**Q.Explain the usage of FromBody attribute**

Ans: The [FromBody] attribute tells ASP.NET Core to **bind a parameter** in your controller method **from the request body**, instead of the query string, route, or headers.

**CODE:**

Department.cs

namespace WebApi.Models

{

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

}

}

Employee.cs

using System;

namespace WebApi.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; }

}

}

Skill.cs

namespace WebApi.Models

{

public class Skill

{

public int Id { get; set; }

public string Name { get; set; }

}

}

EmployeeController.cs

using Microsoft.AspNetCore.Mvc;

using WebApi.Models;

//using WebApi.Filters;

namespace WebApi.Controllers

{

[ApiController]

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

//[ServiceFilter(typeof(CustomAuthFilter))] // Attach custom auth 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 = "John Doe",

Salary = 50000,

Permanent = true,

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

Skills = new List<Skill>

{

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

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

},

DateOfBirth = new DateTime(1990, 1, 1)

},

new Employee

{

Id = 2,

Name = "Jane Smith",

Salary = 70000,

Permanent = false,

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

Skills = new List<Skill>

{

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

new Skill { Id = 4, Name = "SQL" }

},

DateOfBirth = new DateTime(1985, 5, 15)

}

};

}

[HttpGet]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public ActionResult<List<Employee>> GetStandard()

{

// Uncomment for exception testing:

// throw new Exception("Demo error in GET");

return Ok(\_employees);

}

[HttpPost]

public IActionResult AddEmployee([FromBody] Employee emp)

{

\_employees.Add(emp);

return Ok(emp);

}

[HttpPut("{id}")]

public IActionResult UpdateEmployee(int id, [FromBody] Employee emp)

{

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

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

employee.Name = emp.Name;

employee.Salary = emp.Salary;

employee.Permanent = emp.Permanent;

employee.Department = emp.Department;

employee.Skills = emp.Skills;

employee.DateOfBirth = emp.DateOfBirth;

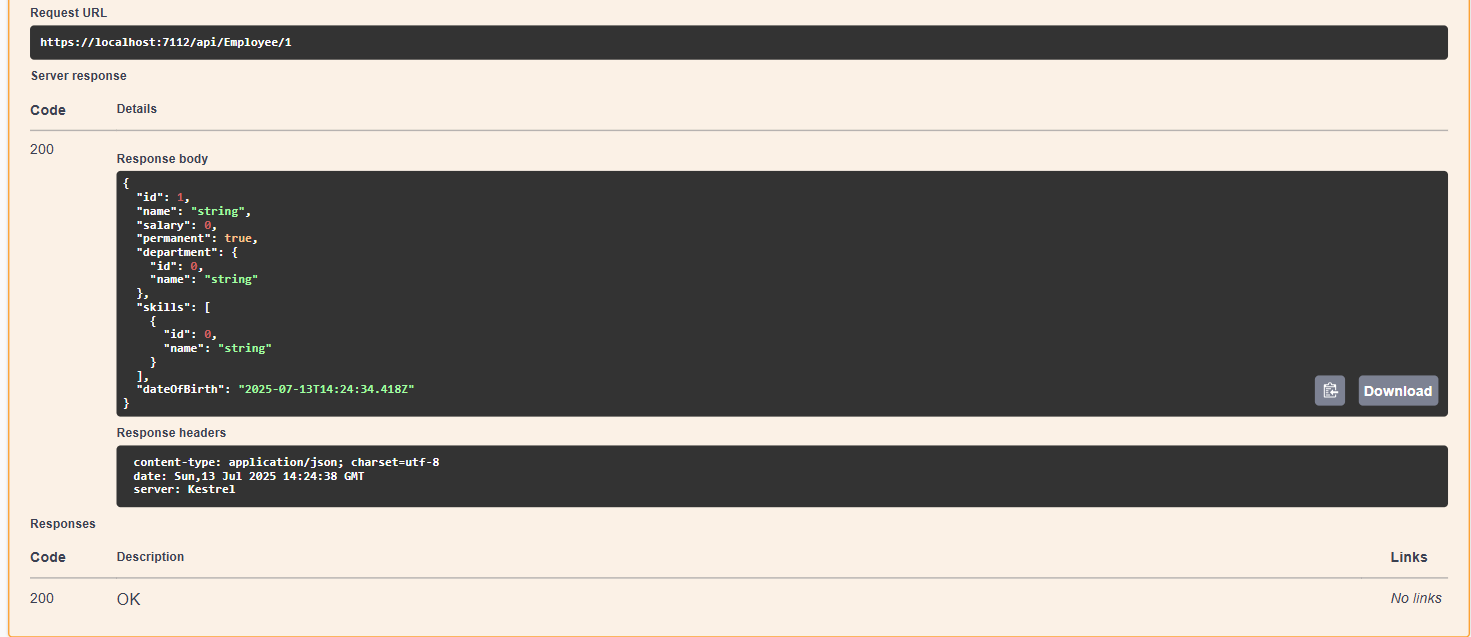
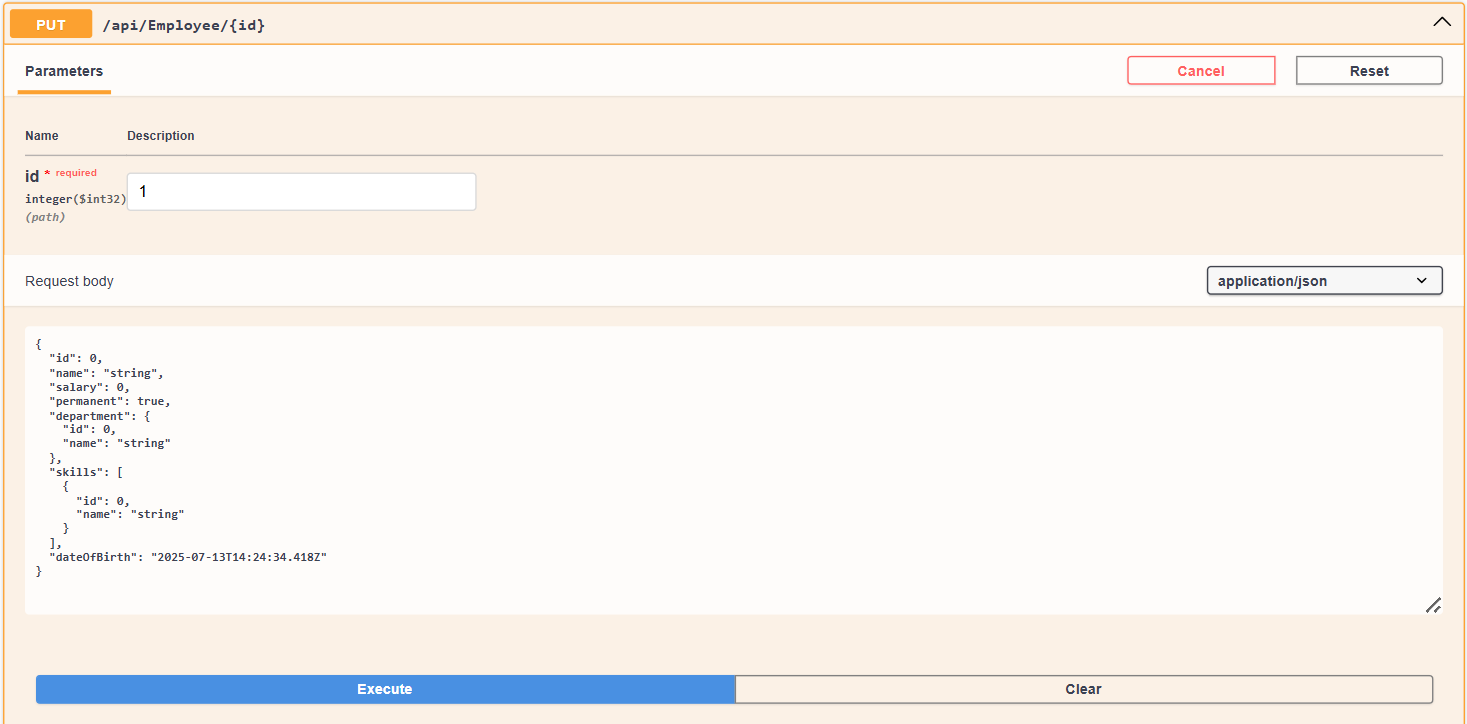
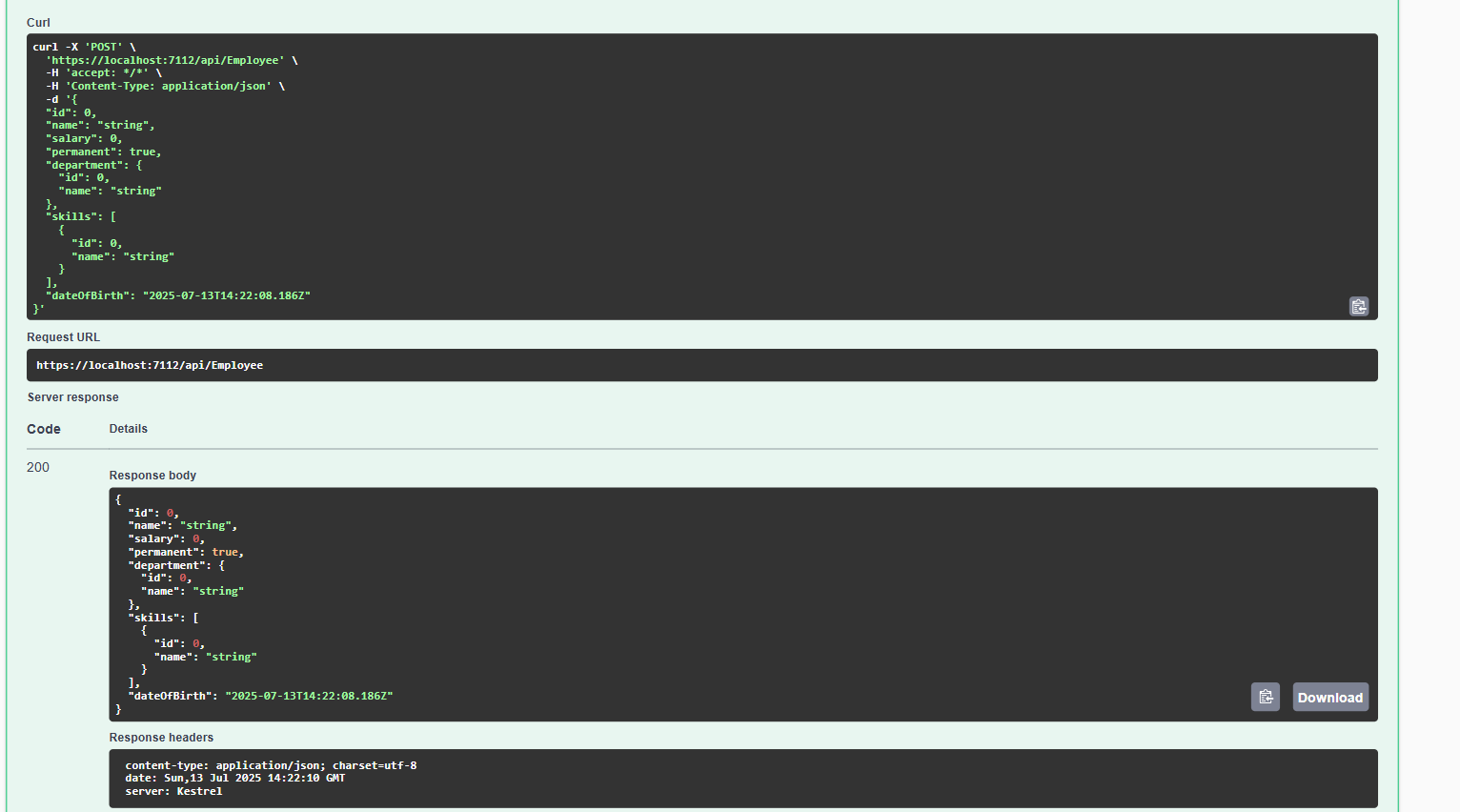
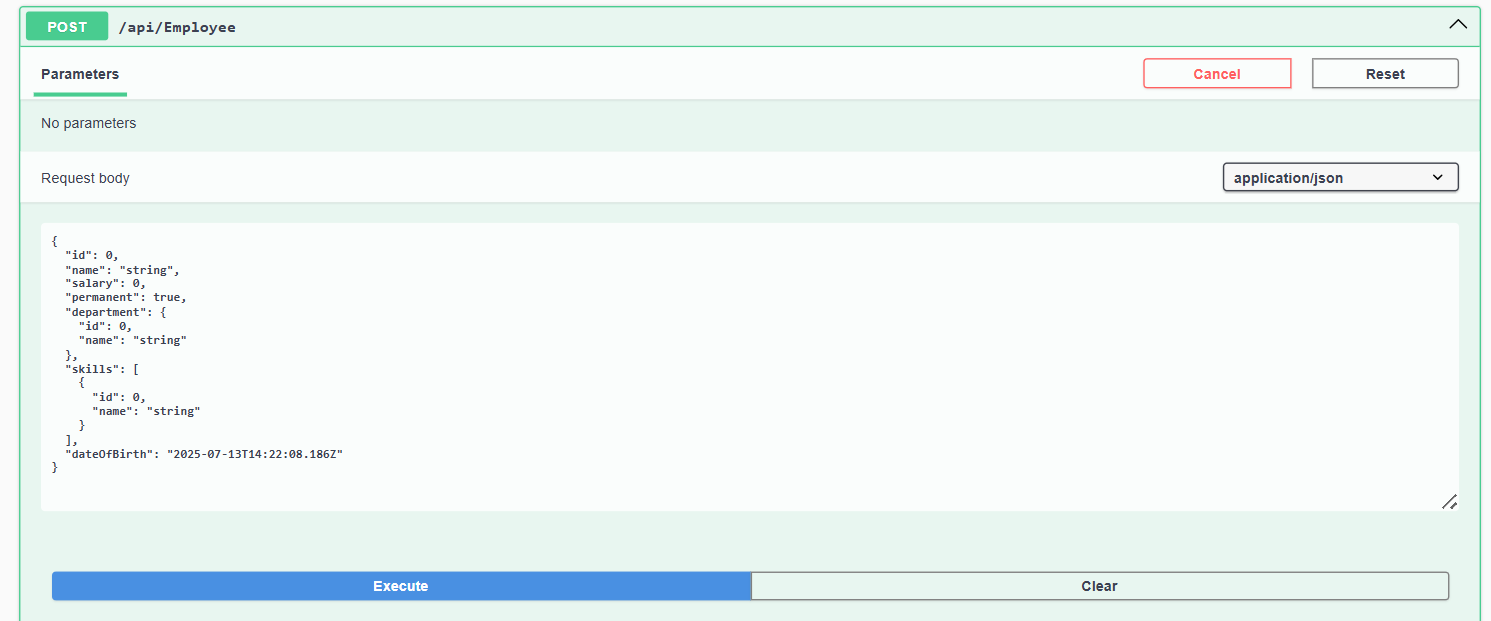
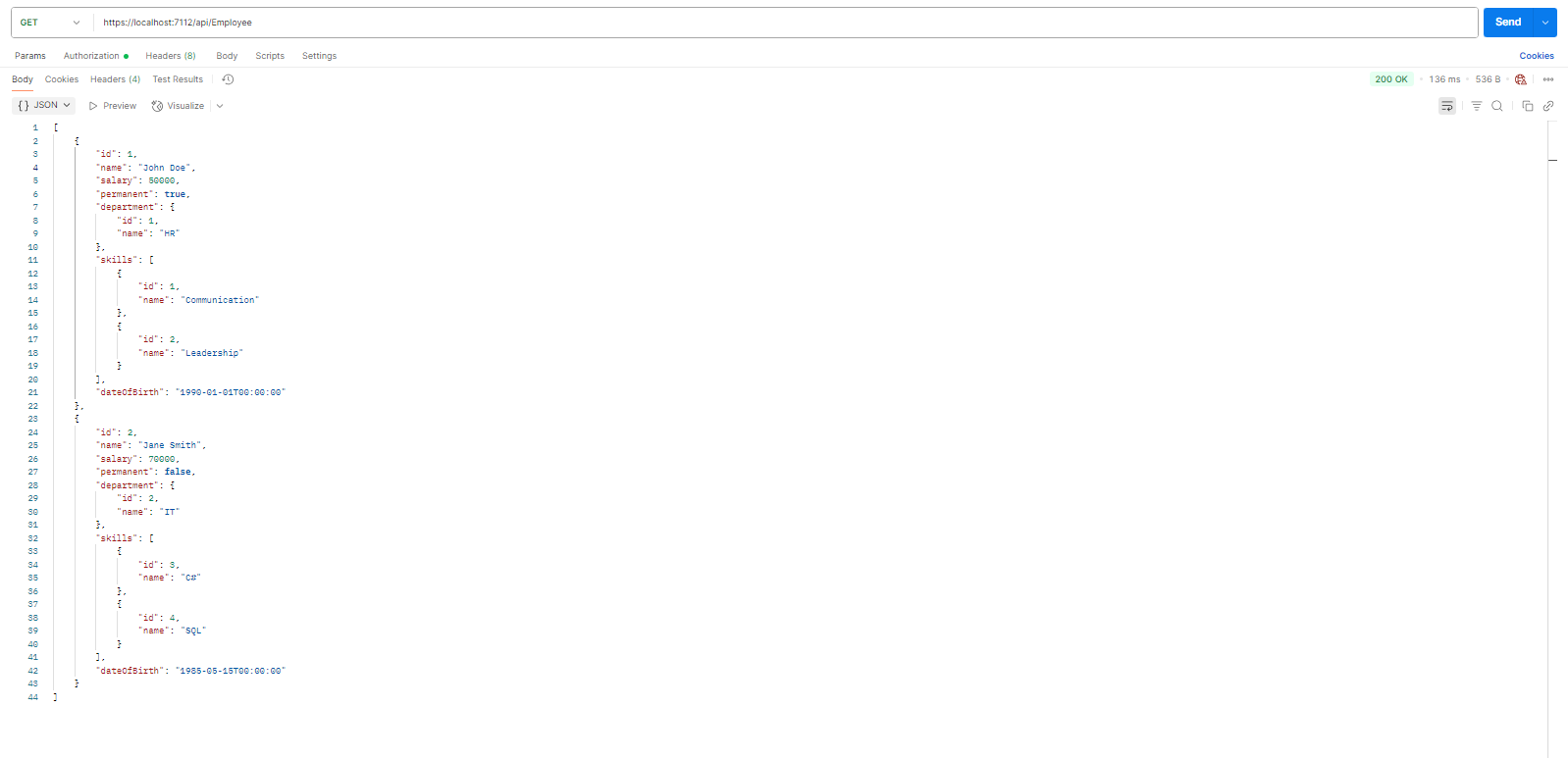
return Ok(employee);

}

}

}

**OUTPUT TEST1:**

****

**Code:**

**CustomAuthFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace WebApi.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

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

if (!hasAuth)

{

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);

}

}

}

**CustomExceptionFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace WebApi.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

var filePath = Path.Combine(Directory.GetCurrentDirectory(), "Logs", "errors.txt");

Directory.CreateDirectory(Path.GetDirectoryName(filePath)!);

File.AppendAllText(filePath,

$"[{DateTime.Now}] {context.Exception.Message}{Environment.NewLine}");

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

{

StatusCode = StatusCodes.Status500InternalServerError

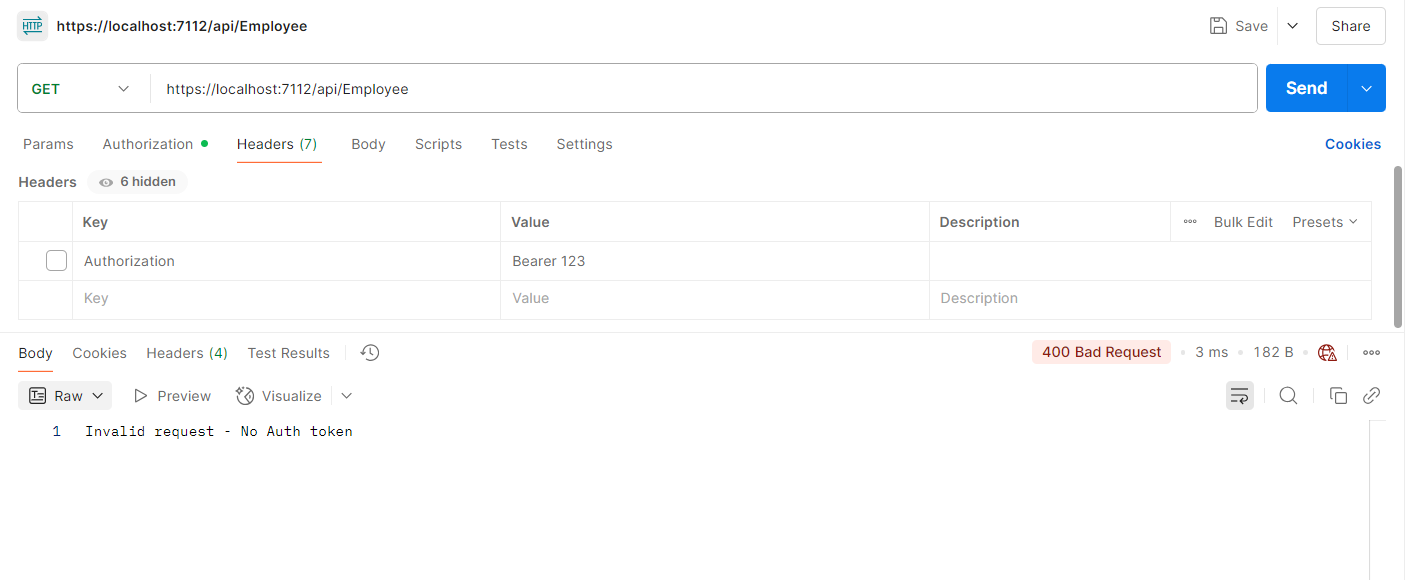
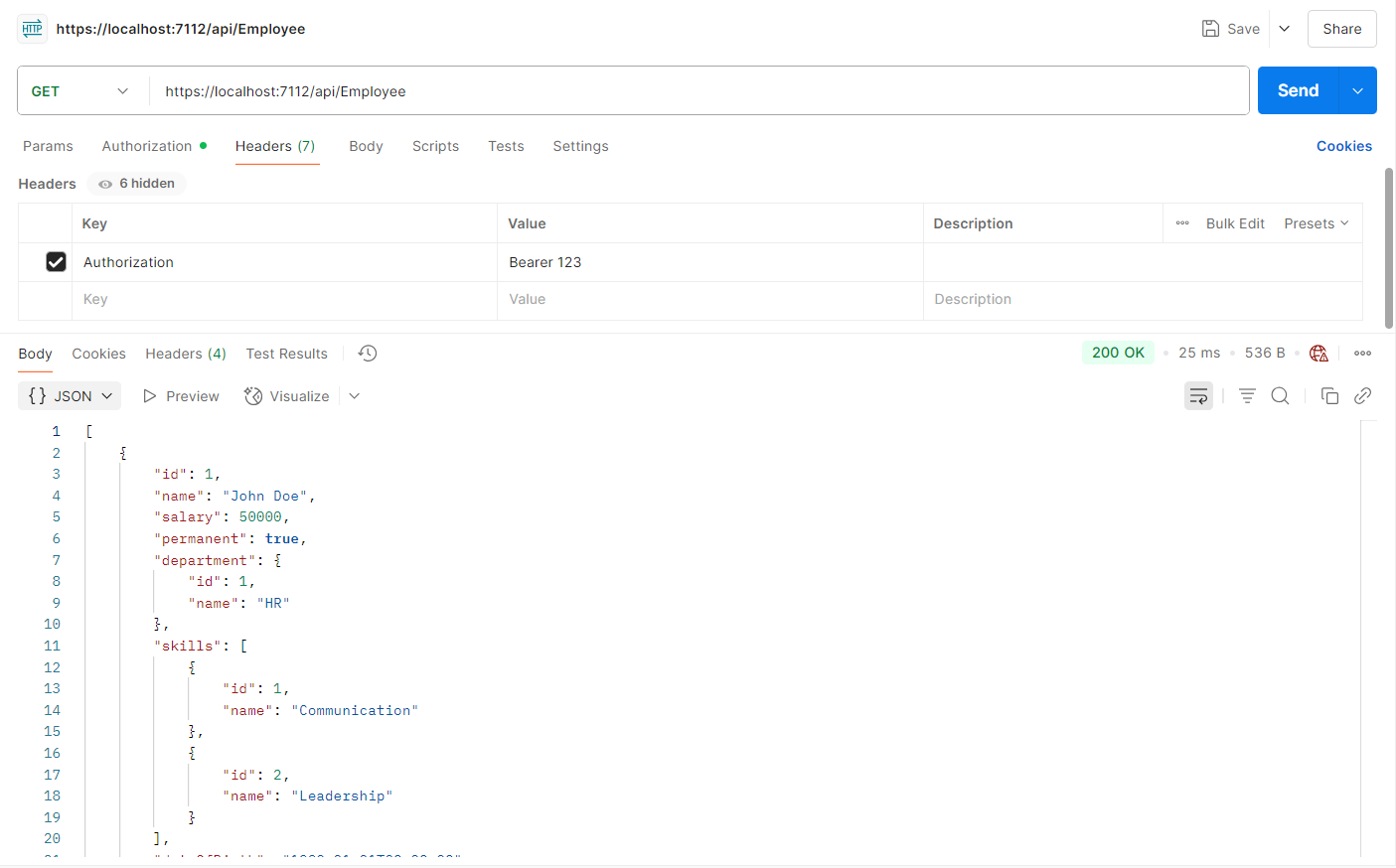
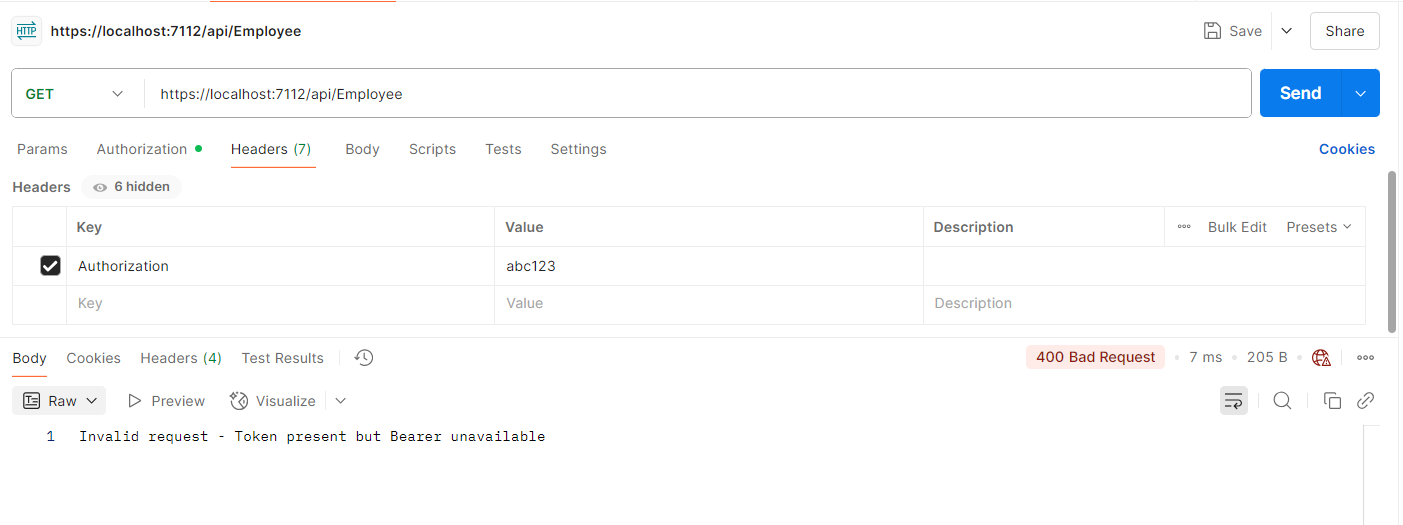
};

}

}

}

**OUTPUT TEST2**

****

**CODE**

**EmployeeController.cs**

using Microsoft.AspNetCore.Mvc;

using WebApi.Models;

using WebApi.Filters;

namespace WebApi.Controllers

{

[ApiController]

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

[ServiceFilter(typeof(CustomAuthFilter))] // Attach custom auth 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 = "John Doe",

Salary = 50000,

Permanent = true,

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

Skills = new List<Skill>

{

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

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

},

DateOfBirth = new DateTime(1990, 1, 1)

},

new Employee

{

Id = 2,

Name = "Jane Smith",

Salary = 70000,

Permanent = false,

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

Skills = new List<Skill>

{

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

new Skill { Id = 4, Name = "SQL" }

},

DateOfBirth = new DateTime(1985, 5, 15)

}

};

}

[HttpGet]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status500InternalServerError)]

public ActionResult<List<Employee>> GetStandard()

{

throw new Exception("Demo error in GET"); // Uncomment this line

// return Ok(\_employees); -- Comment out the normal return

}

[HttpPost]

public IActionResult AddEmployee([FromBody] Employee emp)

{

\_employees.Add(emp);

return Ok(emp);

}

[HttpPut("{id}")]

public IActionResult UpdateEmployee(int id, [FromBody] Employee emp)

{

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

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

employee.Name = emp.Name;

employee.Salary = emp.Salary;

employee.Permanent = emp.Permanent;

employee.Department = emp.Department;

employee.Skills = emp.Skills;

employee.DateOfBirth = emp.DateOfBirth;

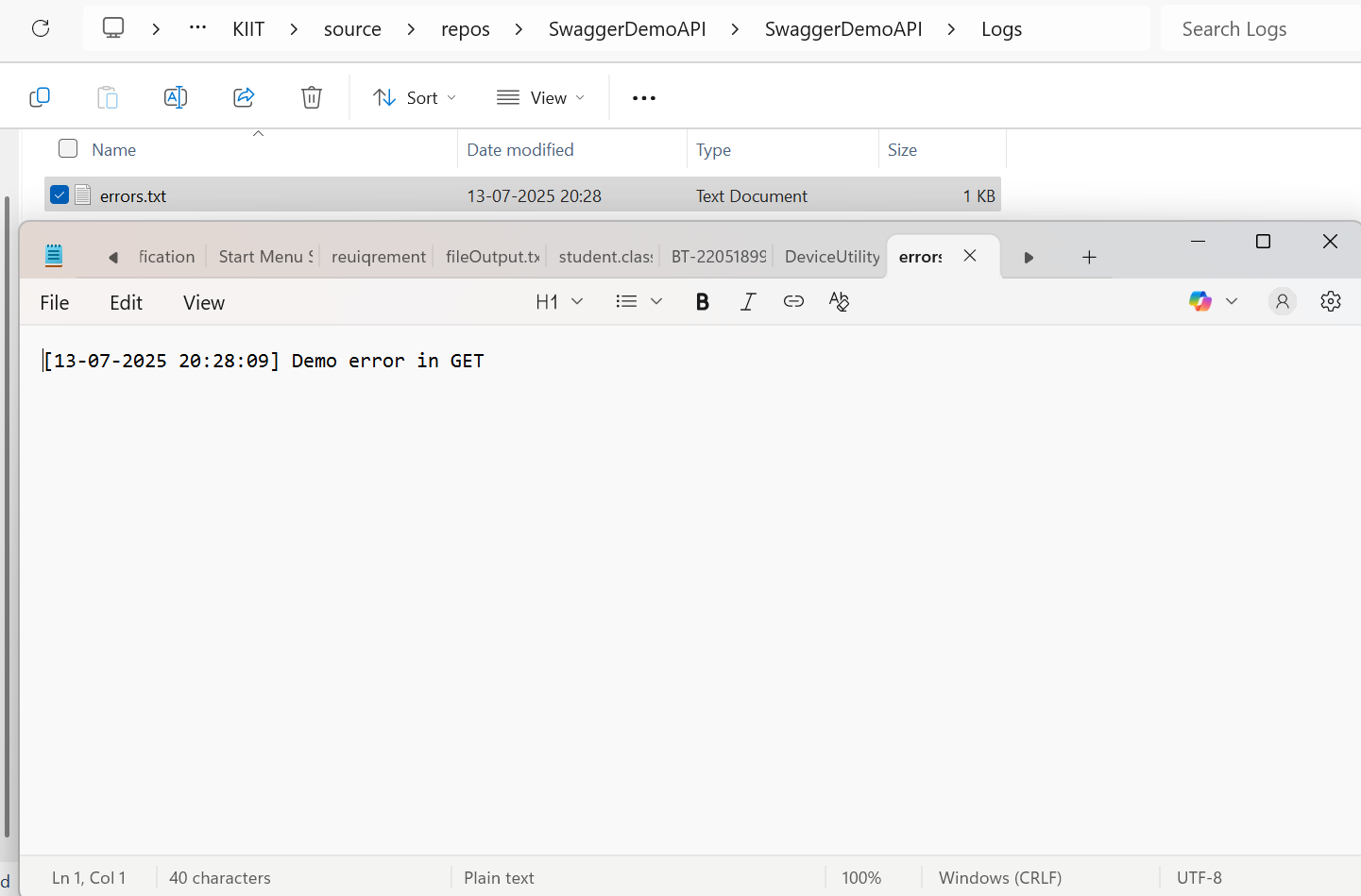
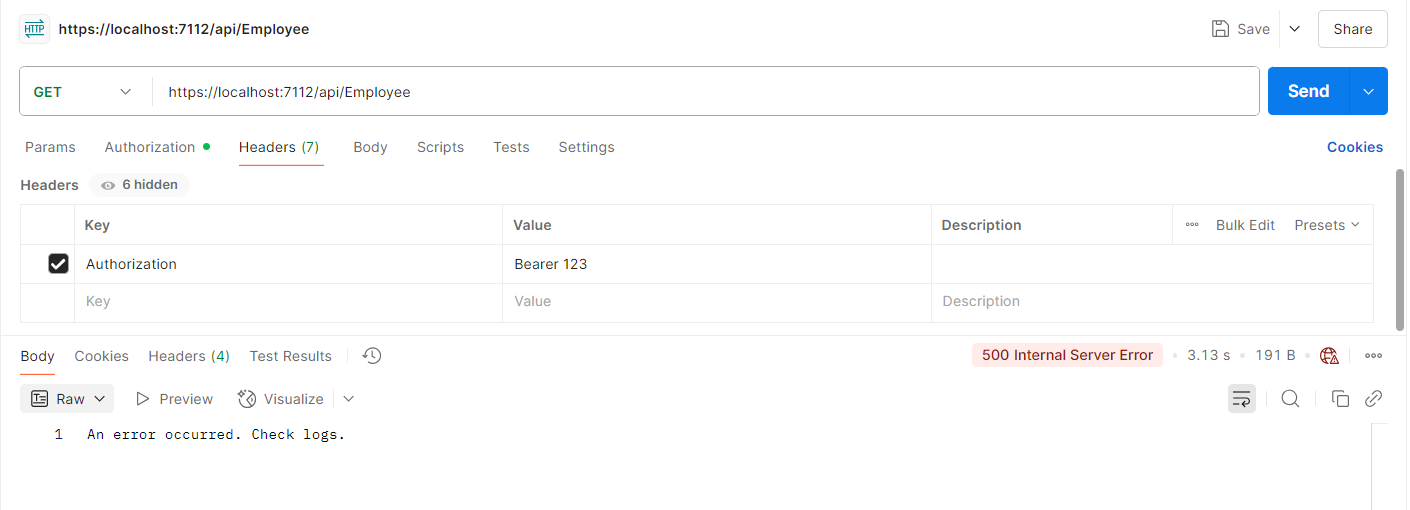
return Ok(employee);

}

}

}

**OUTPUT TEST3**

****