<Employee>> GetStandard()

{

return Ok(employees);

}

[HttpGet]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public ActionResult Get()

{

throw new Exception("Test exception from GET");

}

[HttpPost]

public ActionResult Post([FromBody] Employee employee)

{

if (employee == null)

return BadRequest("Invalid data");

return Ok($"Employee {employee.Name} received.");

}

[HttpPut("{id}")]

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

{

if (id <= 0)

return BadRequest("Invalid employee id");

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

if (employee == null)

return BadRequest("Invalid employee id");

// Update employee fields from input

employee.Name = updatedEmployee.Name;

employee.Salary = updatedEmployee.Salary;

employee.Permanent = updatedEmployee.Permanent;

employee.Department = updatedEmployee.Department;

employee.Skills = updatedEmployee.Skills;

employee.DateOfBirth = updatedEmployee.DateOfBirth;

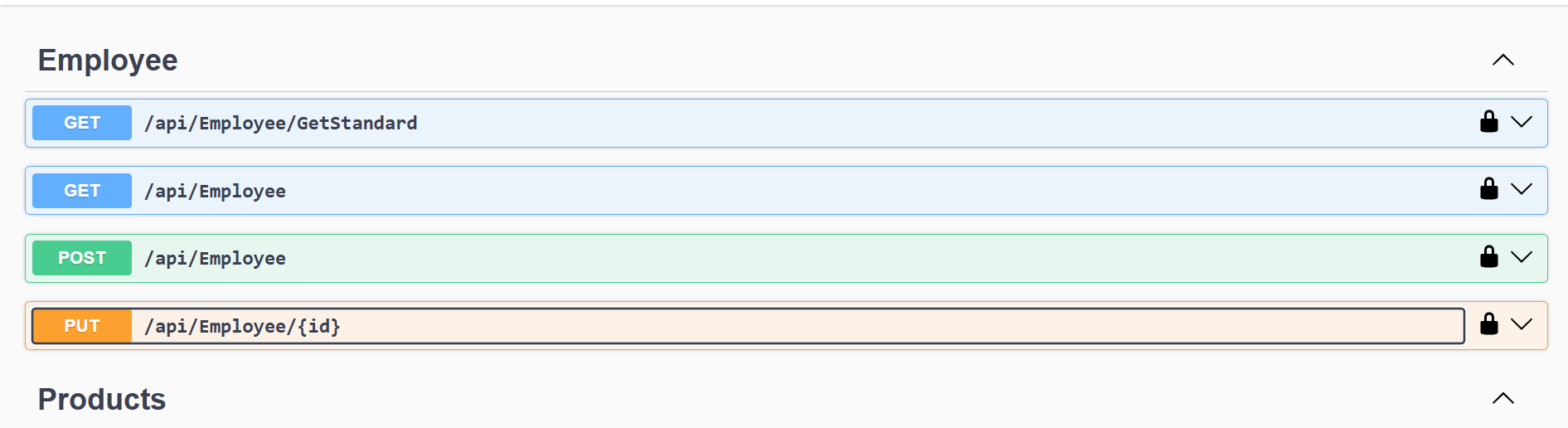
return Ok(employee);

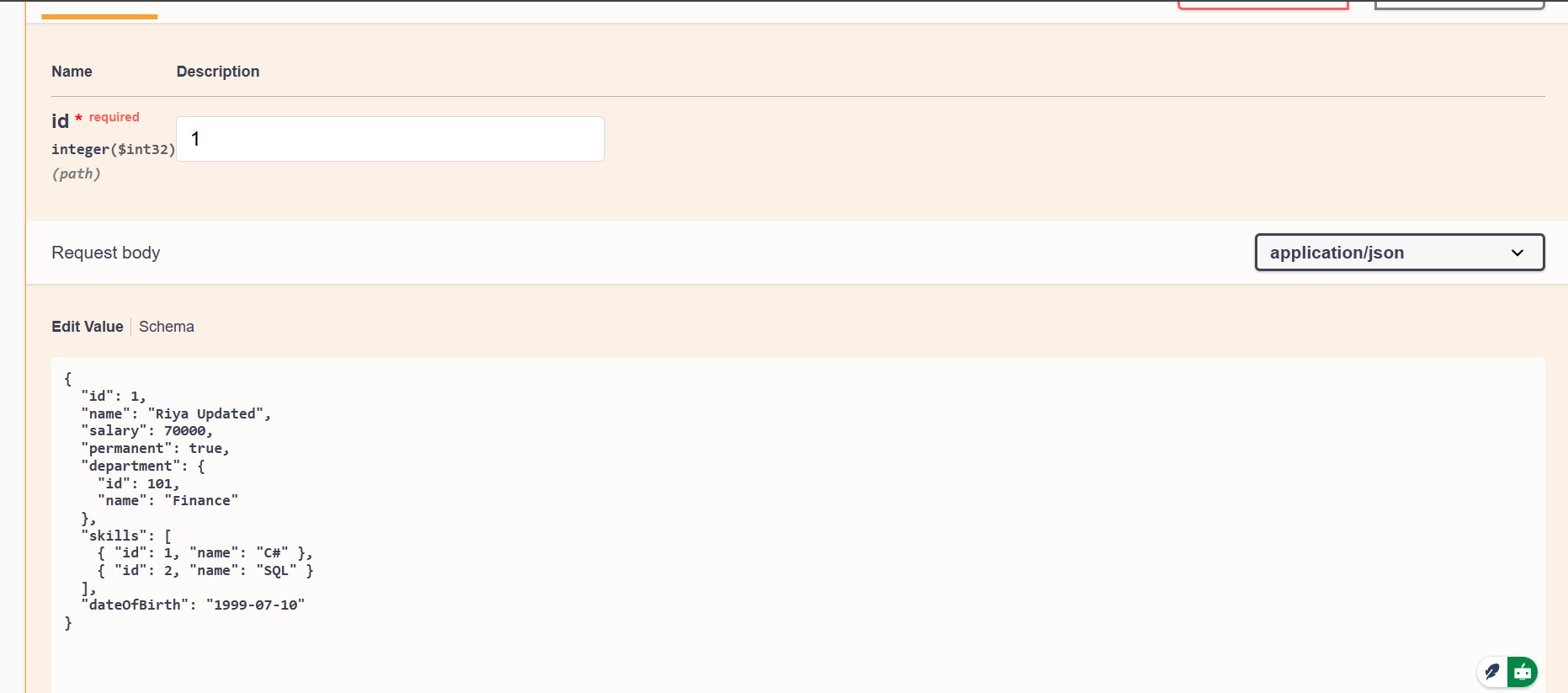
}

}

}

**OUTPUT -:**







****Lab 5 –**** Demonstrate creation of an Action method to perform data create, update & delete operation

**Objectives:**

* Explain CORS enablement for Web API access for local application
  + What is CORS?, How to enable CORS thru Startup.cs, Install Cors nuget package to Web API application
* Demonstrate security in WebAPI
  + Bearer and Jwt token authentication, Use Authorize attribute & send roles in Jwt token, Setting in Startup.cs for AddAuthentication and AddJwtBearer with validation attributes, UseAuthentication, AllowAnonymous to AuthController to generate token, Claims

1. **Controllers/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 ASPCoreRestAPI.Controllers

{

[ApiController]

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

[AllowAnonymous] // Allow unauthenticated access

public class AuthController : ControllerBase

{

[HttpGet("GetToken")]

public IActionResult GetToken()

{

var token = GenerateJSONWebToken(101, "Admin"); // or "POC"

return Ok(new { token });

}

private string GenerateJSONWebToken(int userId, string userRole)

{

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

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), // 🔁 Set to 2 minutes to test expiration

signingCredentials: credentials);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

}

1. **Filters/CustomAuthFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System.Linq;

namespace ASPCoreRestAPI.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

var hasHeader = context.HttpContext.Request.Headers.TryGetValue("Authorization", out var token);

if (!hasHeader)

{

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");

return;

}

base.OnActionExecuting(context);

}

}

}

1. **Program.cs**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

// 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

};

});

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

c.RoutePrefix = "swagger"; // this ensures Swagger UI is at /swagger

});

}

app.UseHttpsRedirection();

// Enable authentication before authorization

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

