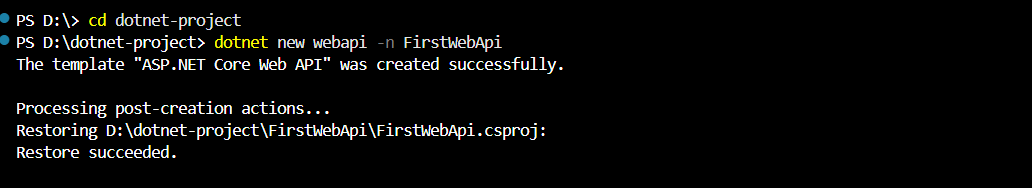
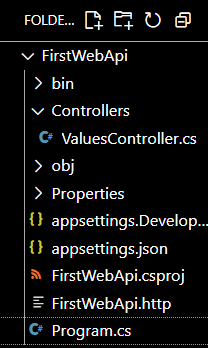
WEEK – 4 SOLUTION ( ASP.NET CORE WEB API )

# HANDS ON EXERCISE SOLUTION - 1

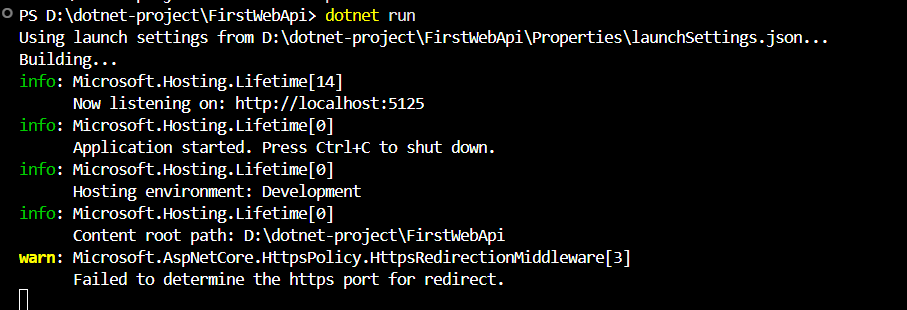
1. Creating a new webAPI project

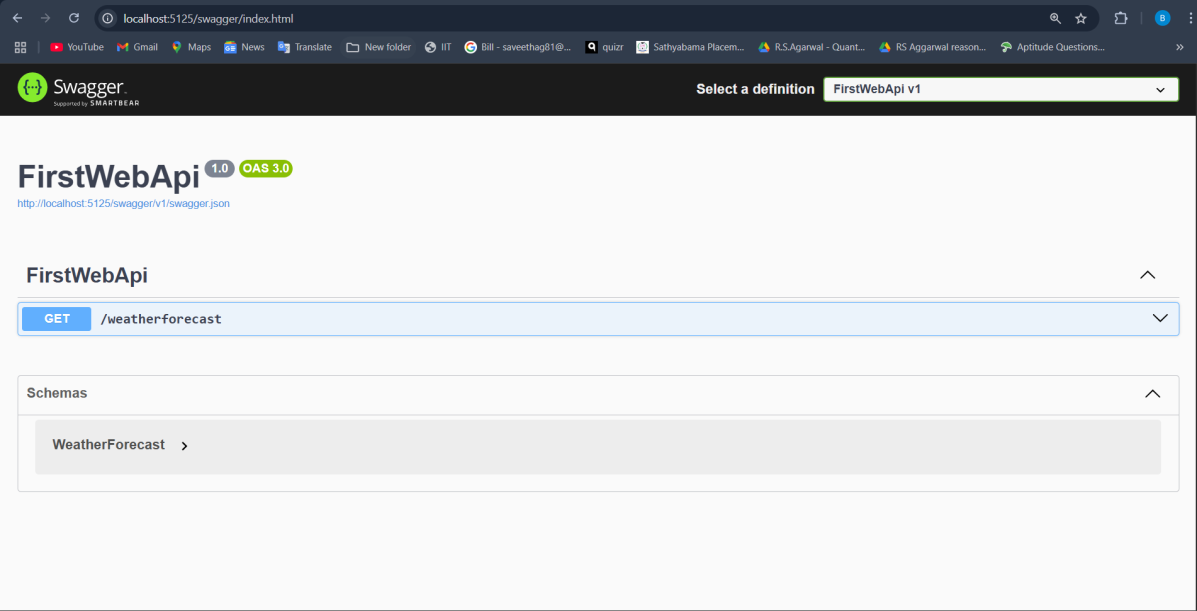


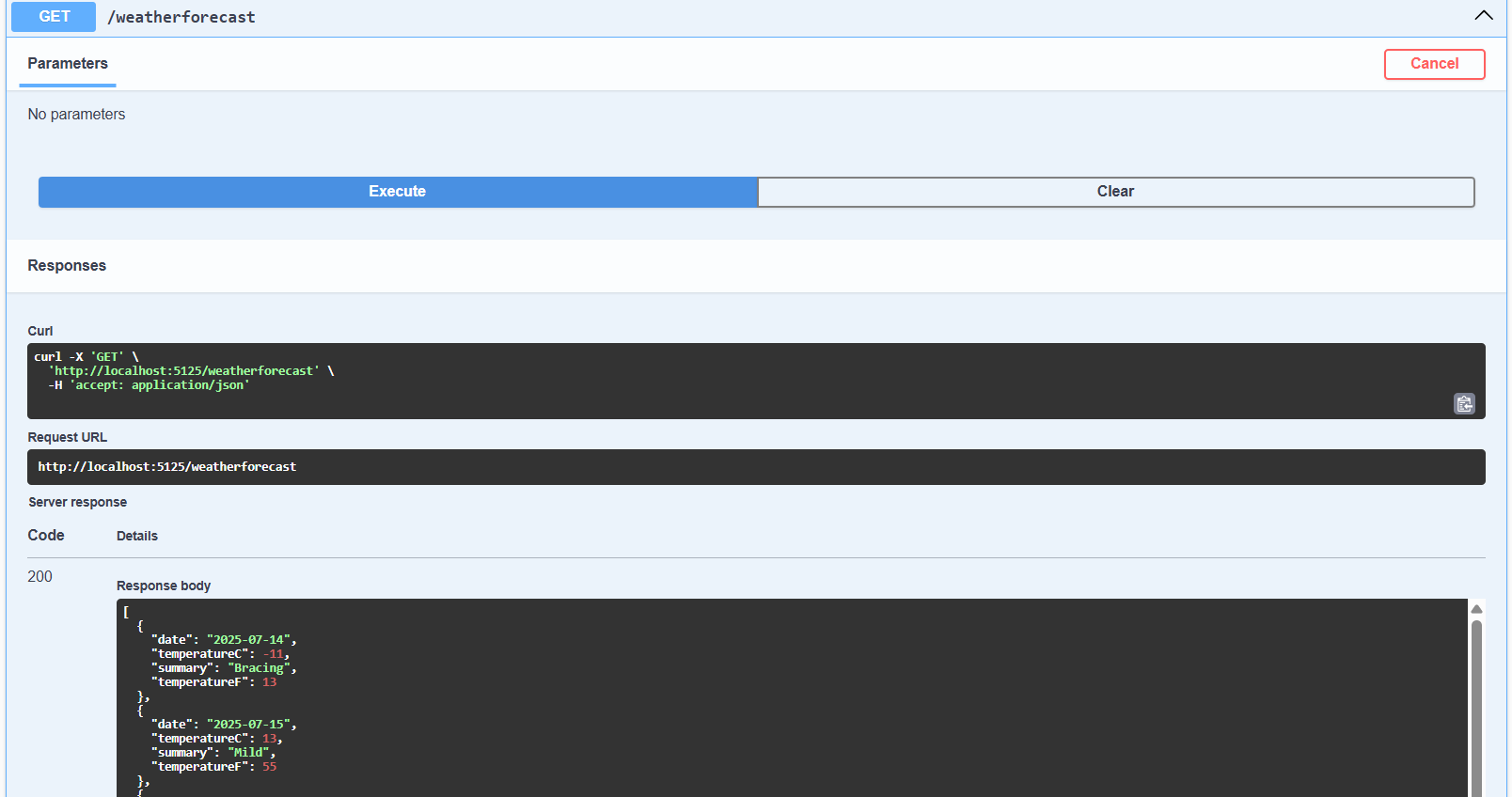
1. Folder creation and update after build



1. Running the web API project



1. Swagger application to check if created API exisits
2. Checking if the desired value is return successfully after the test



## CONTROLLERS / VALUECONTROL.CS

using Microsoft.AspNetCore.Mvc; using System.Collections.Generic;

namespace FirstWebApi.Controllers

***{***

[ApiController]

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

public class ValuesController : ControllerBase

***{***

private static List<string> values = new List<string> { "value1", "value2" };

[HttpGet]

public ActionResult<IEnumerable<string>> Get()

***{***

return values;

***}***

[HttpGet("{id}")]

public ActionResult<string> Get(int id)

***{***

if (id < 0 || id >= values.Count) return NotFound();

return values[id];

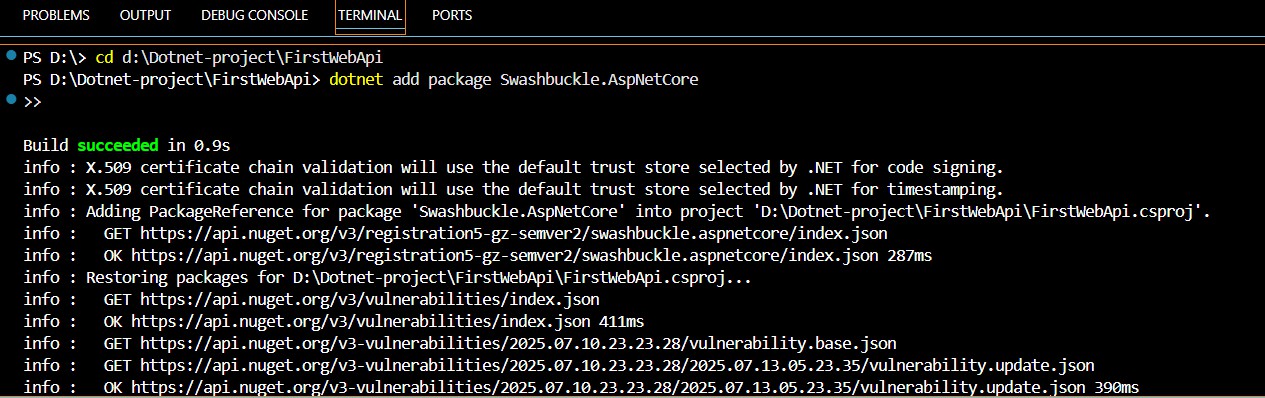
***}***

***}***

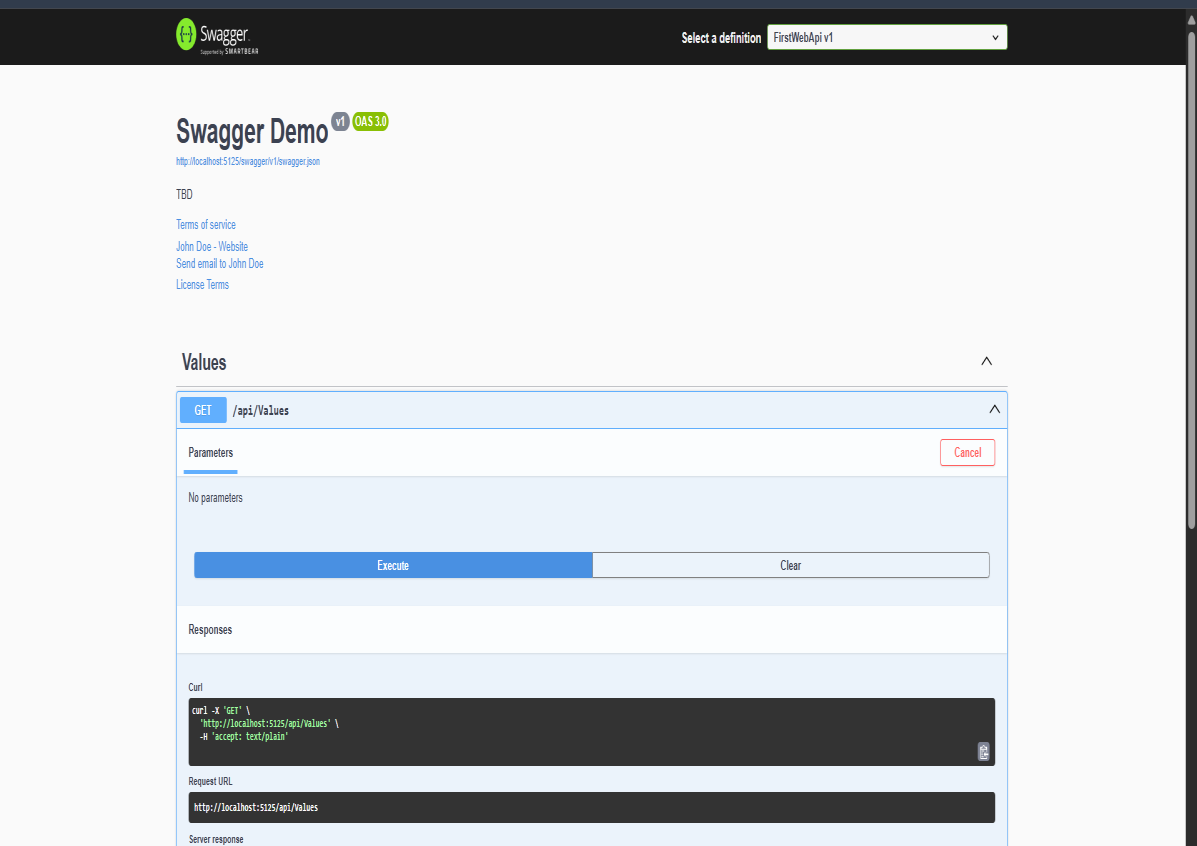
***}***

# HANDS-ON EXERCISE-2 SOLUTION

1. Importing or adding the swaggers library to the project

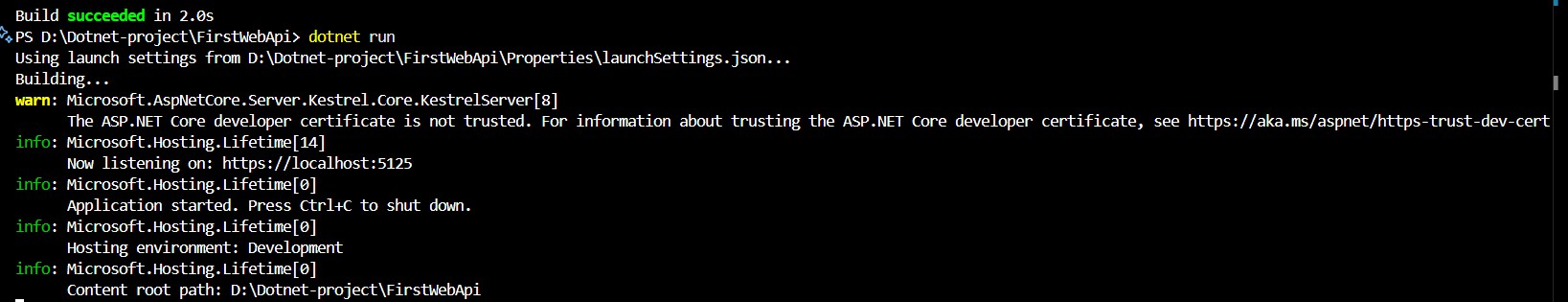


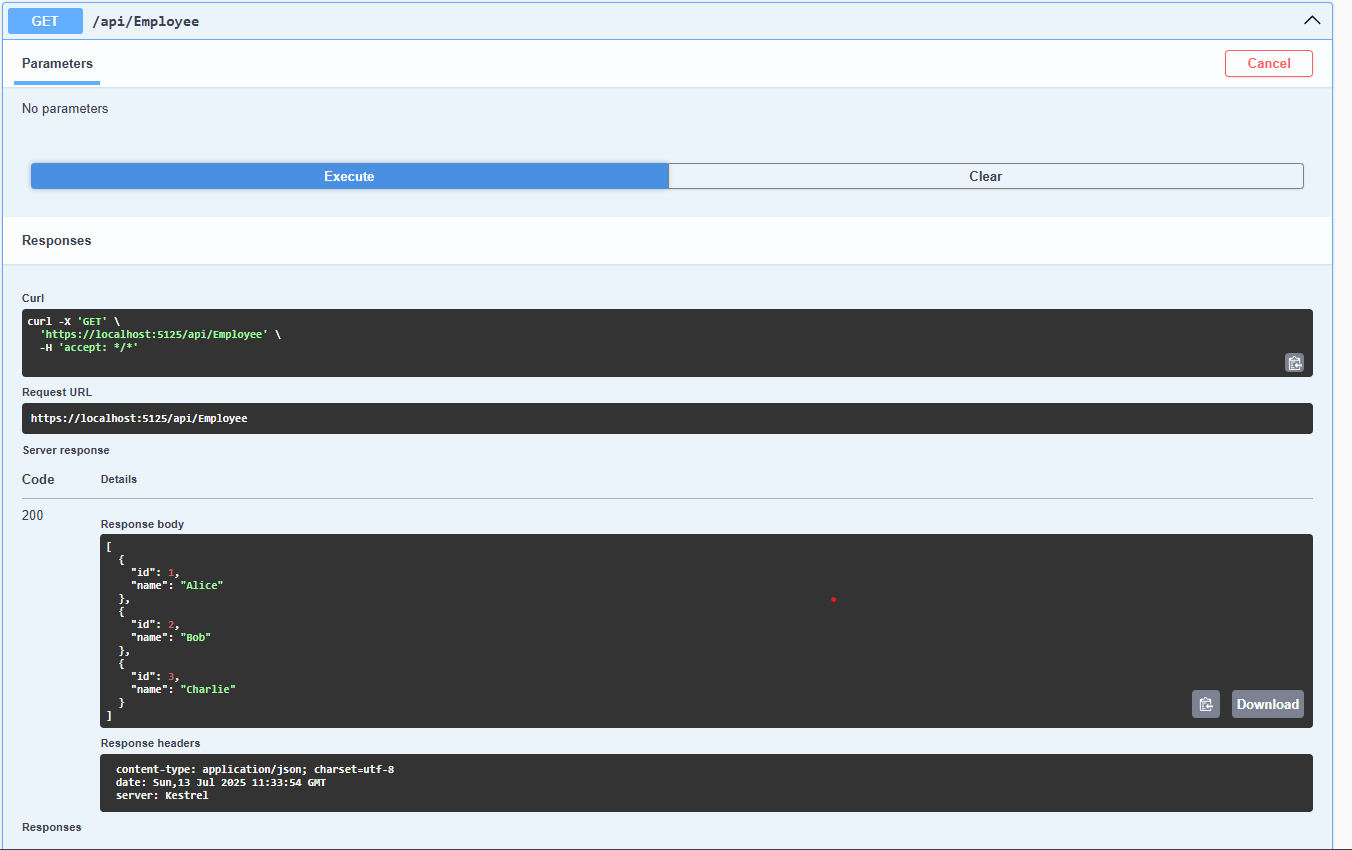
1. After adding the configuration services



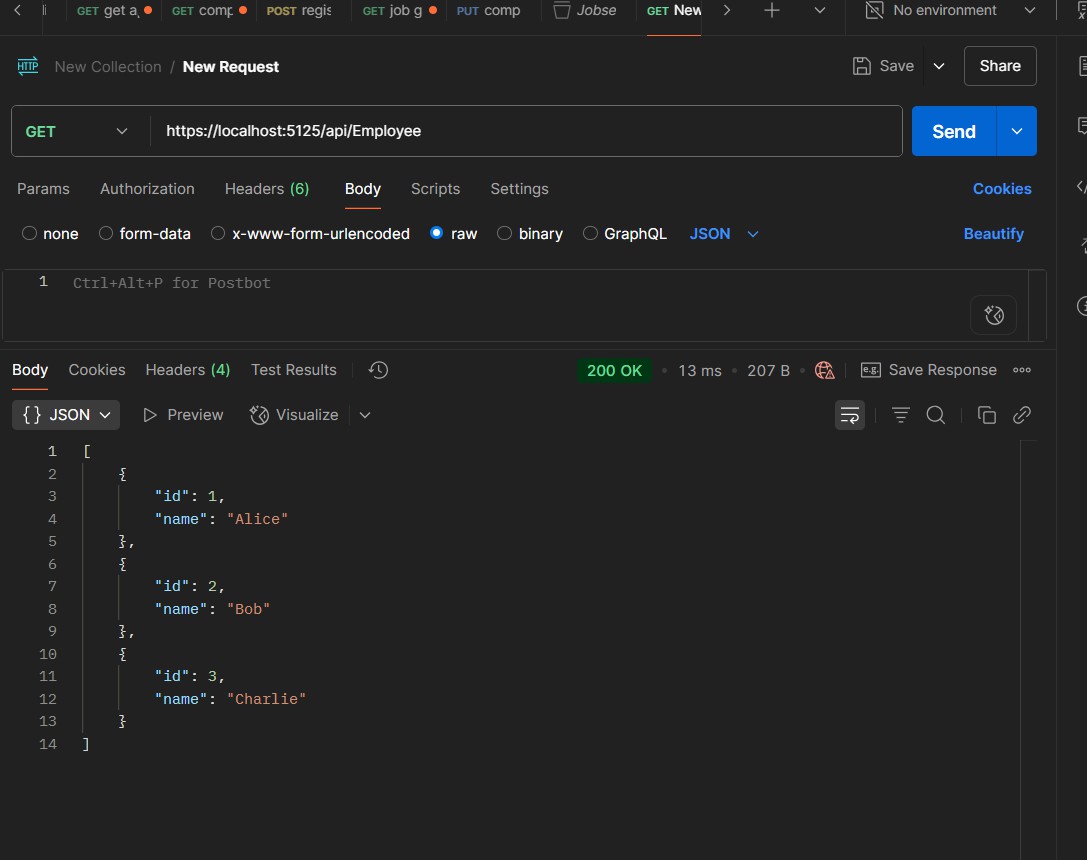
1. Changing the local host handling from HTTP to HTTPS

Through launchsettings.json in properties folder we change the application URL



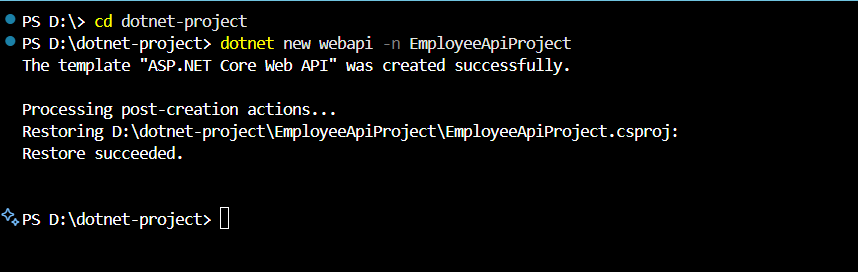


1. Postman testing

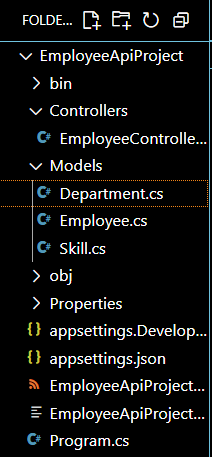


# HANDS-ON EXERCISE: 3 SOLUTION

1. Project creation



1. Employee API working folder



## MODELS/ EMPLOYEE.CS

using System;

using System.Collections.Generic;

namespace EmployeeApiProject.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; }

***}***

***}***

## MODELS/ DEPARTMENT.CS

namespace EmployeeApiProject.Models

***{***

public class Department

***{***

public int Id { get; set; }

public string Name { get; set; }

***}***

***}***

Models/Skills

namespace EmployeeApiProject.Models

***{***

public class Skill

***{***

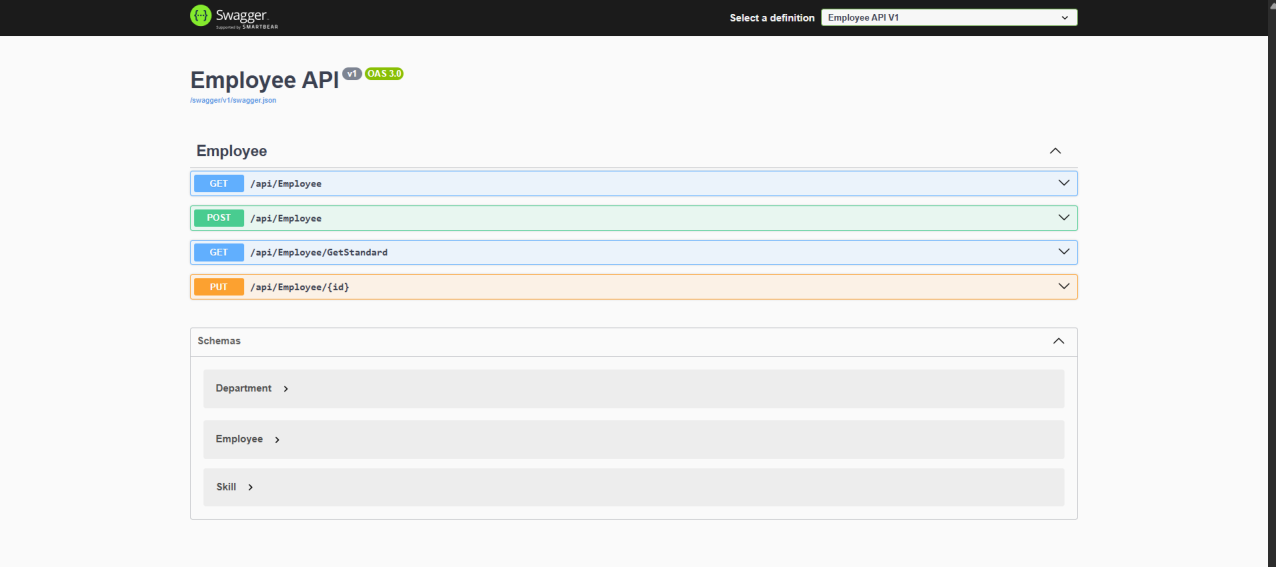
public int Id { get; set; }

public string Name { get; set; }

***}***

***}***

1. ***Viewing the employee API through Swagger***



Employee Controller

using Microsoft.AspNetCore.Mvc; using EmployeeApiProject.Models;

namespace EmployeeApiProject.Controllers

***{***

[ApiController]

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

public class EmployeeController : ControllerBase

***{***

private static List<Employee> \_employees;

public EmployeeController()

***{***

if (\_employees == null)

***{***

\_employees = GetStandardEmployeeList();

***}***

***}***

private List<Employee> GetStandardEmployeeList()

***{***

return new List<Employee>

***{***

new Employee

***{***

Id = 1,

Name = "Alice Johnson", Salary = 60000, Permanent = true,

Department = new Department { Id = 1, Name = "HR" }, Skills = new List<Skill>

***{***

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

new Skill { Id = 2, Name = "Employee Relations" }

***},***

DateOfBirth = new DateTime(1985, 10, 12)

***},***

new Employee

***{***

Id = 2,

Name = "Bob Smith", Salary = 80000, Permanent = false,

Department = new Department { Id = 2, Name = "IT" }, Skills = new List<Skill>

***{***

new Skill { Id = 3, Name = "C#" },

new Skill { Id = 4, Name = "ASP.NET" }

***},***

DateOfBirth = new DateTime(1990, 5, 25)

***}***

***};***

***}***

[HttpGet]

[ProducesResponseType(StatusCodes.Status200OK)] public ActionResult<List<Employee>> Get()

***{***

return Ok(\_employees);

***}***

[HttpGet("GetStandard")]

public ActionResult<Employee> GetStandard()

***{***

var employee = \_employees.FirstOrDefault(); if (employee == null)

return NotFound(); return Ok(employee);

***}***

[HttpPost]

public IActionResult Post(Employee employee)

***{***

\_employees.Add(employee);

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

***}***

[HttpPut("{id}")]

public IActionResult Put(int id, Employee updatedEmployee)

***{***

var existingEmployee = \_employees.FirstOrDefault(e => e.Id == id); if (existingEmployee == null)

return NotFound();

existingEmployee.Name = updatedEmployee.Name; existingEmployee.Salary = updatedEmployee.Salary; existingEmployee.Permanent = updatedEmployee.Permanent; existingEmployee.Department = updatedEmployee.Department; existingEmployee.Skills = updatedEmployee.Skills; existingEmployee.DateOfBirth = updatedEmployee.DateOfBirth;

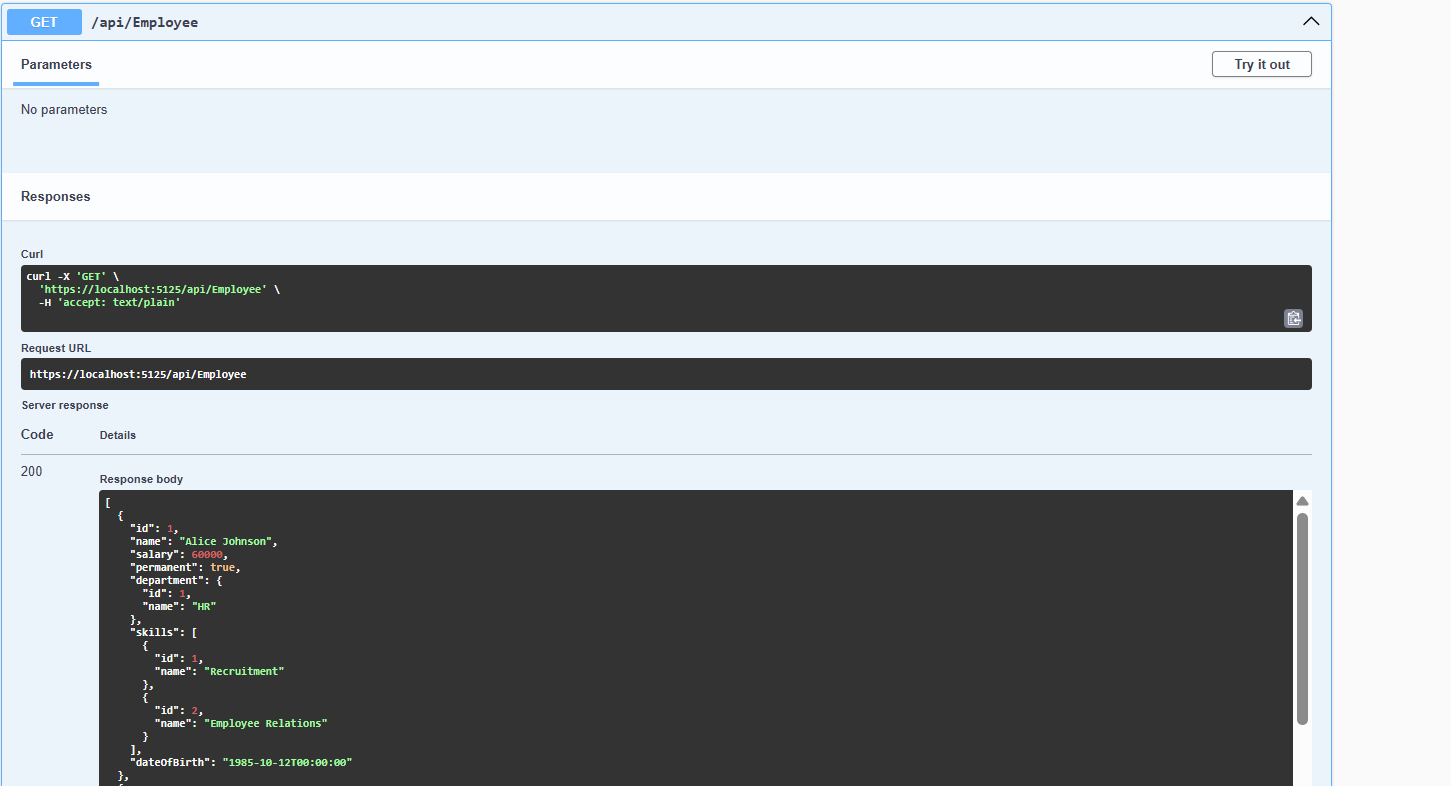
return NoContent();

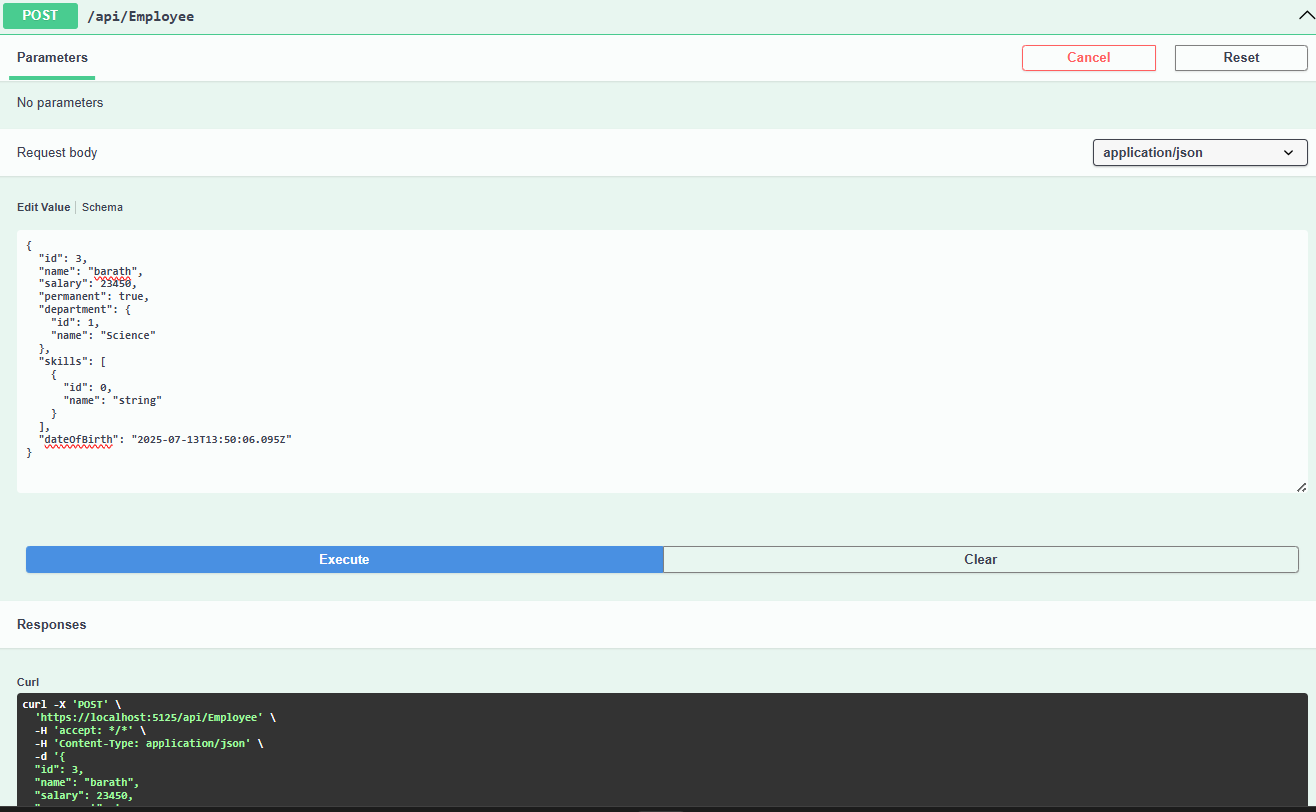
***}***

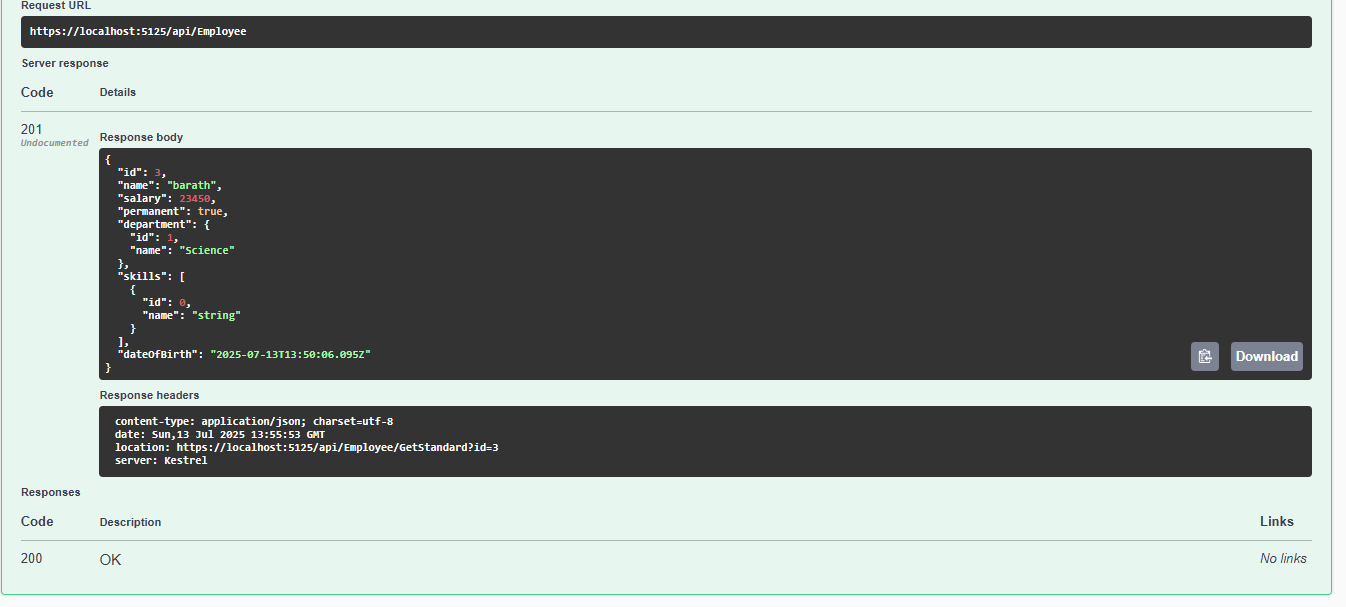
***}***

***}***

Output :







## IMPLEMENTING PUBLIC ACTIONRESULT<EMPLOYEE> GETSTANDRAD()

public ActionResult<Employee> GetStandard()

***{***

var employee = \_employees.FirstOrDefault(); if (employee == null)

return NotFound();

return Ok(employee);

***}***

[HttpPost]

public IActionResult Post(Employee employee)

***{***

\_employees.Add(employee);

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

***}***

[HttpPut("{id}")]

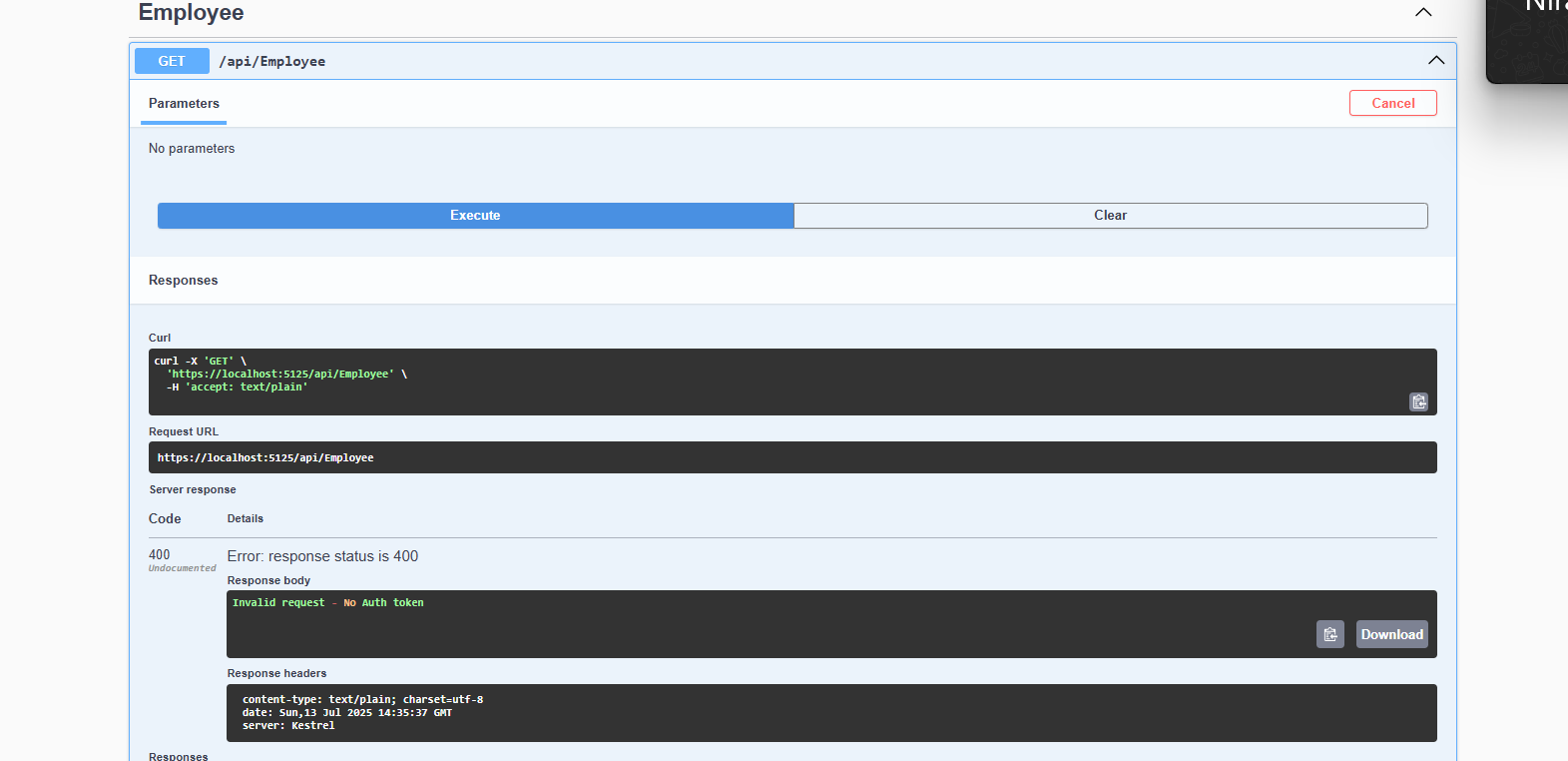
public IActionResult Put(int id, Employee updatedEmployee)

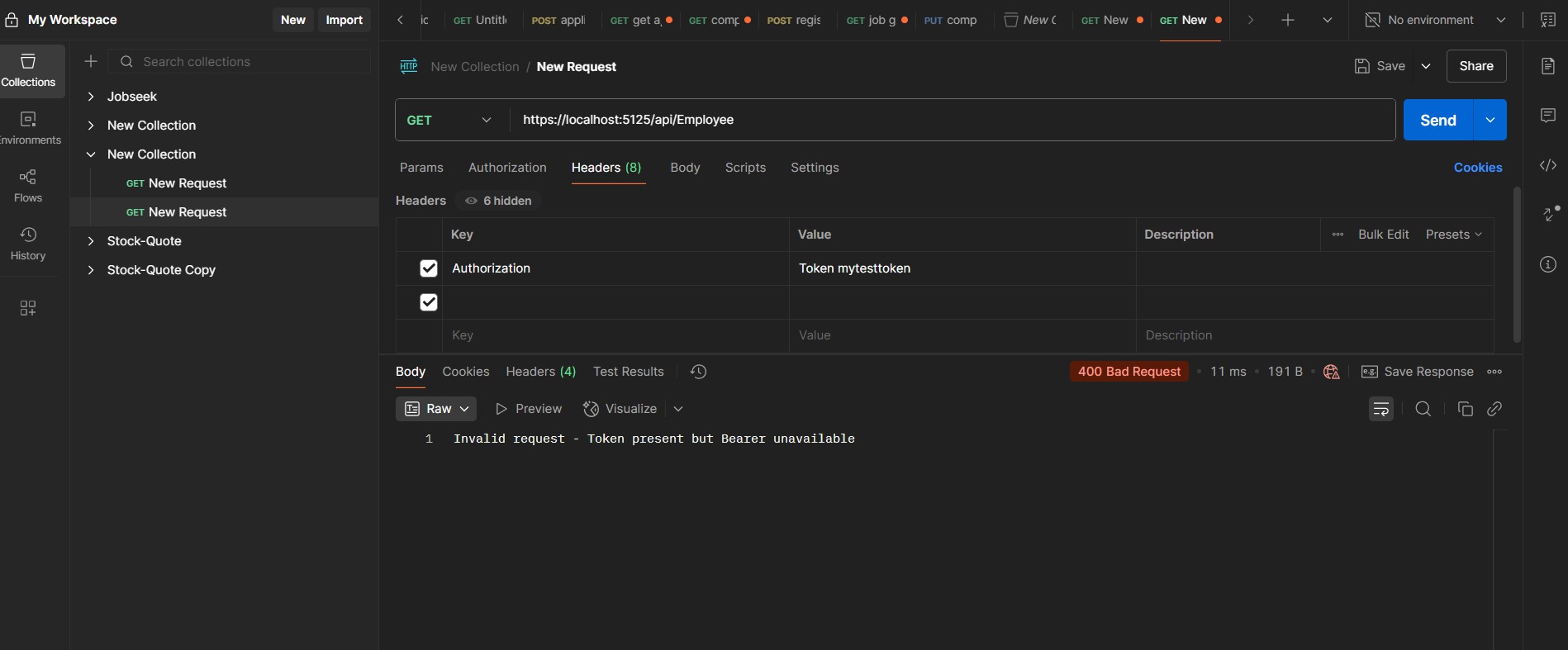
***{***

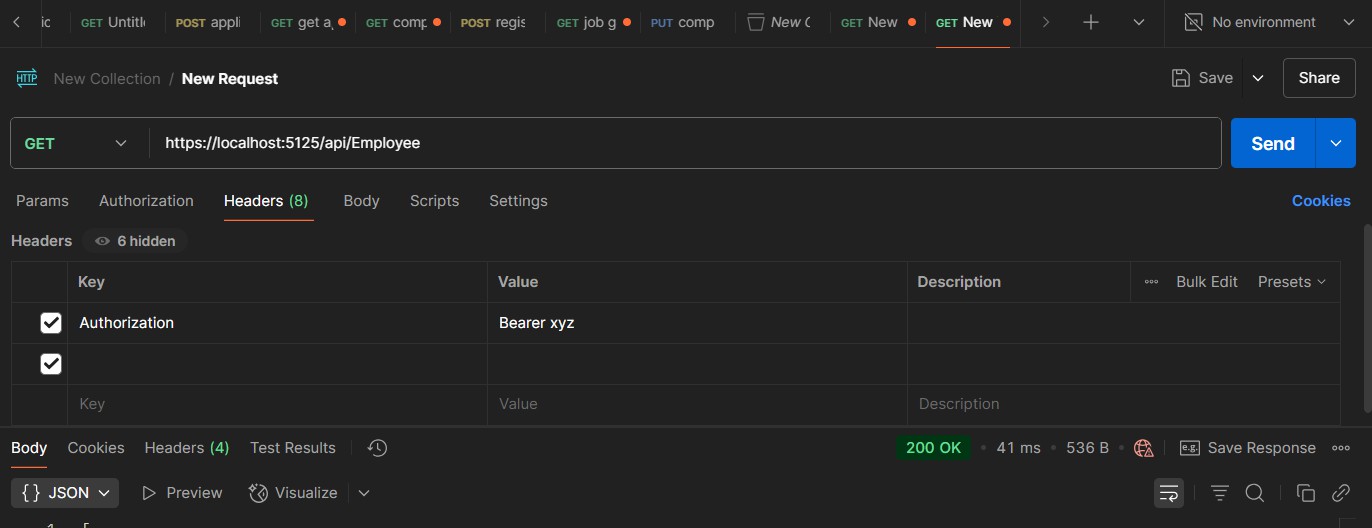
var existingEmployee = \_employees.FirstOrDefault(e => e.Id == id); if (existingEmployee == null)

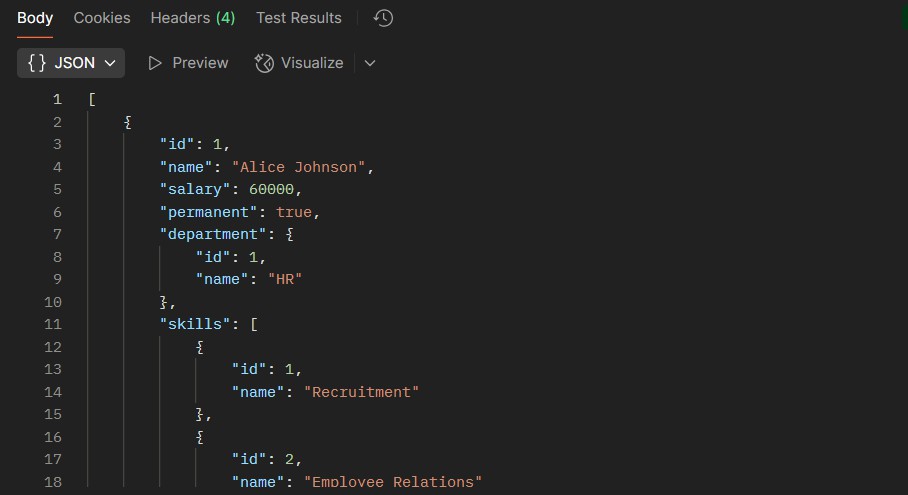
return NotFound();

2 Create a Custom action filter for Authorization. Output:

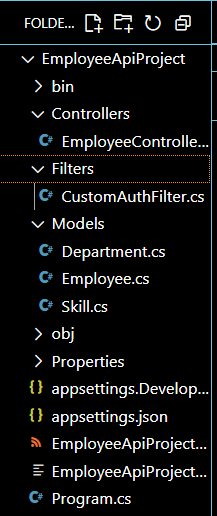








Working Folder:



## *3)* CUSTOM EXCEPTION FILTER

**Filter/customexceptionfilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace EmployeeApiProject.Filters

***{***

public class CustomExceptionFilter : IExceptionFilter

***{***

public void OnException(ExceptionContext context)

***{***

string logFilePath = Path.Combine(Directory.GetCurrentDirectory(), "logs.txt");

string errorDetails = $"[{DateTime.Now}] Exception:

{context.Exception.Message}\n{context.Exception.StackTrace}\n";

File.AppendAllText(logFilePath, errorDetails);

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

***{***

StatusCode = 500

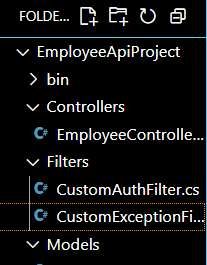
***};***

***}***

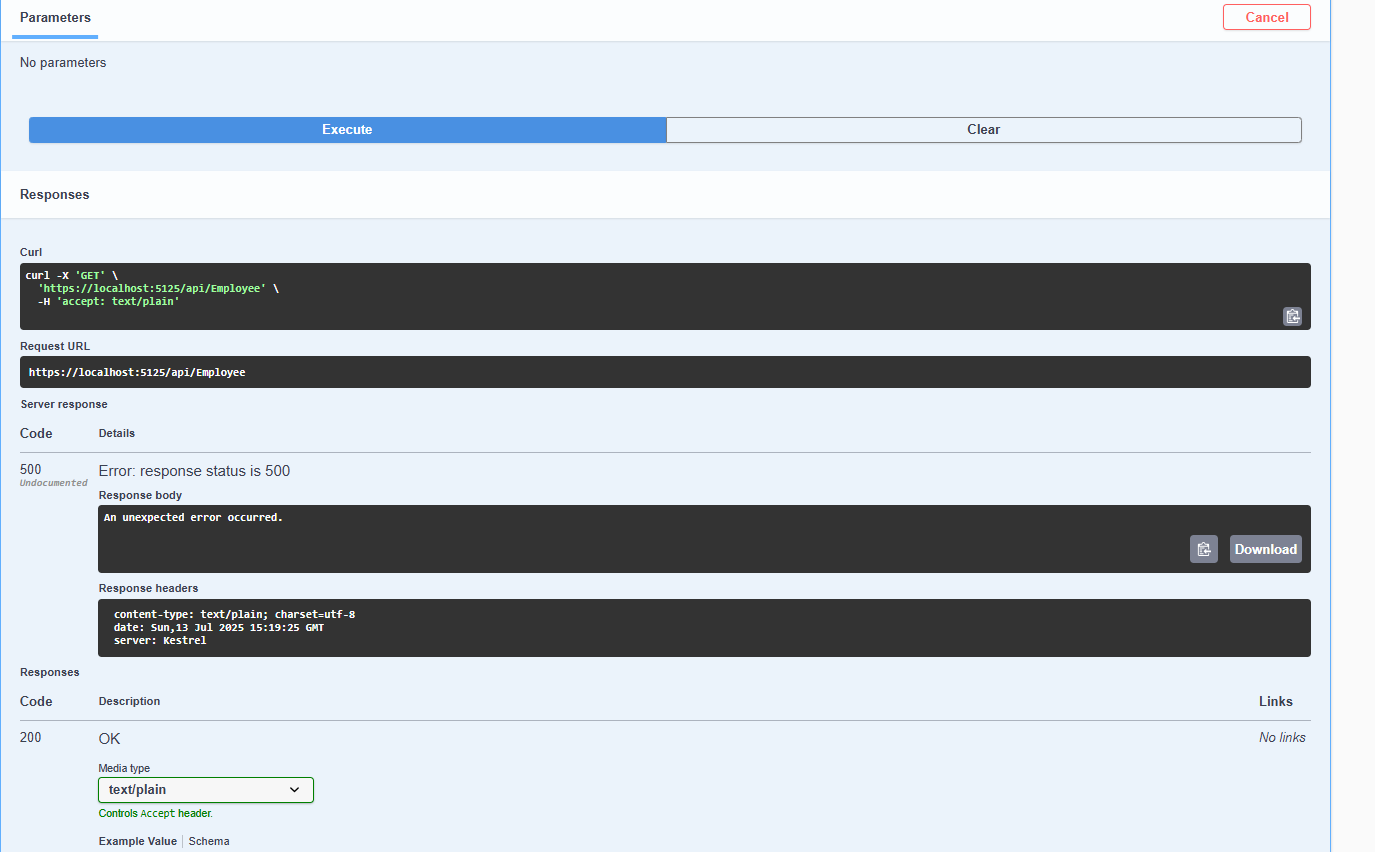
***}***

***}***

Working Folder :



Output:



# MANDATORY HANDS-ON 4 : SOLUTION

**Updated Employee controller code for PUT function as per the given requirements**

[HttpPut("{id}")]

[ProducesResponseType(StatusCodes.Status200OK)] [ProducesResponseType(StatusCodes.Status400BadRequest)]

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

***{***

if (id <= 0)

***{***

return BadRequest("Invalid employee id");

***}***

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

if (existingEmployee == null)

***{***

return BadRequest("Invalid employee id");

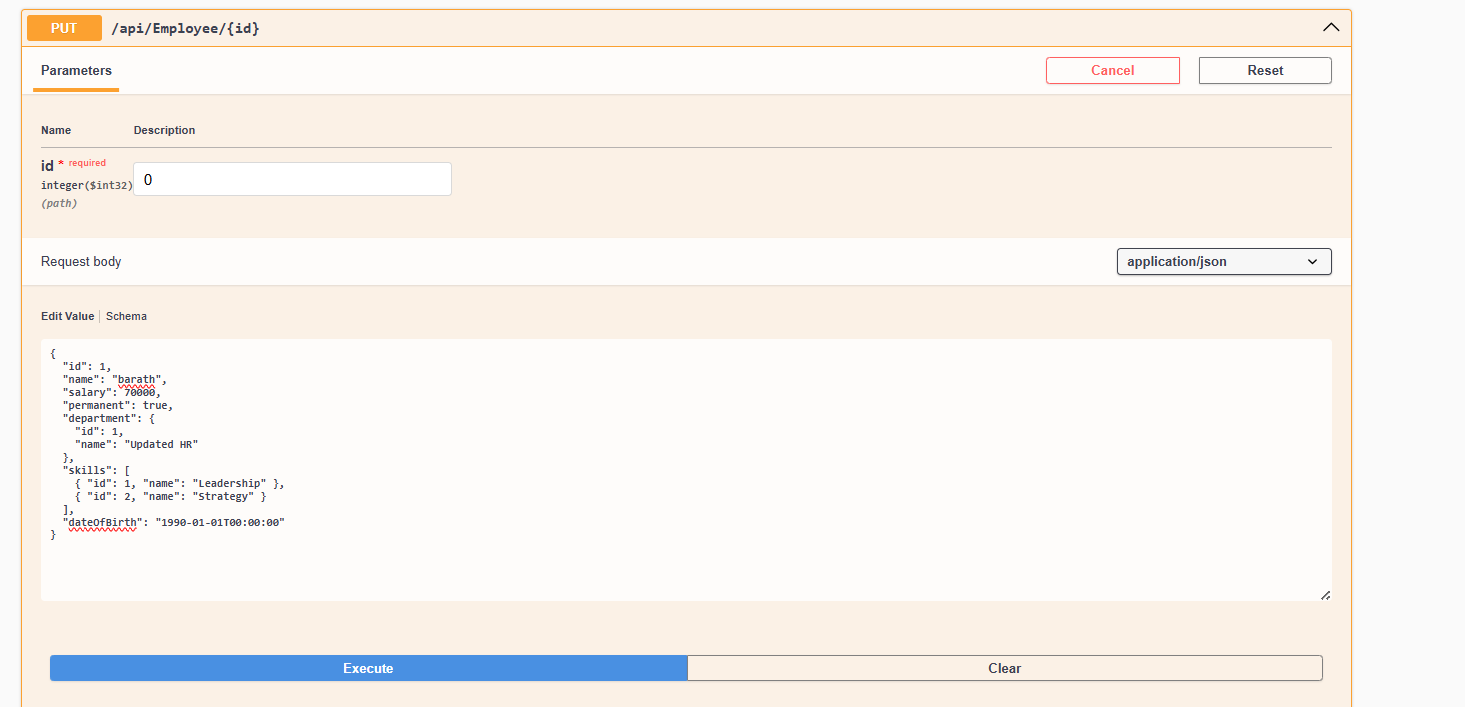
***}***

existingEmployee.Name = updatedEmployee.Name; existingEmployee.Salary = updatedEmployee.Salary; existingEmployee.Permanent = updatedEmployee.Permanent; existingEmployee.Department = updatedEmployee.Department; existingEmployee.Skills = updatedEmployee.Skills; existingEmployee.DateOfBirth = updatedEmployee.DateOfBirth;

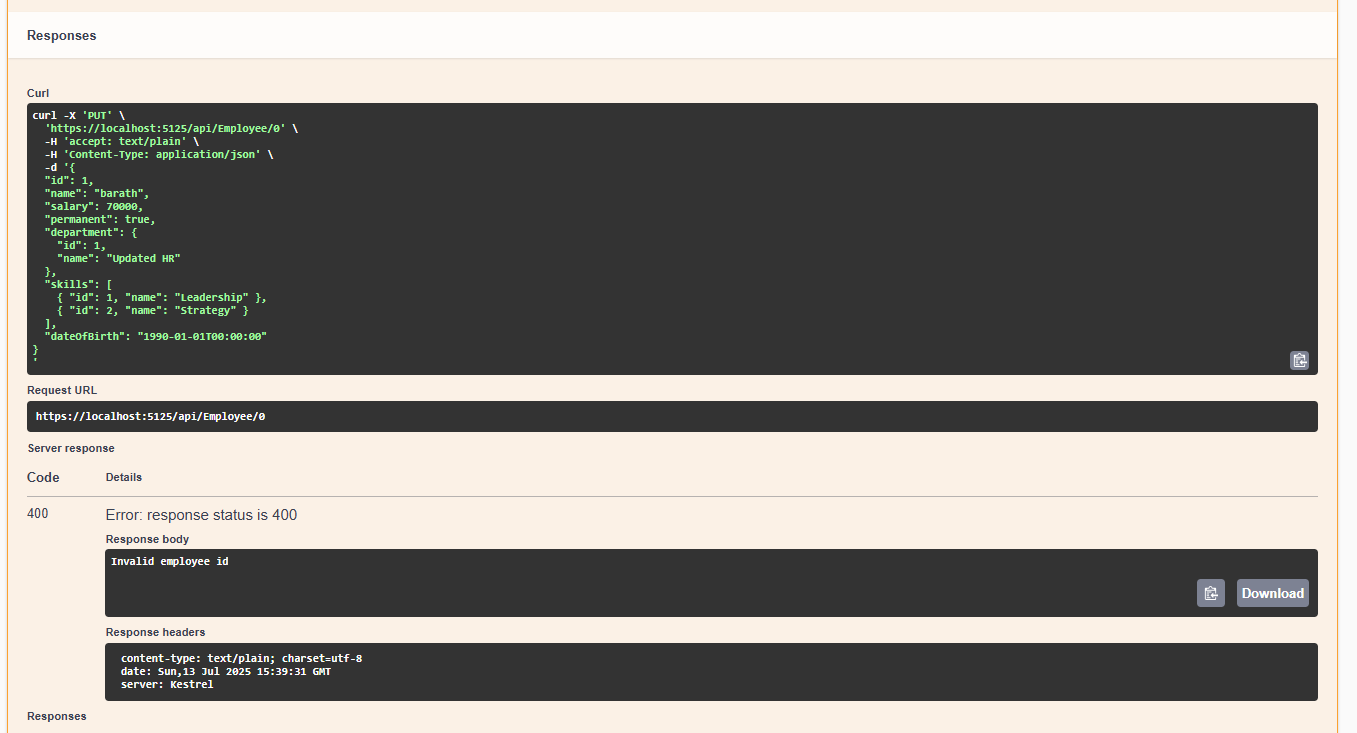
return Ok(existingEmployee);

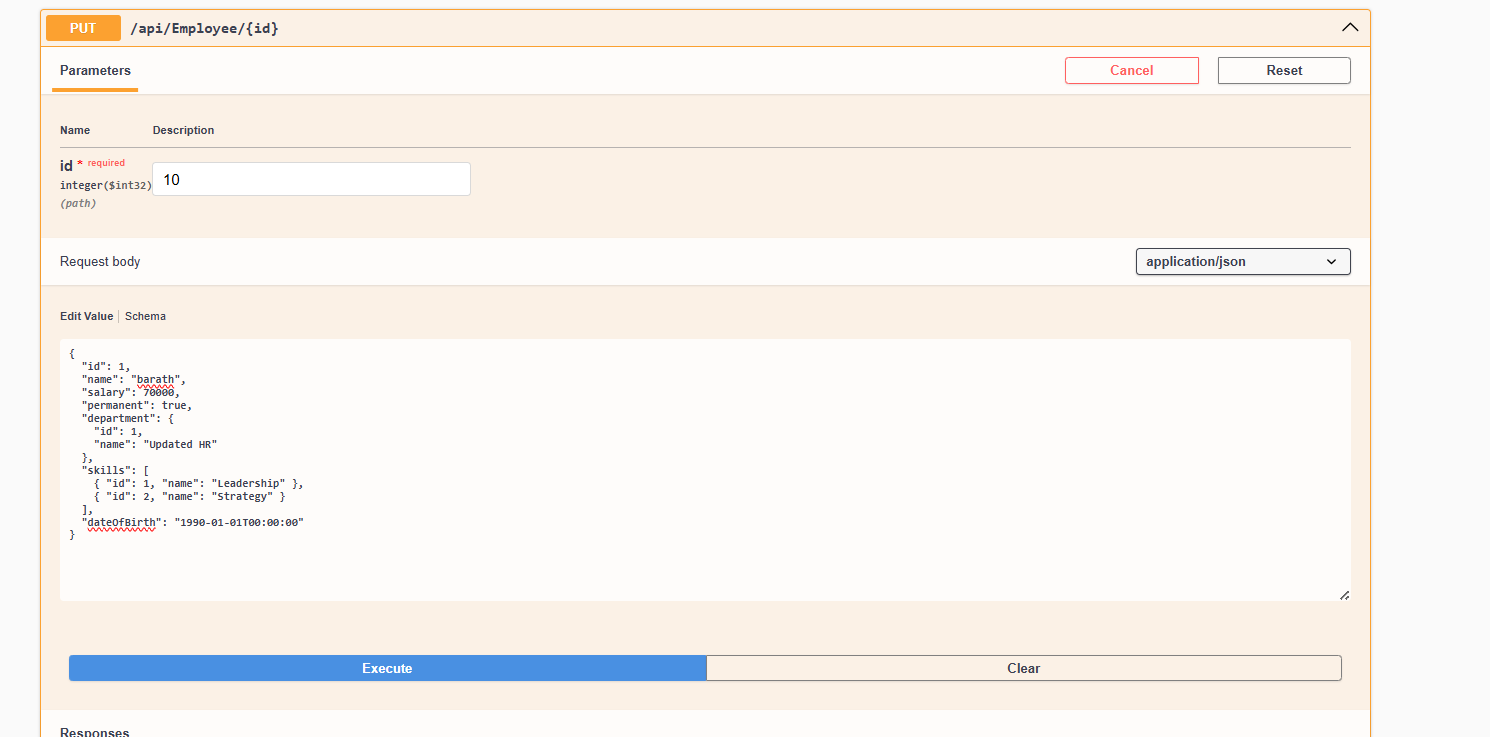
***}***

Output :



Response for the above for id=0;

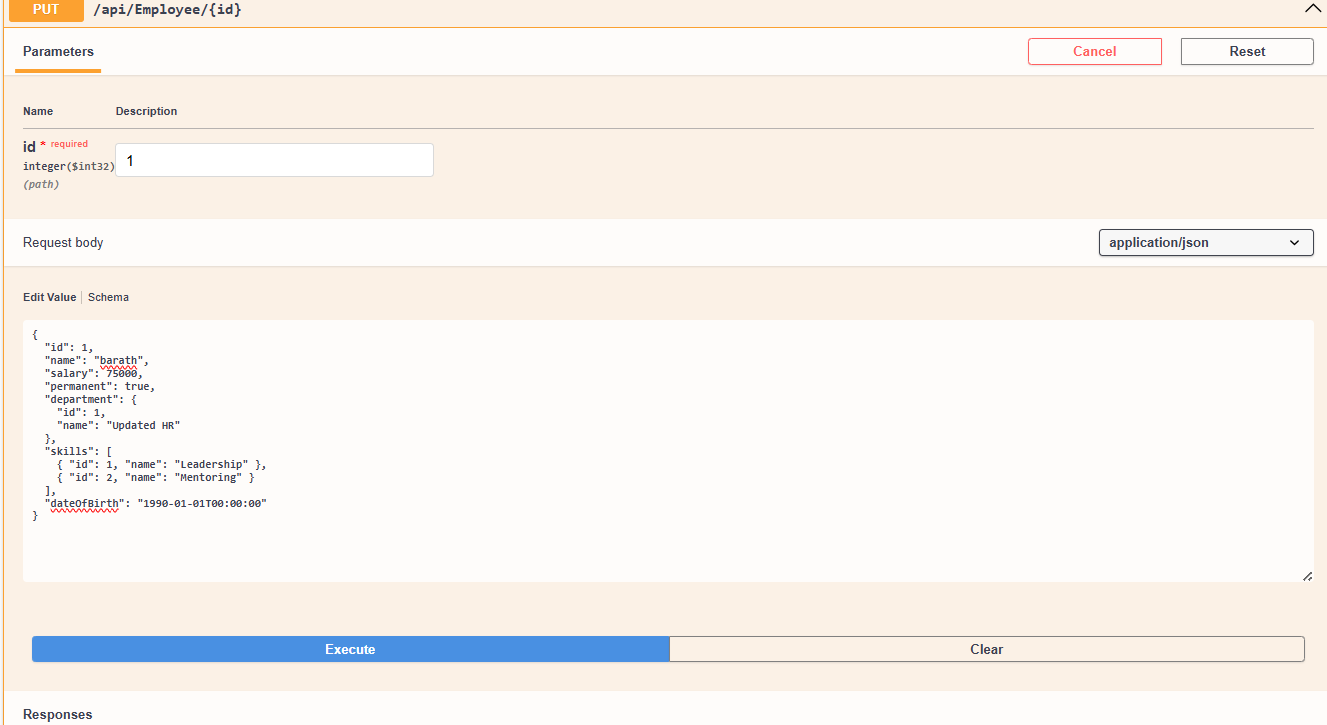




Response of non exisiting id where id = 10 :



**Updated and filtering based on Id**

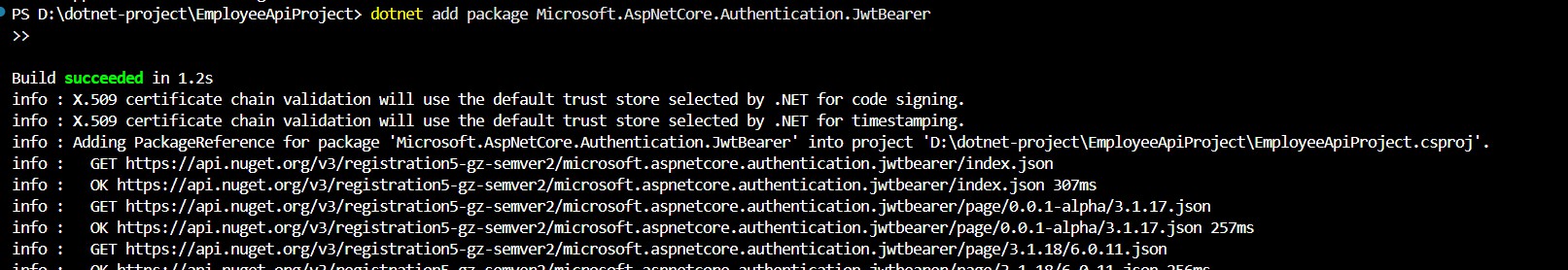


**Response of above call :**



# MANDATORY HANDS-ON EXERCISE : 5 SOLUTION

**Adding the jwt nuGET package**



## AUTHCONTROLL PROGRAM

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using Microsoft.AspNetCore.Authorization; using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims; using System.Text;

namespace EmployeeApiProject.Controllers

***{***

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

public class AuthController : ControllerBase

***{***

[HttpGet]

public IActionResult GetToken()

***{***

var token = GenerateJSONWebToken(123, "Admin"); 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(10), signingCredentials: credentials);

return new JwtSecurityTokenHandler().WriteToken(token);

***}***

***}***

***}***

***UPDATED PROGRAM.CS***

using EmployeeApiProject.Filters;

using Microsoft.AspNetCore.Authentication.JwtBearer; using Microsoft.IdentityModel.Tokens;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container. builder.Services.AddControllers(options =>

***{***

options.Filters.Add<CustomExceptionFilter>();

***});***

// Register Swagger/OpenAPI generation builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(options =>

***{***

options.SwaggerDoc("v1", new OpenApiInfo

***{***

Title = "Employee API", Version = "v1"

***});***

***});***

builder.Services.AddScoped<CustomAuthFilter>(); // Register filter

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

var app = builder.Build();

// Configure the HTTP request pipeline. if (app.Environment.IsDevelopment())

***{***

app.UseSwagger(); app.UseSwaggerUI(options =>

***{***

options.SwaggerEndpoint("/swagger/v1/swagger.json", "Employee API V1"); options.RoutePrefix = string.Empty; // Swagger UI at root

***});***

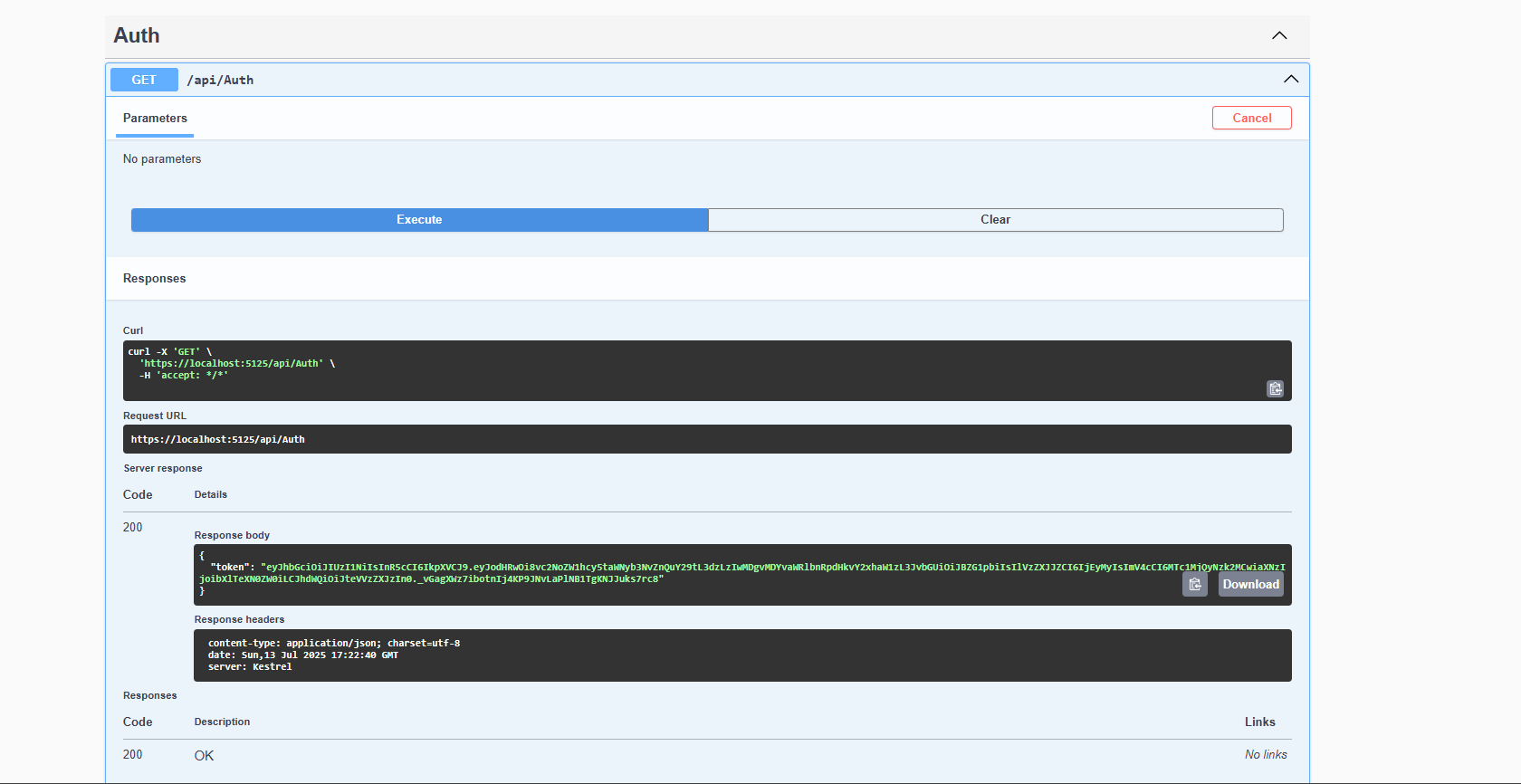
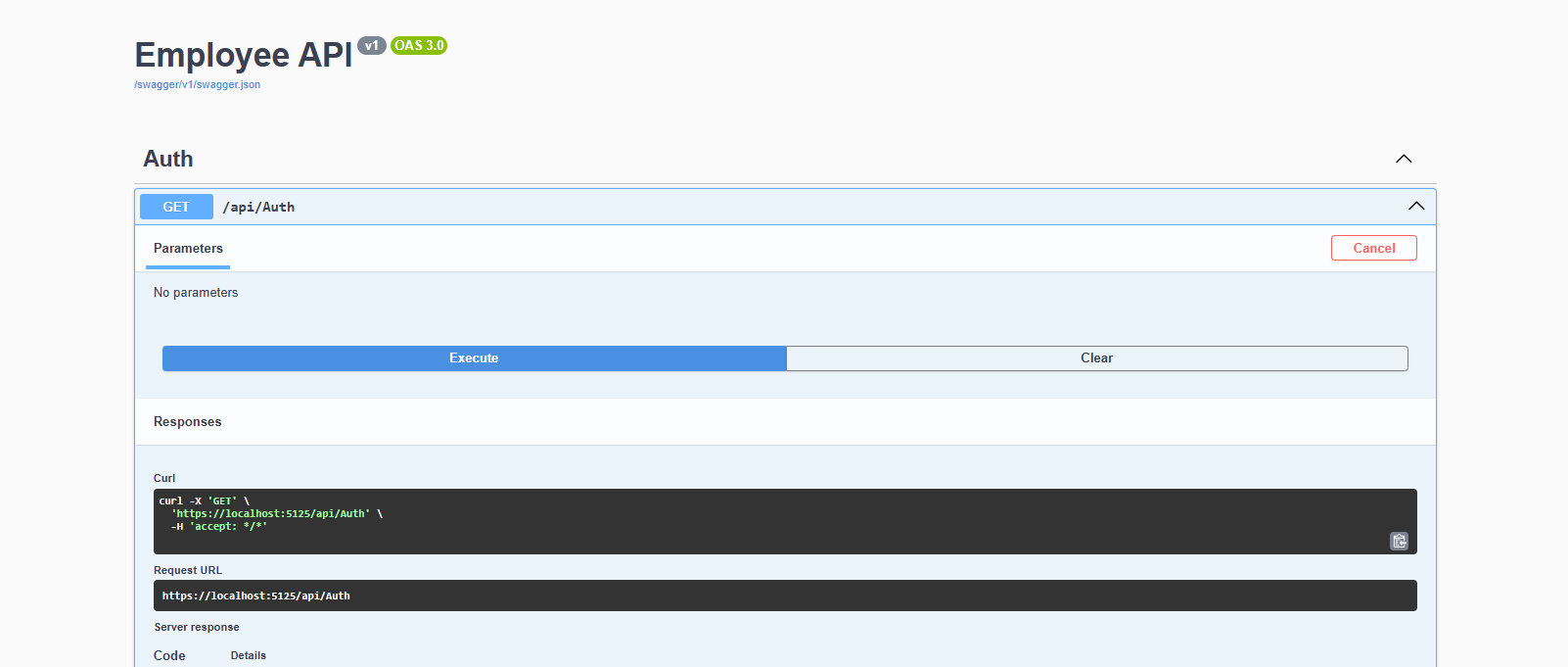
***}***

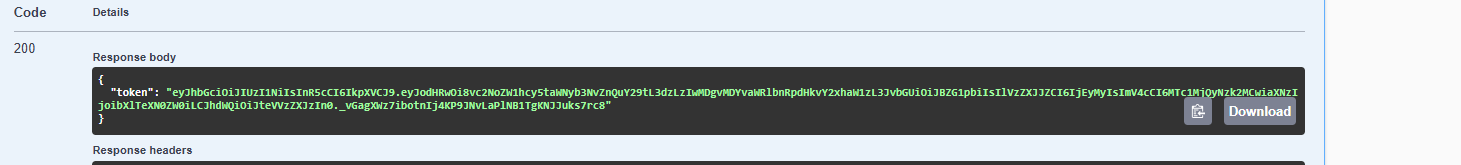
app.UseHttpsRedirection(); app.UseAuthentication();

app.UseAuthorization();

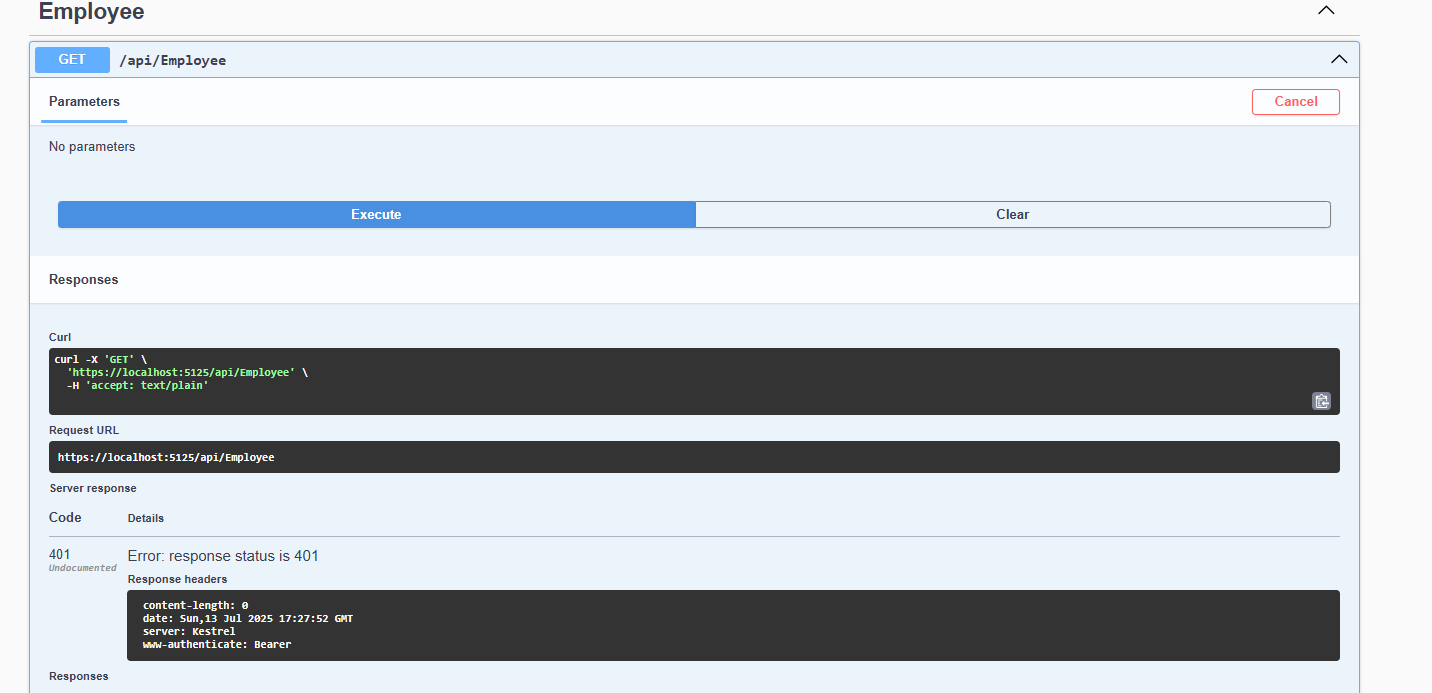
app.MapControllers(); // VERY IMPORTANT: This maps all attribute-routed controllers

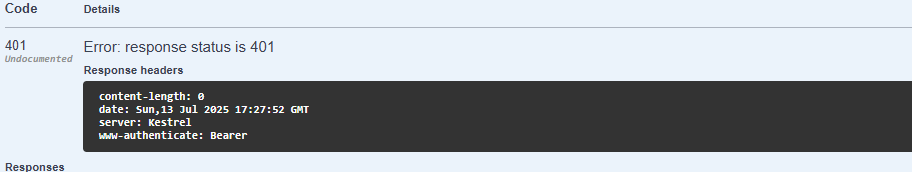
app.Run();



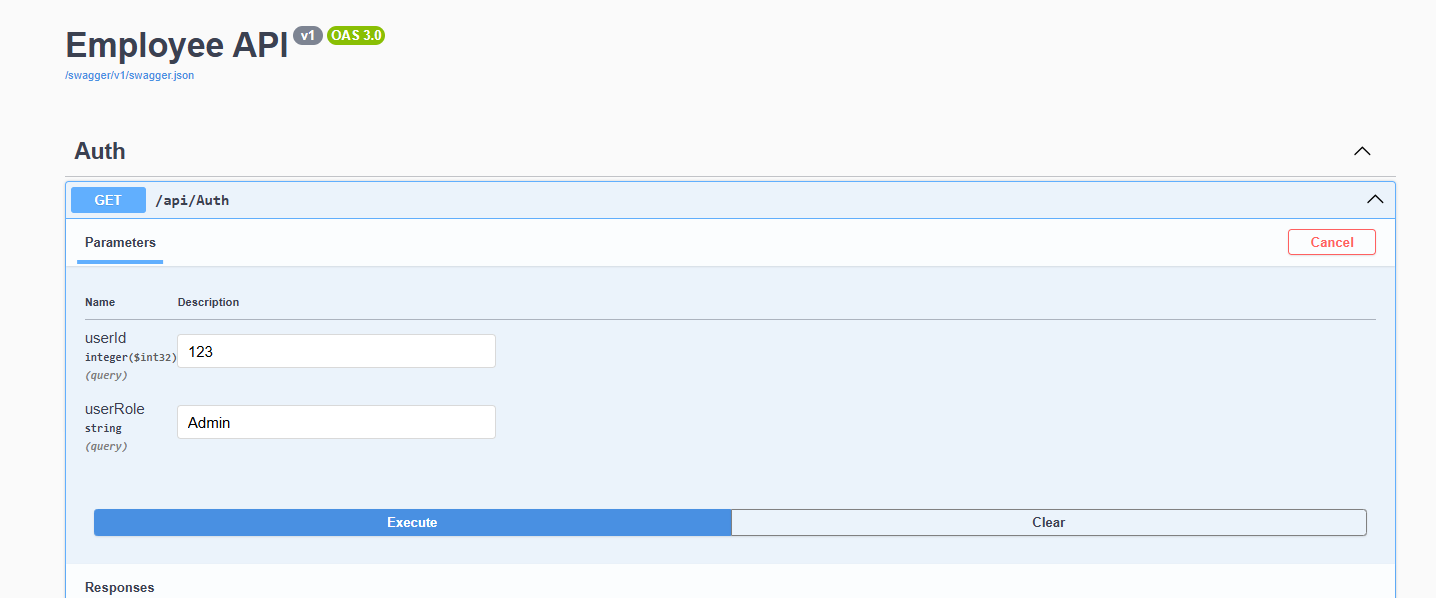


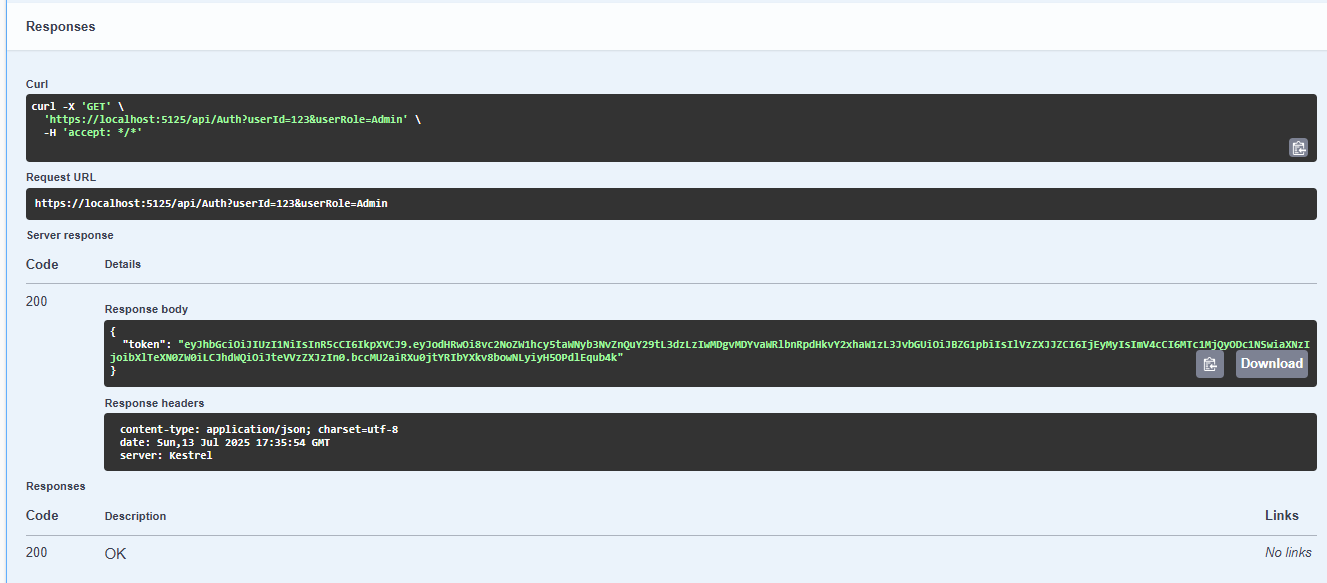
**Accessing the employee data without authentication**



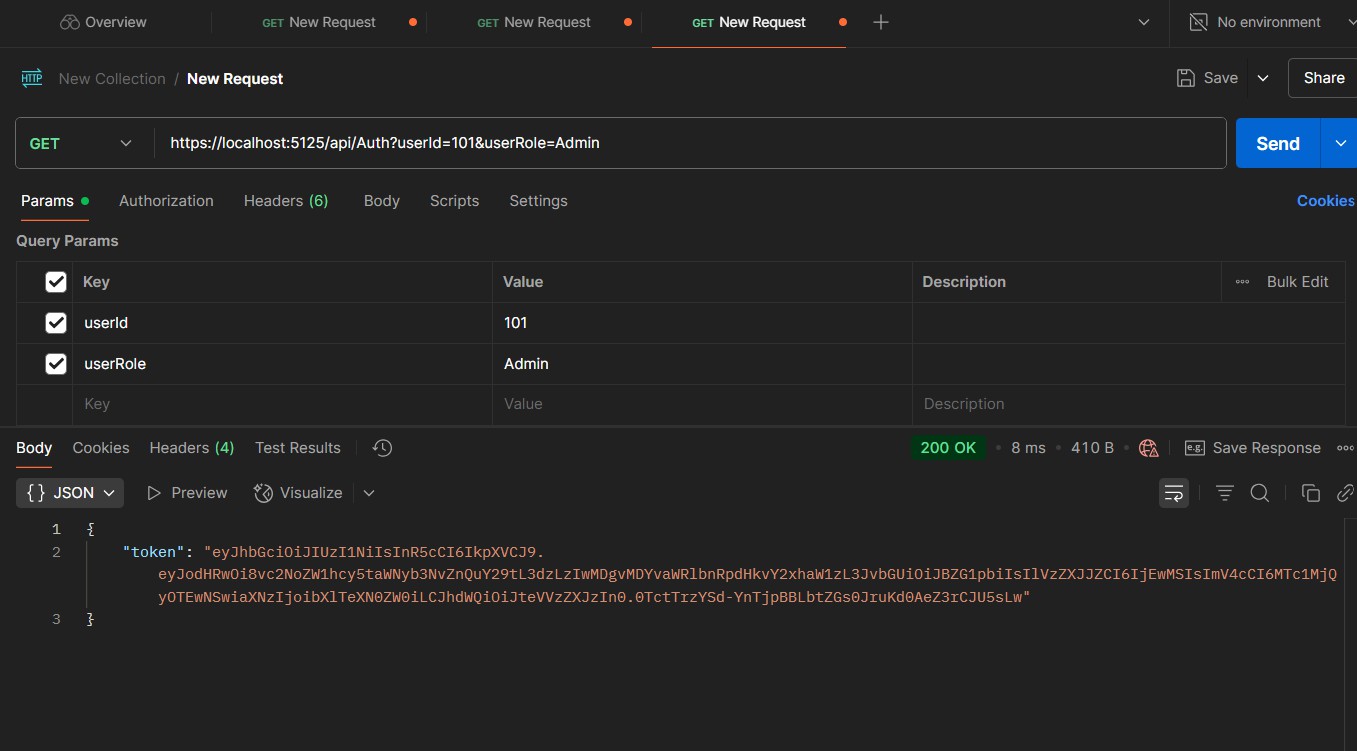


**Accessing the jwt token for specific profile id and role**

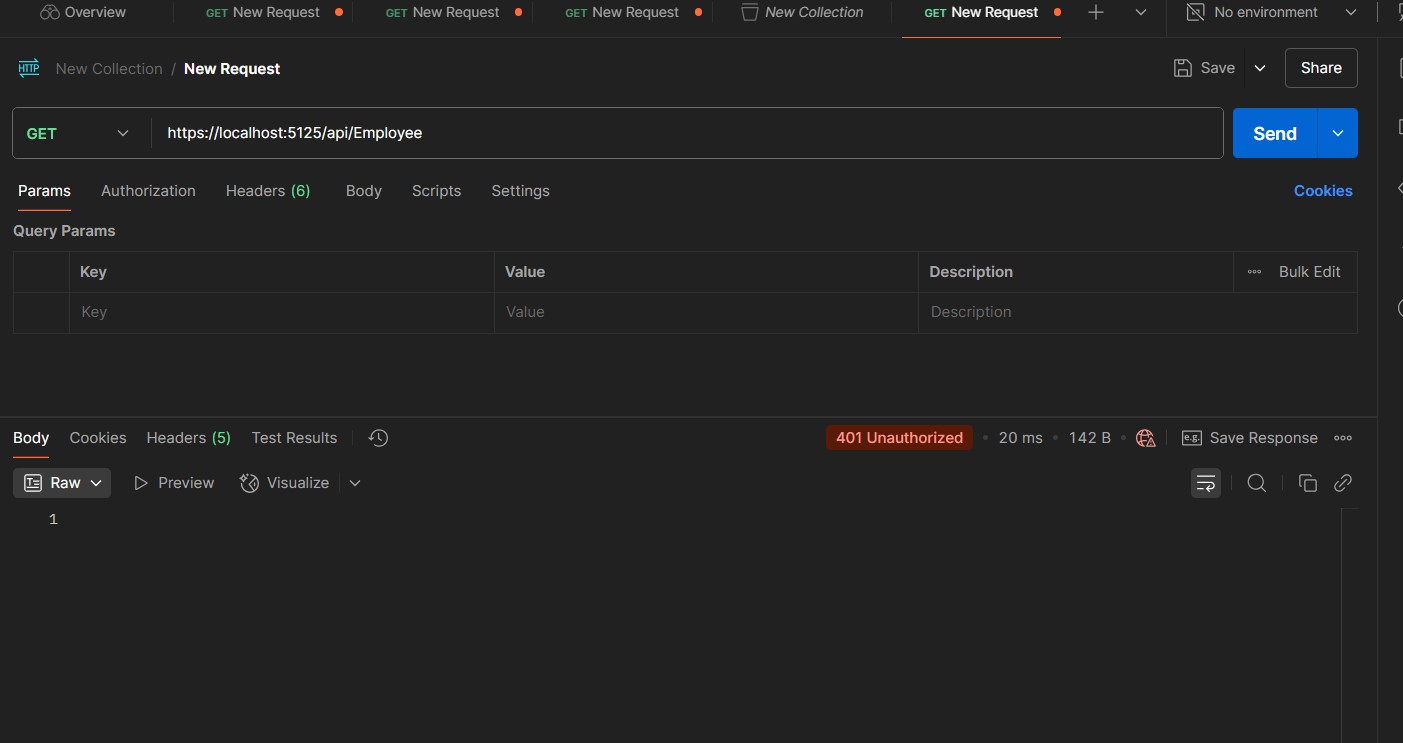




**Testing the api/auth in postman**



**Unauthorize access of the token**



**Program.cs**

using EmployeeApiProject.Filters;

using Microsoft.AspNetCore.Authentication.JwtBearer; using Microsoft.IdentityModel.Tokens;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container. builder.Services.AddControllers(options =>

***{***

options.Filters.Add<CustomExceptionFilter>();

***});***

// Register Swagger/OpenAPI generation builder.Services.AddEndpointsApiExplorer(); builder.Services.AddSwaggerGen(options =>

***{***

options.SwaggerDoc("v1", new OpenApiInfo

***{***

Title = "Employee API", Version = "v1"

***});***

***});***

builder.Services.AddScoped<CustomAuthFilter>(); // Register filter

string securityKey = "mysuperdupersecretkey!1234567890";

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

var app = builder.Build();

// Configure the HTTP request pipeline. if (app.Environment.IsDevelopment())

***{***

app.UseSwagger(); app.UseSwaggerUI(options =>

***{***

options.SwaggerEndpoint("/swagger/v1/swagger.json", "Employee API V1"); options.RoutePrefix = string.Empty; // Swagger UI at root

***});***

***}***

app.UseHttpsRedirection(); app.UseAuthentication();

app.UseAuthorization();

app.MapControllers(); // VERY IMPORTANT: This maps all attribute-routed controllers app.Run();

Updated authcontroller.cs code using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using Microsoft.AspNetCore.Authorization; using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims; using System.Text;

namespace EmployeeApiProject.Controllers

***{***

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

public class AuthController : ControllerBase

***{***

[HttpGet]

public IActionResult GetToken([FromQuery] int userId, [FromQuery] string userRole)

***{***

try

***{***

var token = GenerateJSONWebToken(userId, userRole); return Ok(new { token });

***}***

catch (Exception ex)

***{***

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

***}***

***}***

private string GenerateJSONWebToken(int userId, string userRole)

***{***

var securityKey = new

SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretkey!1234567890"));

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(10), signingCredentials: credentials);

return new JwtSecurityTokenHandler().WriteToken(token);

***}***

***}***

***}***

3) check for Jwt expiration

var token = new JwtSecurityToken( issuer: "mySystem",

audience: "myUsers", claims: claims,

expires: DateTime.Now.AddMinutes(2),

signingCredentials: credentials

***);***