1. **WebAPI HandsOn-1**

### 1. RESTful Web Service, Web API, and Microservice

### RESTful Web Service:

A RESTful Web Service follows the REST (Representational State Transfer) architecture. It uses standard HTTP methods (GET, POST, PUT, DELETE) to perform operations on resources identified by URIs.

**Stateless communication**

**Data formats**: JSON, XML, etc.

**Platform-independent**

**Example**: GET /api/products returns product list

### Web API:

A Web API is a framework that enables developers to build HTTP services for browsers, mobile apps, and other clients.

In ASP.NET Core, it's built using controllers and action methods.

Uses **routing**, **action verbs**, and **status codes**

Can return data in multiple formats (JSON is default)

### Microservice:

A microservice is a small, independent module that performs a specific business function. Multiple microservices communicate via APIs to form a complete application.

**Decoupled architecture**

**Independently deployable**

**Highly scalable and maintainable**

### ****2. Features of REST Architecture****

**Statelessness**: Each client request must contain all the information needed to understand and process it; the server does not store client state between requests.

**Resource-based**: Everything is treated as a resource and identified by a unique URI (Uniform Resource Identifier).

**Standard HTTP Methods**: REST uses HTTP verbs such as GET, POST, PUT, DELETE to perform operations on resources.

**Multiple Representations**: Resources can be represented in various formats such as JSON, XML, HTML, etc., depending on client preference.

**Uniform Interface**: A consistent way to interact with resources regardless of the resource type or client.

**Cacheable Responses**: Responses from the server can be explicitly marked as cacheable or non-cacheable to improve performance.

**Layered Architecture**: REST allows an architecture composed of hierarchical layers where each layer only communicates with the layer directly beneath it.

### ****3. Difference between WebService, Web API, and Microservice****

* **WebService** uses mainly SOAP protocol and is tightly coupled with XML data format.
* **Web API** is a lightweight, flexible service based on REST, which supports multiple data formats like JSON and XML.
* **Web API** is simpler and faster than traditional Web Services and suitable for mobile, web, and desktop apps.
* **Microservices** are small, independent services that focus on doing one task well and can be developed, deployed, and scaled independently.
* **Web API** can be a part of a microservice architecture to expose the microservice's functionalities over HTTP.
* **Microservices** often use Web APIs to communicate with each other internally or with external clients.

### 4. ****HttpRequest and HttpResponse****

**HttpRequest:**

* Represents the data sent by the client to the server.
* Contains HTTP method, URI, headers (like Authorization, Content-Type), and an optional body (especially for POST/PUT).
* Used to carry data like form inputs, JSON payloads, authentication tokens, etc.

**HttpResponse:**

* Represents the server’s reply to an HTTP request.
* Contains a status code (e.g., 200, 404, 500), headers (e.g., Content-Type), and response body (like JSON data or error message).
* Indicates whether the request was successful or failed.

### ****5. Types of Action Verbs in Web API****

* **HttpGet**: Used to retrieve a resource or collection of resources.
* **HttpPost**: Used to create a new resource on the server.
* **HttpPut**: Used to update an existing resource completely.
* **HttpDelete**: Used to delete a resource by ID or key.

These are implemented as attributes above action methods in the controller.

They guide the routing mechanism in identifying which method to execute for a given HTTP request.

### 6.****Types of HttpStatusCodes used in Web API****

* **200 OK**: The request was successful, and the server is returning the requested data.
* **201 Created**: A new resource has been created successfully (used with POST).
* **400 Bad Request**: The client sent invalid data or the request could not be understood by the server.
* **401 Unauthorized**: The request lacks valid authentication credentials.
* **403 Forbidden**: The server understood the request, but refuses to authorize it.
* **404 Not Found**: The requested resource could not be found on the server.
* **500 Internal Server Error**: The server encountered an error while processing the request.

These are returned using action result methods like return Ok(), return BadRequest(), return NotFound(), etc.

First WEBAPI using .NET Core

Model(Product.cs)

namespace WebAPIHandsOn.Models

{

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

}

}

Controller(TestController.cs)

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using WebAPIHandsOn.Models;

namespace WebAPIHandsOn.Controllers

{

[ApiController]

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

public class ProductsController : ControllerBase

{

private static List<Product> products = new()

{

new Product { Id = 1, Name = "Pen" },

new Product { Id = 2, Name = "Notebook" }

};

[HttpGet]

public IActionResult Get() => Ok(products);

[HttpGet("{id}")]

public IActionResult GetById(int id)

{

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

if (product == null) return NotFound();

return Ok(product);

}

[HttpPost]

public IActionResult Post([FromBody] Product product)

{

product.Id = products.Count + 1;

products.Add(product);

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

}

[HttpPut("{id}")]

public IActionResult Put(int id, [FromBody] Product updatedProduct)

{

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

if (product == null) return NotFound();

product.Name = updatedProduct.Name;

return Ok(product);

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

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

if (product == null) return NotFound();

products.Remove(product);

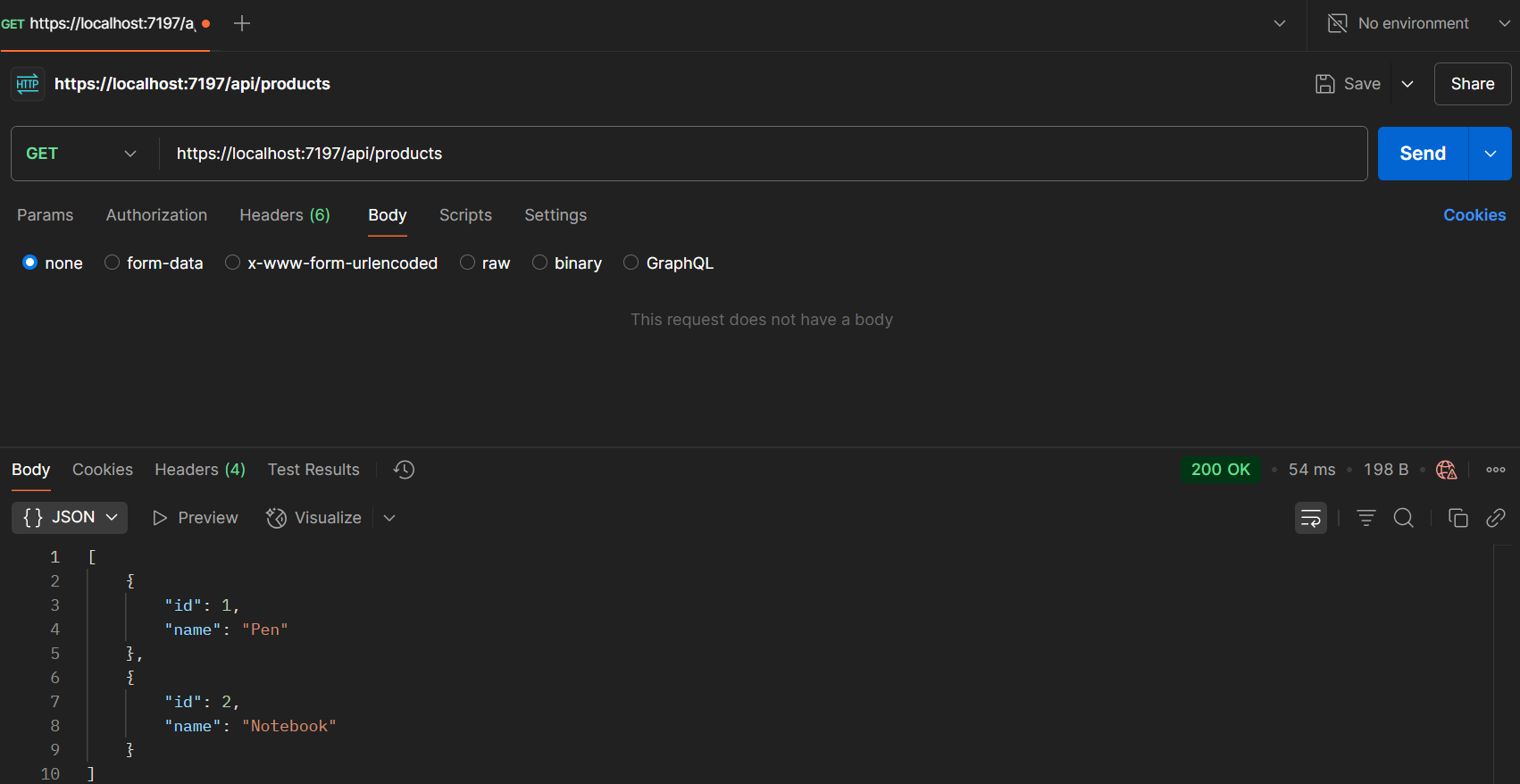
return NoContent();

}

}

}

Testing API:



1. **WebAPI HandsOn-2**

**Modified Program.cs**

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

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"),

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();

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

app.UseAuthorization();

app.MapControllers();

app.Run();

**Employee Model:**

using System.ComponentModel.DataAnnotations;

namespace WebAPIHandsOn.Models

{

public class Employee

{

public int Id { get; set; }

[Required]

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

[Required]

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

}

}

**Employee Controller:**

using Microsoft.AspNetCore.Mvc;

using WebAPIHandsOn.Models;

using System.Collections.Generic;

using System.Linq;

namespace WebAPIHandsOn.Controllers

{

[ApiController]

[Route("Employee")]

public class EmployeeController : ControllerBase

{

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

{

new Employee { Id = 1, Name = "Alice", Department = "HR" },

new Employee { Id = 2, Name = "Bob", Department = "IT" },

new Employee { Id = 3, Name = "Charlie", Department = "Finance" }

};

// GET: /Emp

[HttpGet]

[ProducesResponseType(200)]

public ActionResult<IEnumerable<Employee>> GetAll()

{

return Ok(employees);

}

[HttpGet("{id}")]

[ProducesResponseType(200)]

[ProducesResponseType(404)]

public ActionResult<Employee> GetById(int id)

{

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

if (emp == null) return NotFound();

return Ok(emp);

}

[HttpPost]

[ProducesResponseType(201)]

public ActionResult<Employee> Create(Employee emp)

{

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

employees.Add(emp);

return CreatedAtAction(nameof(GetById), new { id = emp.Id }, emp);

}

[HttpPut("{id}")]

[ProducesResponseType(204)]

[ProducesResponseType(404)]

public IActionResult Update(int id, Employee emp)

{

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

if (existing == null) return NotFound();

existing.Name = emp.Name;

existing.Department = emp.Department;

return NoContent();

}

[HttpDelete("{id}")]

[ProducesResponseType(204)]

[ProducesResponseType(404)]

public IActionResult Delete(int id)

{

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

if (emp == null) return NotFound();

employees.Remove(emp);

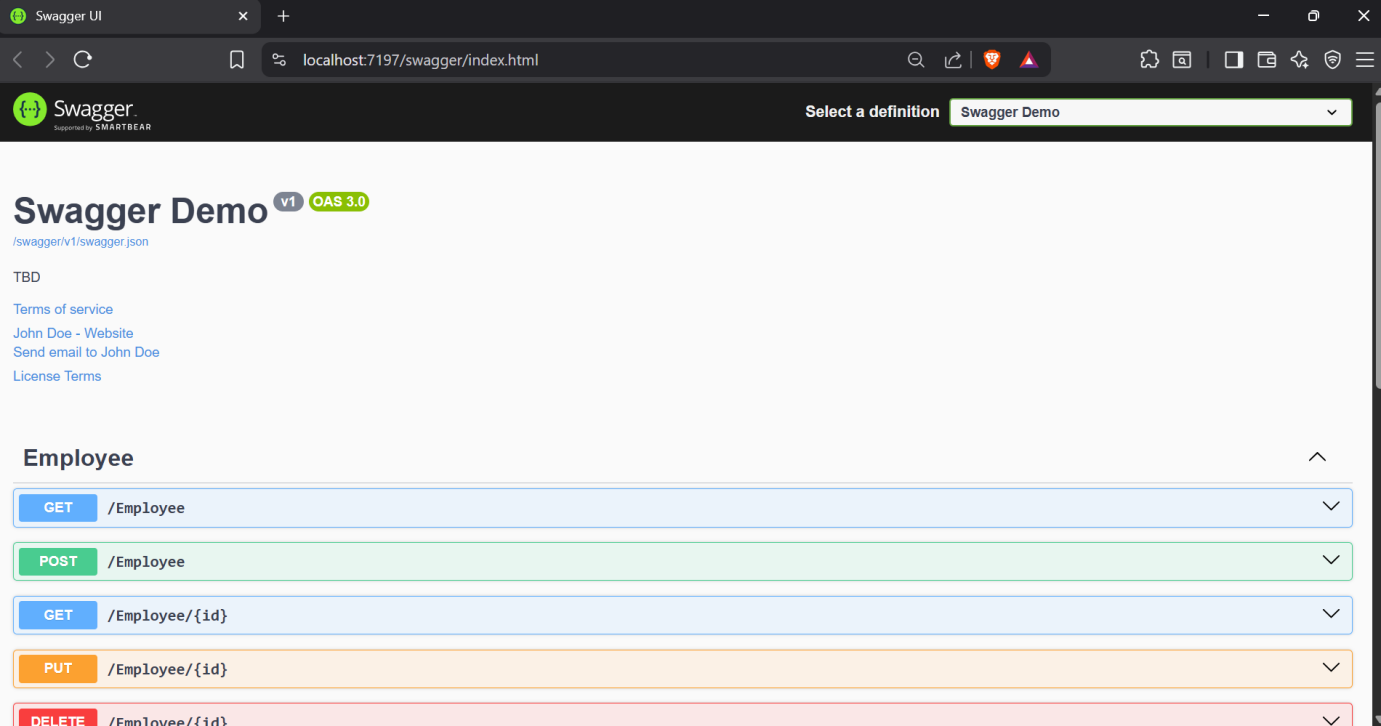
return NoContent();

}

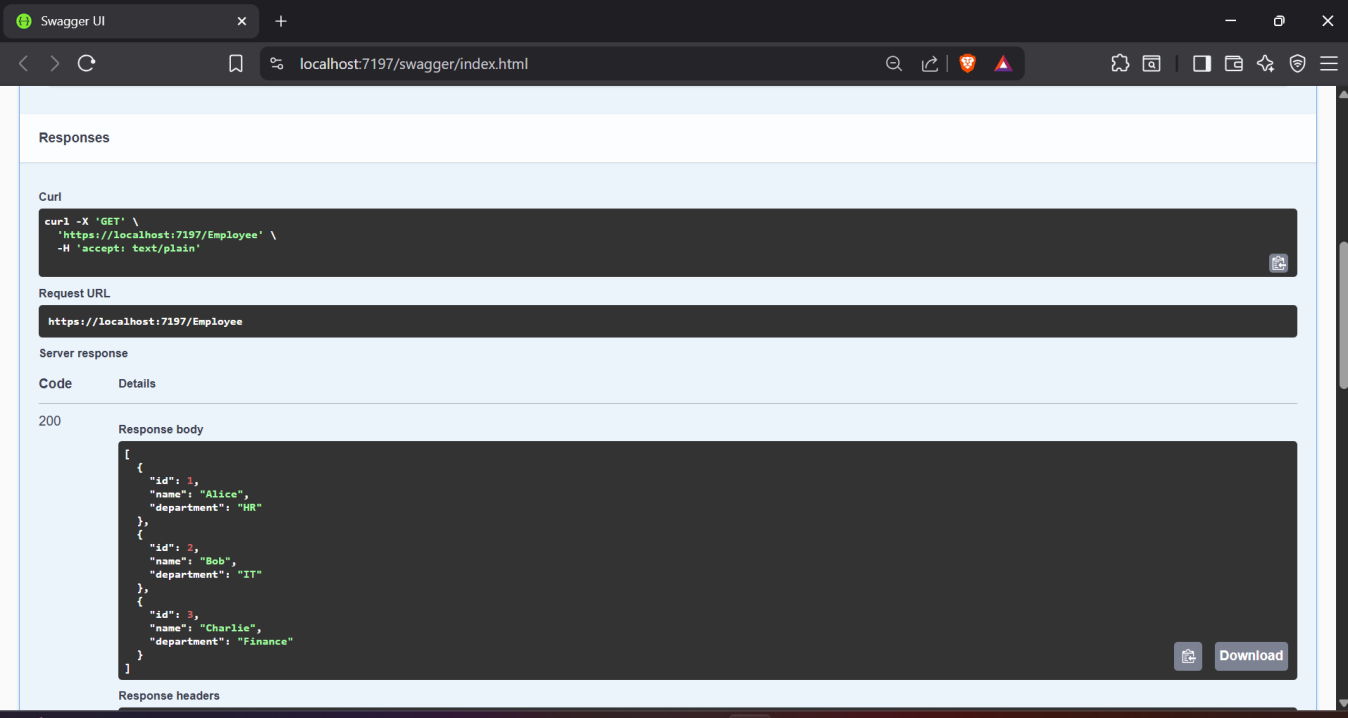
}

}

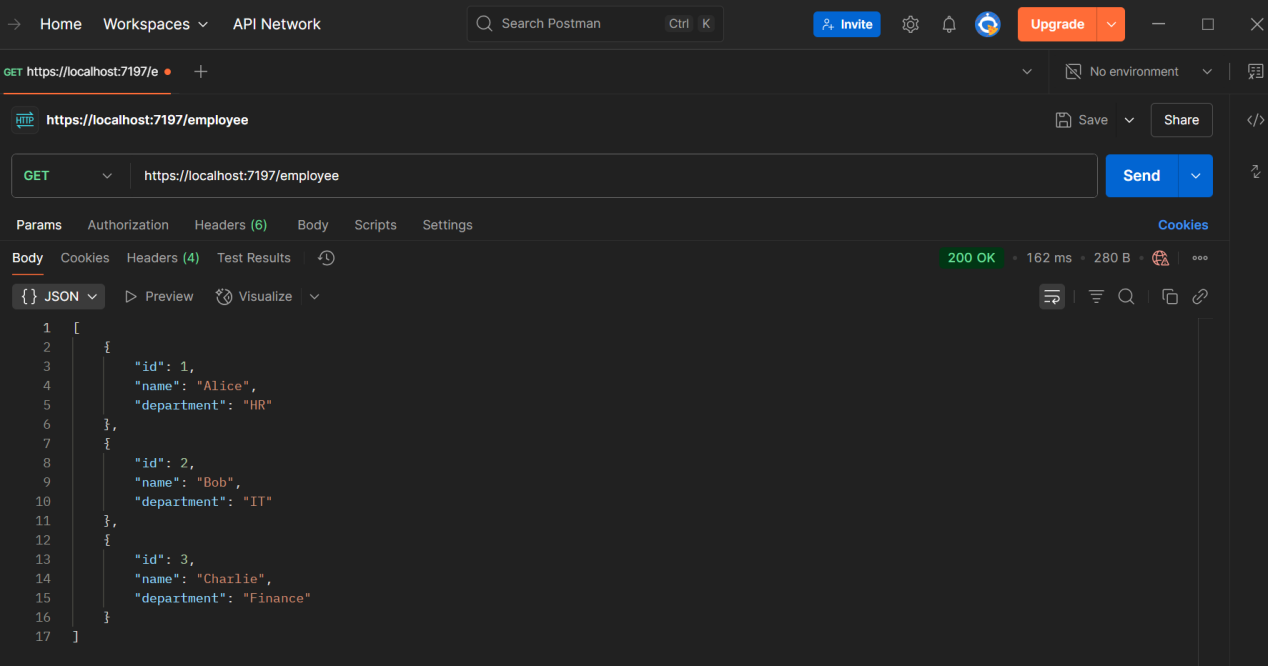
**Output(Swagger page):**



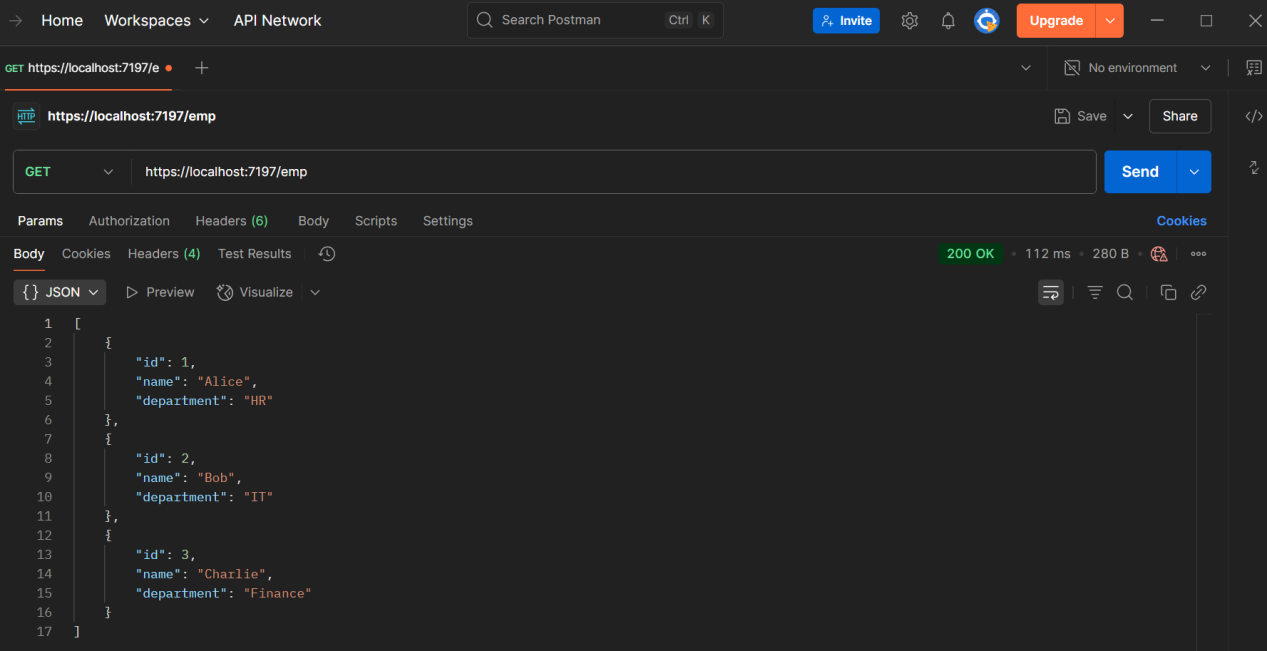
**Testing Get method:**



**GET method using POSTMAN tool**



**Checking GET method after modifying route to “Emp”**



1. **WebAPIHandsOn-3**

**Employee Model**

namespace WebAPIHandsOn.Models

{

public class Department

{

public int Id { get; set; }

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

}

public class Skill

{

public int Id { get; set; }

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

}

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 Department Department { get; set; }

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

public DateTime DateOfBirth { get; set; }

}

}

EmployeeController:

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using WebAPIHandsOn.Filters;

using WebAPIHandsOn.Models;

using System;

using System.Collections.Generic;

namespace WebAPIHandsOn.Controllers

{

[ApiController]

[Route("[controller]")]

public class EmployeeController : ControllerBase

{

private readonly ILogger<EmployeeController> \_logger;

public EmployeeController(ILogger<EmployeeController> logger)

{

\_logger = logger;

}

[HttpGet]

[ProducesResponseType(typeof(List<Employee>), StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public ActionResult<List<Employee>> Get()

{

var employees = GetStandardEmployeeList();

return Ok(employees);

}

[HttpPost]

public IActionResult Post([FromBody] Employee emp)

{

return Created("", emp);

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John",

Salary = 50000,

Permanent = true,

Department = new Department { Id = 1, Name = "HR" },

Skills = new List<Skill>

{

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

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

},

DateOfBirth = new DateTime(1990, 1, 1)

}

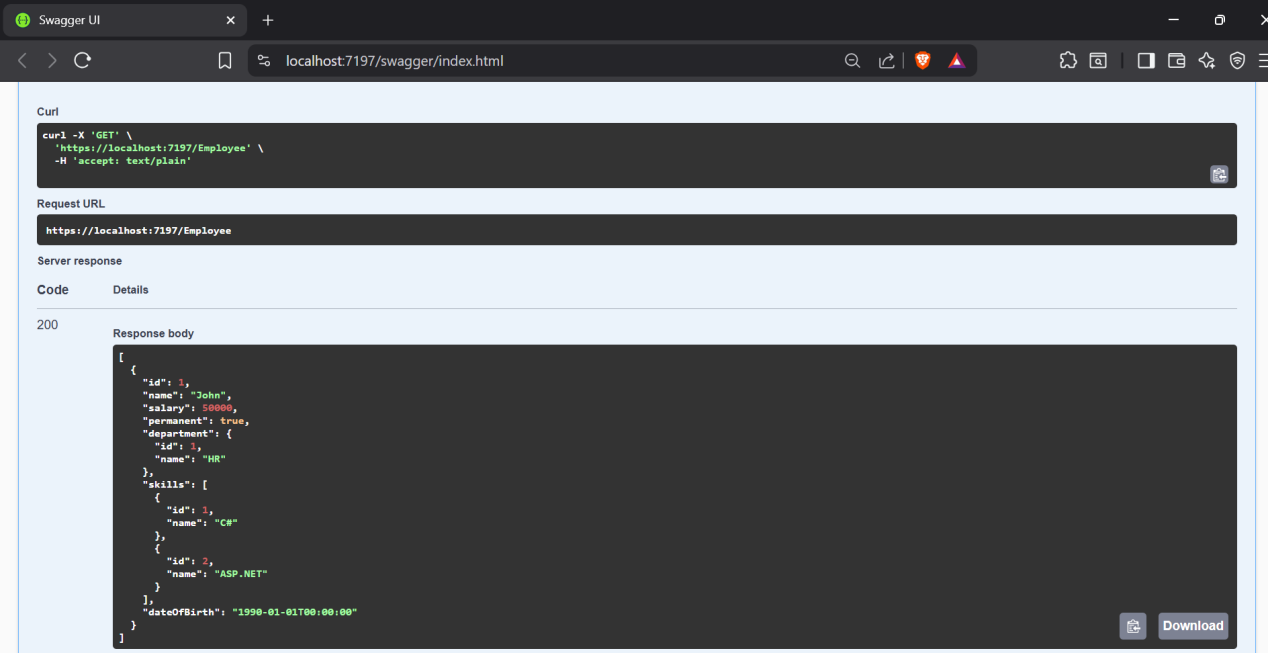
};

}

}

}

**Checking GET request**



**Custom Auth filter:**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace WebAPIHandsOn.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

var authHeader = context.HttpContext.Request.Headers["Authorization"].FirstOrDefault();

if (string.IsNullOrEmpty(authHeader))

{

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

return;

}

if (!authHeader.Contains("Bearer"))

{

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

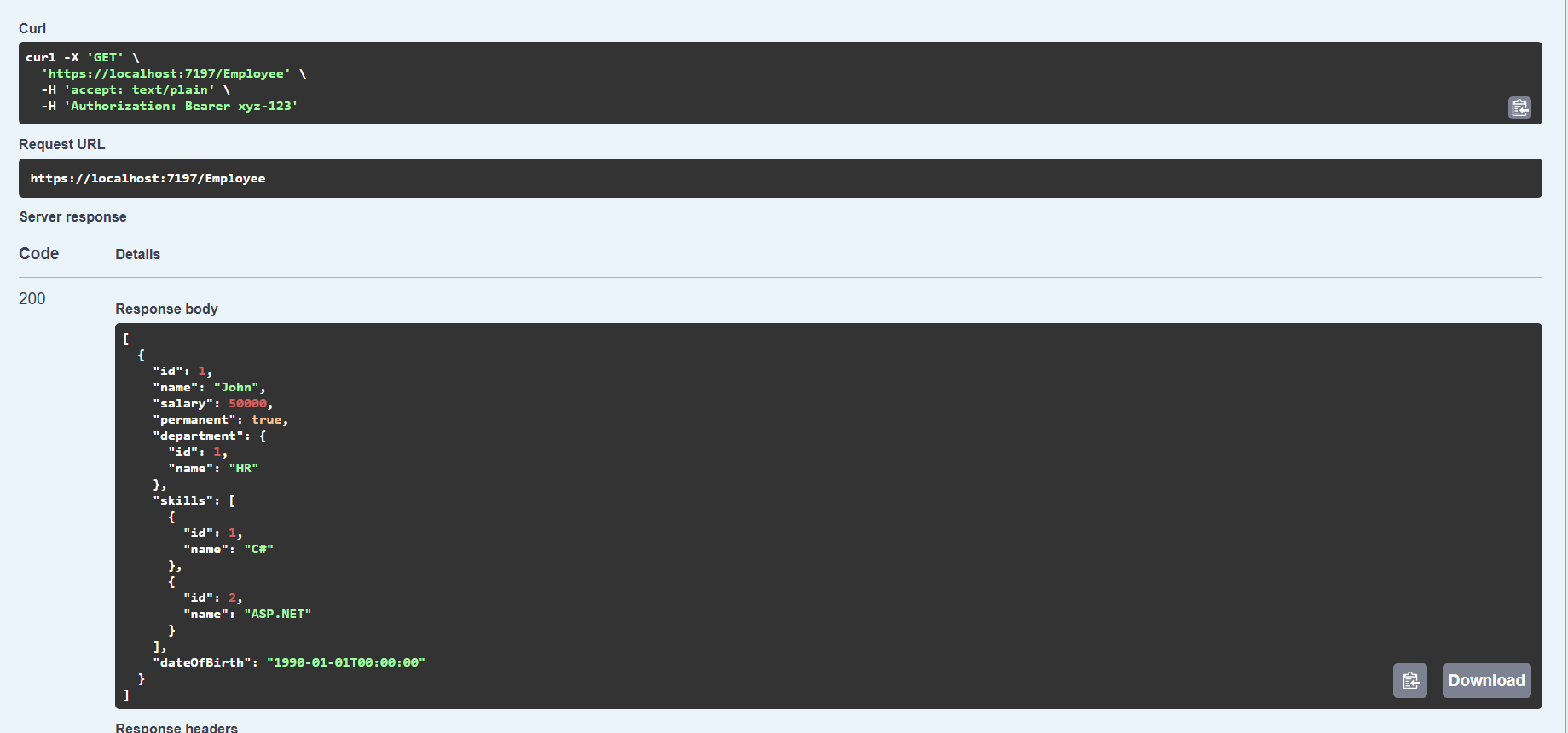
}

}

}

}

**Bearer Token response:**



**Custom Exception filter:**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace WebAPIHandsOn.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

var exception = context.Exception;

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

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

{

StatusCode = StatusCodes.Status500InternalServerError

};

context.ExceptionHandled = true;

}

}

}

**Throwing exception in controller**

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using WebAPIHandsOn.Filters;

using WebAPIHandsOn.Models;

using System;

using System.Collections.Generic;

namespace WebAPIHandsOn.Controllers

{

[ApiController]

[Route("[controller]")]

[TypeFilter(typeof(CustomAuthFilter))]

public class EmployeeController : ControllerBase

{

private readonly ILogger<EmployeeController> \_logger;

public EmployeeController(ILogger<EmployeeController> logger)

{

\_logger = logger;

}

[HttpGet]

[ProducesResponseType(typeof(List<Employee>), StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public ActionResult<List<Employee>> Get()

{

throw new Exception("Custom test exception");

var employees = GetStandardEmployeeList();

return Ok(employees);

}

[HttpPost]

public IActionResult Post([FromBody] Employee emp)

{

return Created("", emp);

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John",

Salary = 50000,

Permanent = true,

Department = new Department { Id = 1, Name = "HR" },

Skills = new List<Skill>

{

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

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

},

DateOfBirth = new DateTime(1990, 1, 1)

}

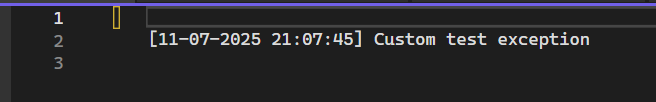
};

}

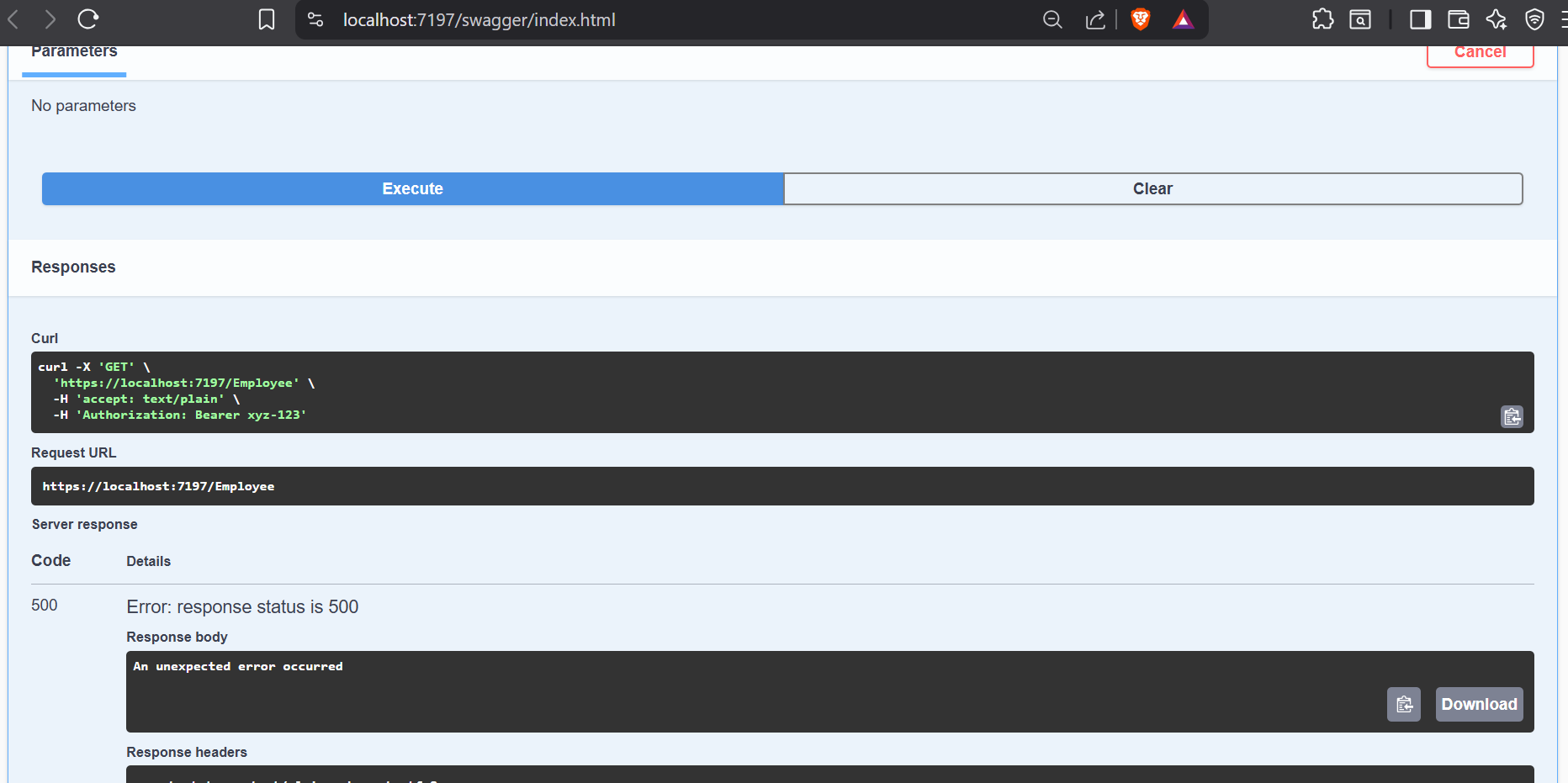
}

}

**Details of exception in log file:**



**Get Method with Exception**



1. **WebAPIHandsOn-4**

**Creating PUT method in controller**

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using WebAPIHandsOn.Filters;

using WebAPIHandsOn.Models;

using System;

using System.Collections.Generic;

using System.Linq;

namespace WebAPIHandsOn.Controllers

{

[ApiController]

[Route("[controller]")]

[TypeFilter(typeof(CustomAuthFilter))]

public class EmployeeController : ControllerBase

{

private readonly ILogger<EmployeeController> \_logger;

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

{

new Employee

{

Id = 1,

Name = "John",

Salary = 50000,

Permanent = true,

Department = new Department { Id = 1, Name = "HR" },

Skills = new List<Skill>

{

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

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

},

DateOfBirth = new DateTime(1990, 1, 1)

}

};

public EmployeeController(ILogger<EmployeeController> logger)

{

\_logger = logger;

}

[HttpGet]

[ProducesResponseType(typeof(List<Employee>), StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public ActionResult<List<Employee>> Get()

{

//throw new Exception("Custom test exception");

return Ok(employees);

}

[HttpPost]

[ProducesResponseType(typeof(Employee), StatusCodes.Status201Created)]

public IActionResult Post([FromBody] Employee emp)

{

employees.Add(emp);

return Created("", emp);

}

[HttpPut("{id}")]

[ProducesResponseType(typeof(Employee), 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;

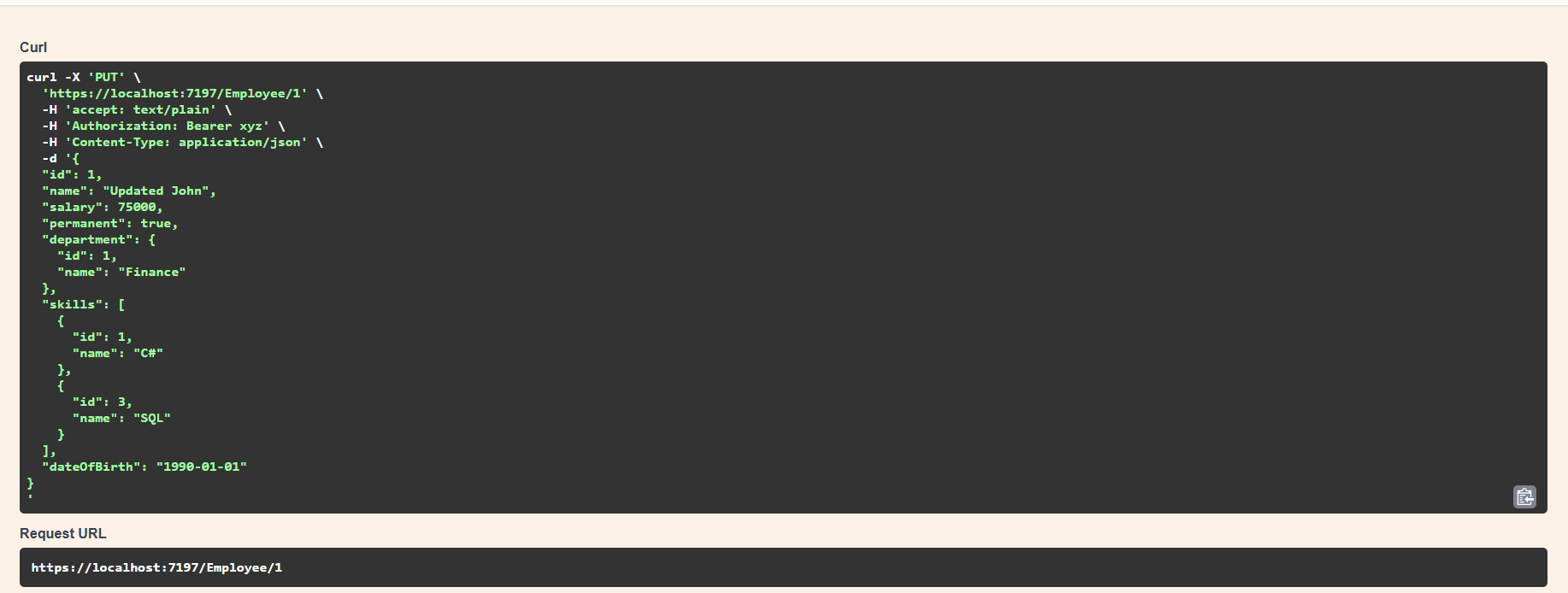
return Ok(existingEmployee);

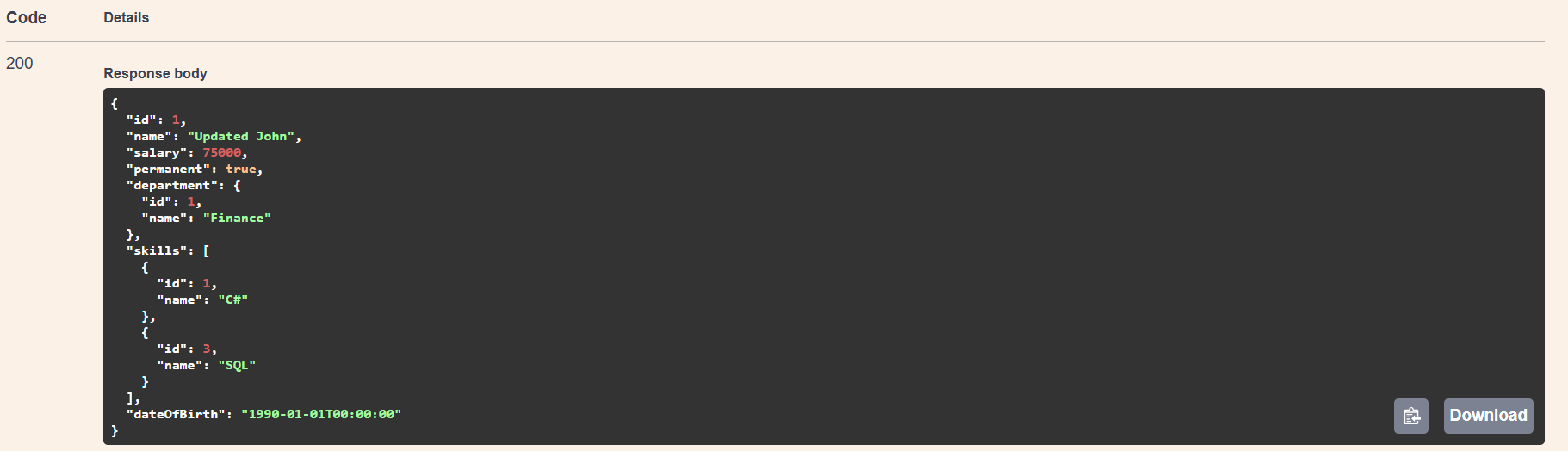
}

}

}

**Making PUT request:**





1. **WebAPI HandsOn-5**

**5.1:JSON Web Token**

**5.2:Use the JWT generated thru the AuthController to be used in POSTMAN request.**

**5.3:Check for JWT expiration**

**5.4:Add the roles to be authorized in the Authorize attribute.**

**AuthController:**

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 WebAPIHandsOn.Controllers

{

[ApiController]

[Route("[controller]")]

public class AuthController : ControllerBase

{

[HttpGet]

[AllowAnonymous]

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("a7^D\*9kL!p4@XzC1qMfZrTgWbE6uYjNv"));//changed due to size constraints of original string(mysupersecretkey)

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);

}

}

}

**EmployeeController:**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace WebAPIHandsOn.Controllers

{

[ApiController]

[Route("[controller]")]

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

public class EmployeeController : ControllerBase

{

[HttpGet]

public IActionResult Get()

{

return Ok(new[] {

new { Id = 1, Name = "John", Salary = 50000 },

new { Id = 2, Name = "Jane", Salary = 60000 }

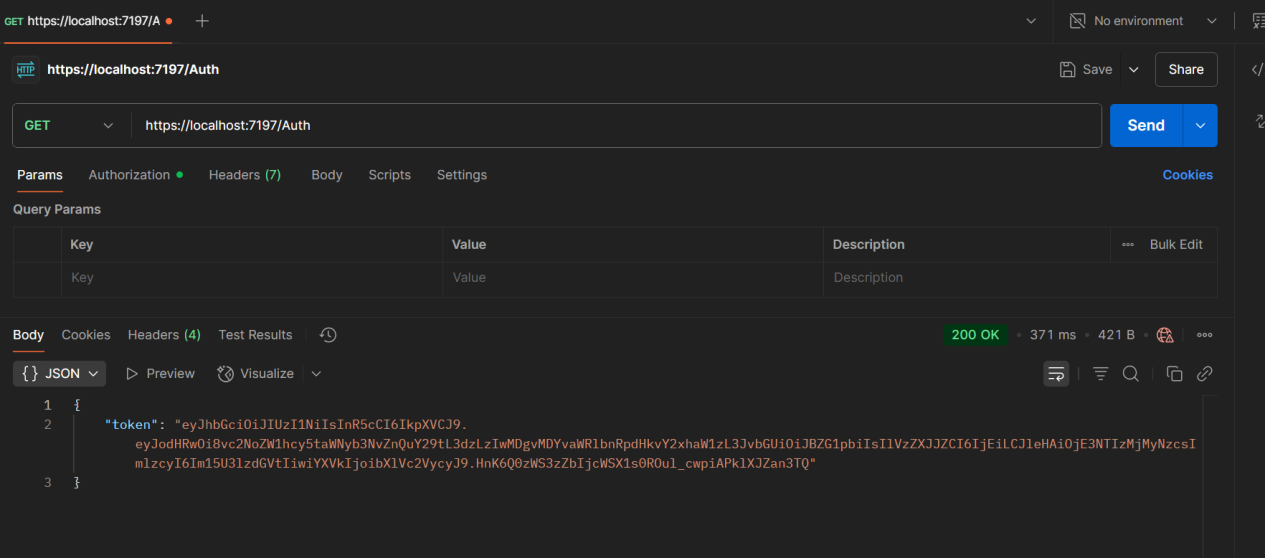
});

}

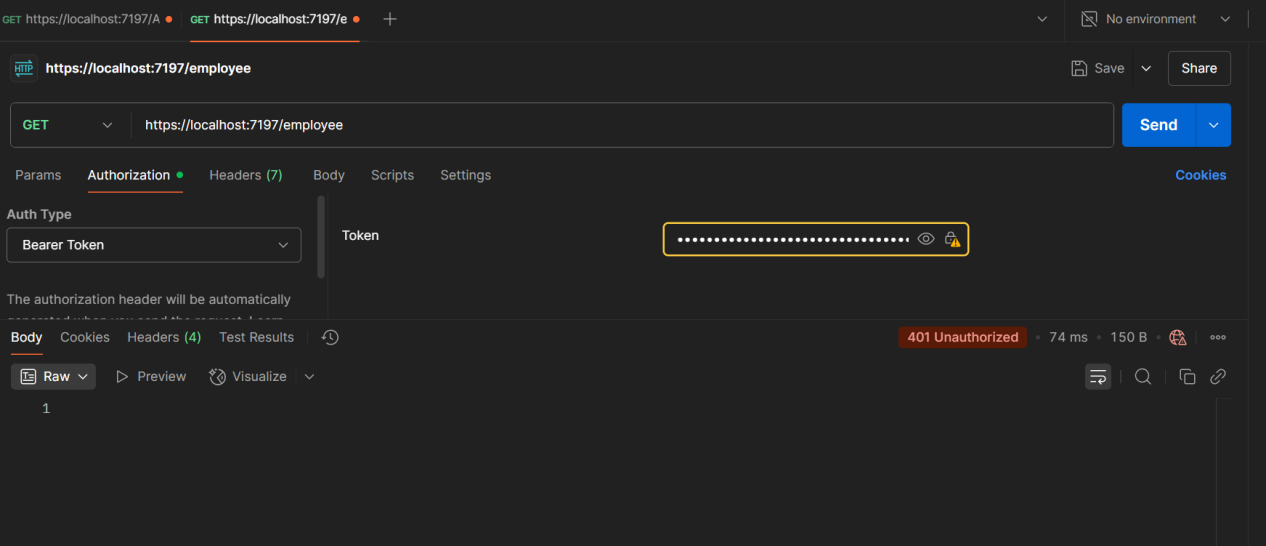
}

}

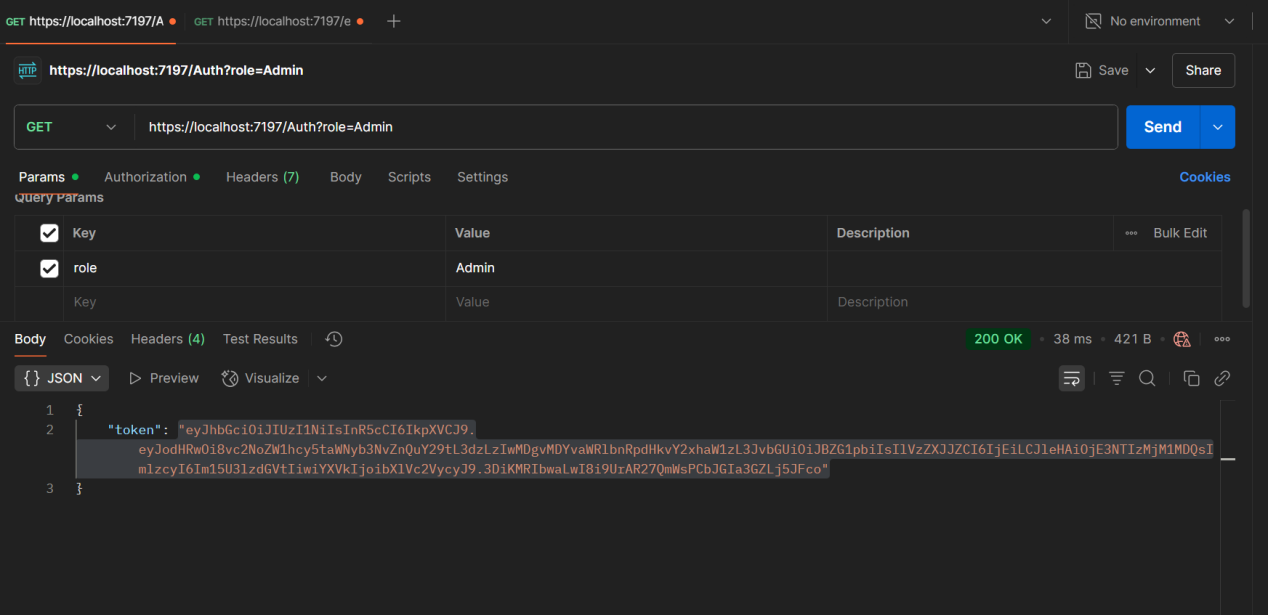
**Getting the token:**

****

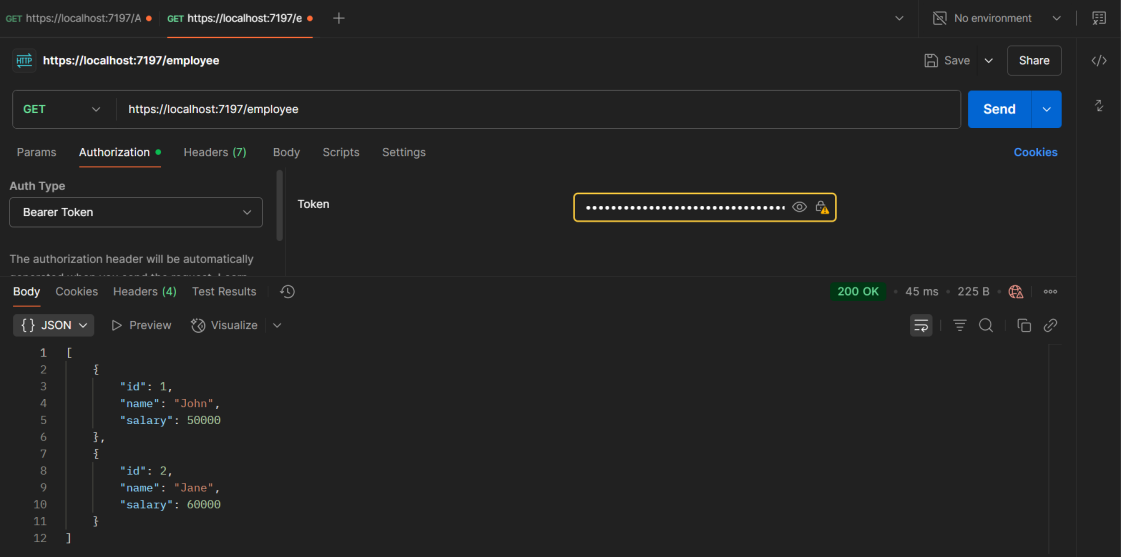
**No Role mentioned(unauthorized):**

****

**Role mentioned and request sent:**

****

**Suucessful request:**

****

**Token expiry (after 2 min):**

