**First Web Api using custom model class**

**Task 1:**

Create a Custom class ‘Employee’ of the below defined structure

Create a new controller - EmployeeController with Read Write actions

**Solution:**

**Employee.cs:**

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; }

}

}

**EmployeeController.cs:**

using Microsoft.AspNetCore.Mvc;

using WebApi.Filters;

using WebApi.Models;

[ApiController]

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

[TypeFilter(typeof(CustomAuthFilter))]

[TypeFilter(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 = "John Doe",

Salary = 50000,

Permanent = true,

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

Skills = new List<Skill>

{

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

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

},

DateOfBirth = new DateTime(1990, 5, 1)

},

new Employee

{

Id = 2,

Name = "Jane Smith",

Salary = 60000,

Permanent = false,

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

Skills = new List<Skill>

{

new Skill { Id = 3, Name = "Communication" }

},

DateOfBirth = new DateTime(1985, 7, 20)

}

};

}

[HttpGet]

[ProducesResponseType(typeof(List<Employee>), 200)]

[ProducesResponseType(500)]

public ActionResult<List<Employee>> Get()

{

throw new Exception("Simulated exception");

return Ok(\_employees);

}

[HttpPost]

public ActionResult Post([FromBody] Employee emp)

{

\_employees.Add(emp);

return Ok("Employee added successfully.");

}

[HttpPut]

public ActionResult Put([FromBody] Employee emp)

{

var existing = \_employees.FirstOrDefault(e => e.Id == emp.Id);

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

existing.Name = emp.Name;

existing.Salary = emp.Salary;

return Ok("Employee updated successfully.");

}

[HttpGet("standard")]

public ActionResult<Employee> GetStandrad()

{

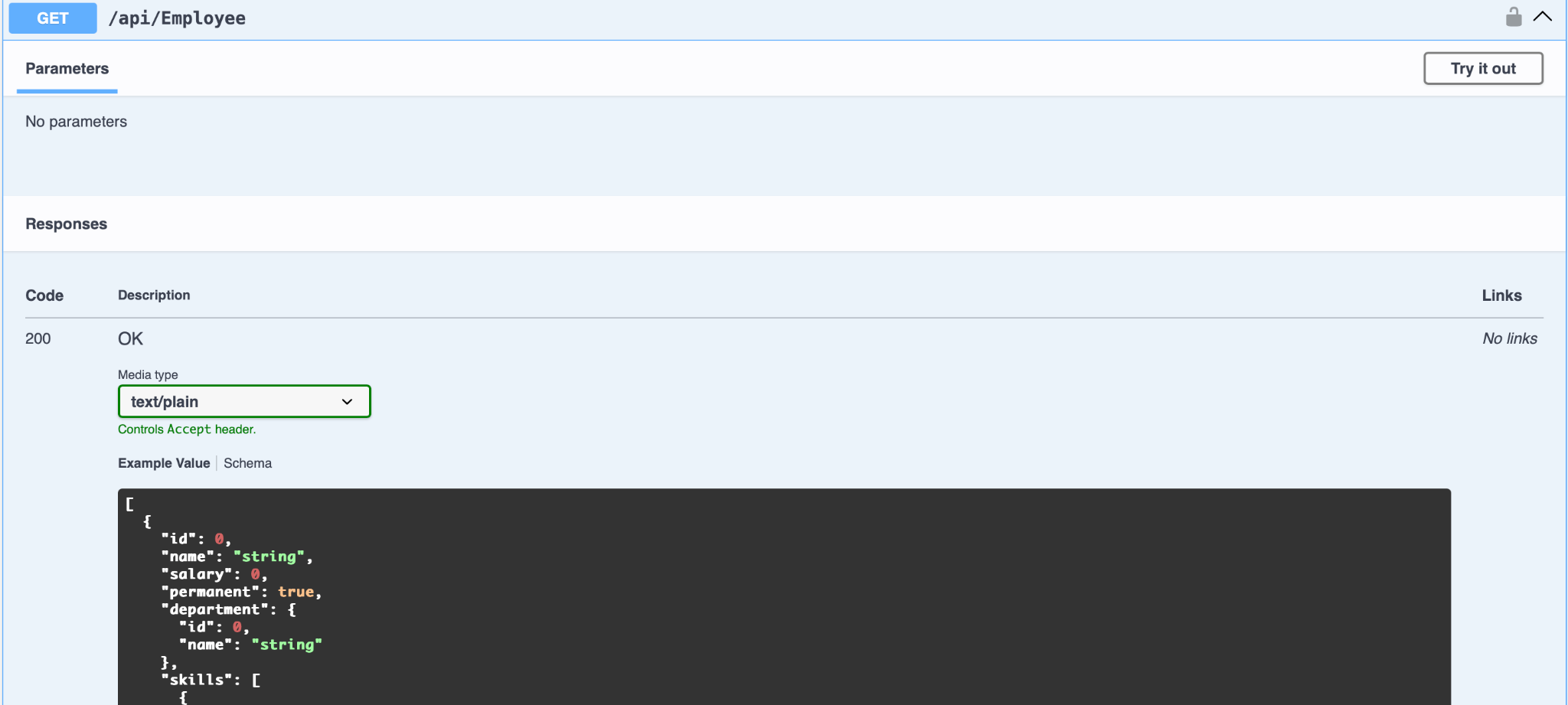
return Ok(\_employees.FirstOrDefault());

}

}

**Local Host:**

**http://localhost:5289/swagger/index.html**

****

**Task 2:**

Create a Custom action filter for Authorization.

**Solution:**

**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** headers = context.HttpContext.Request.Headers;

if (!headers.ContainsKey("Authorization"))

{

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

return;

}

**var** token = headers["Authorization"].ToString();

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

{

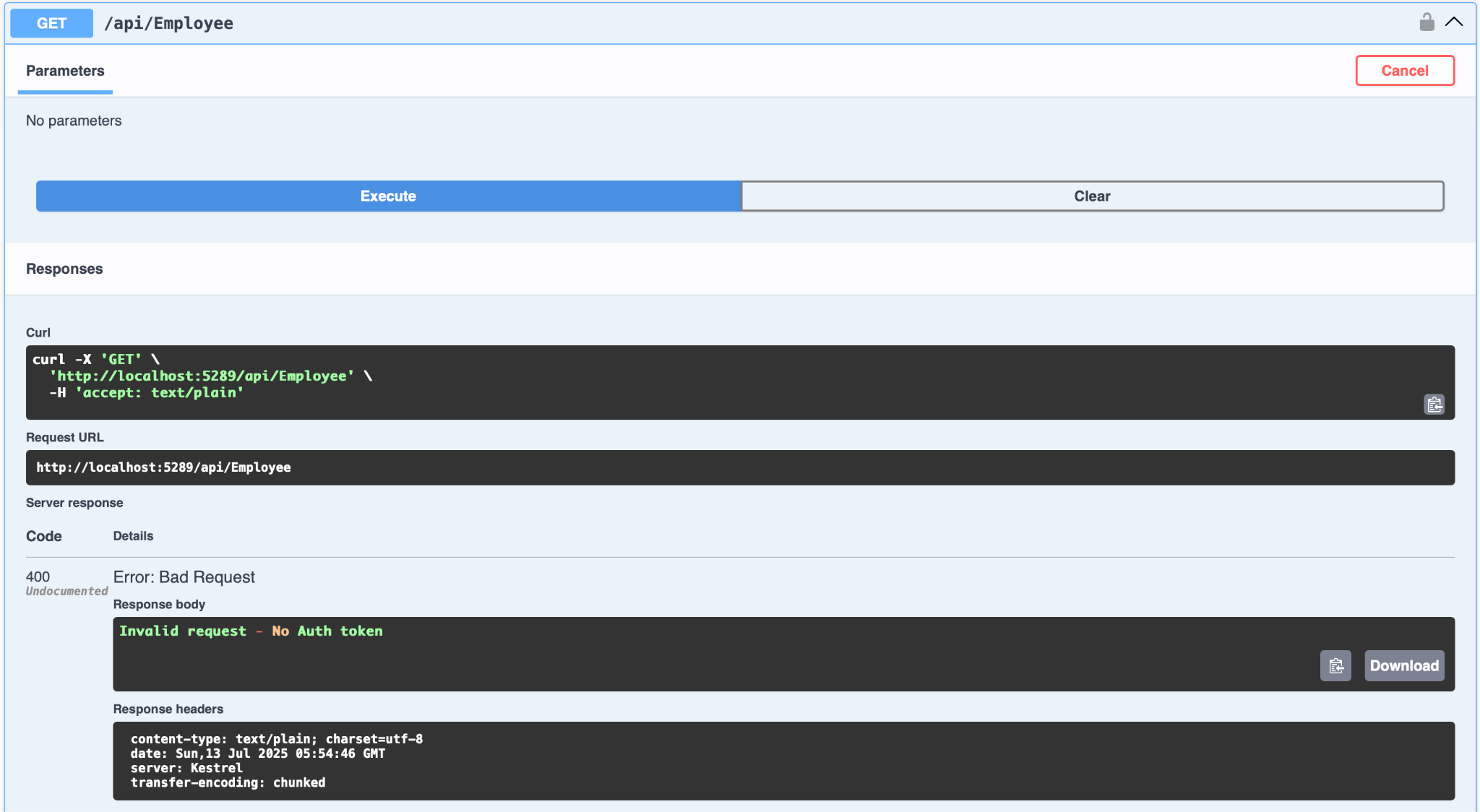
context.Result = new BadRequestObjectResult("Invalid request - Token present but Bearer unavailable");

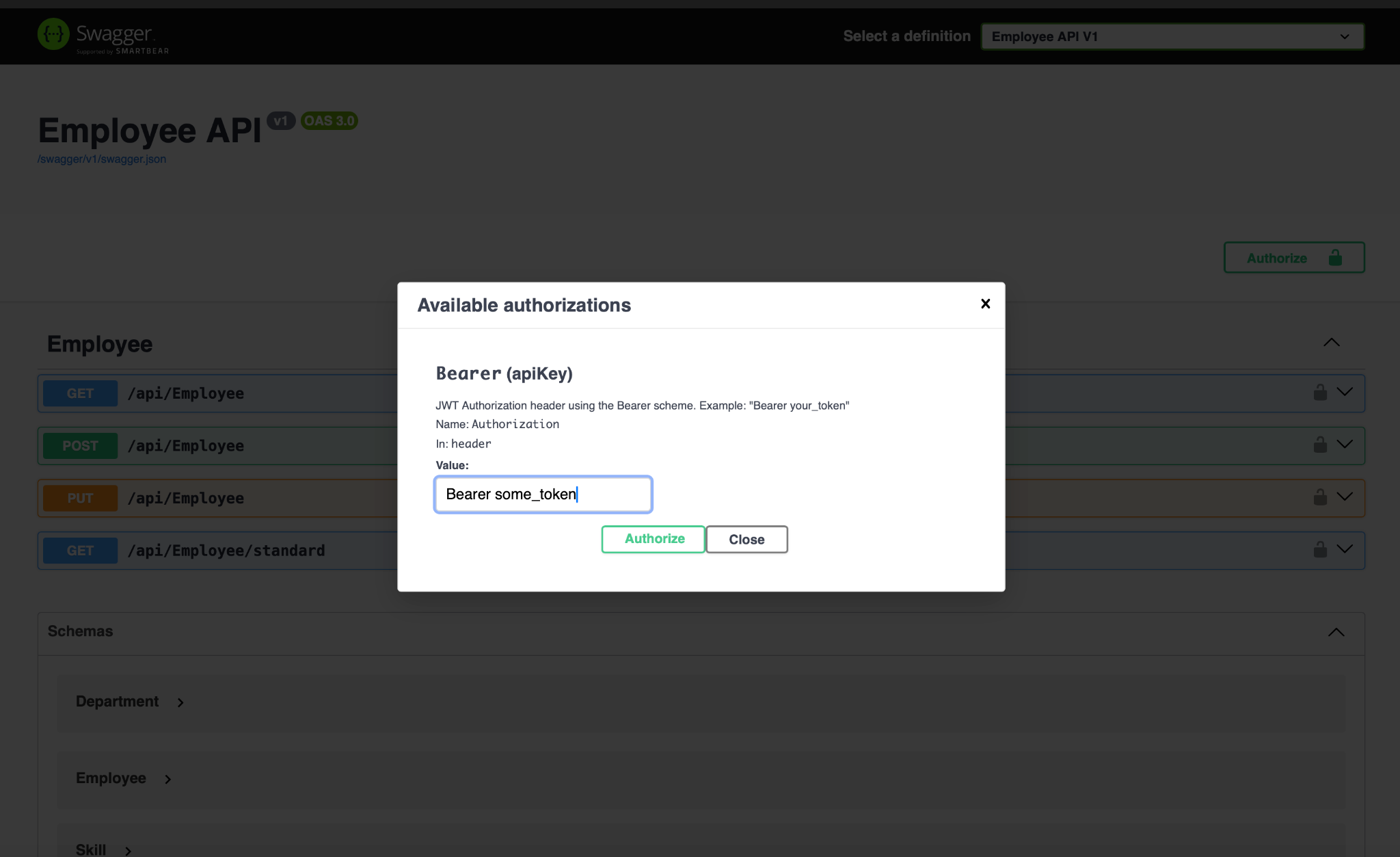
}

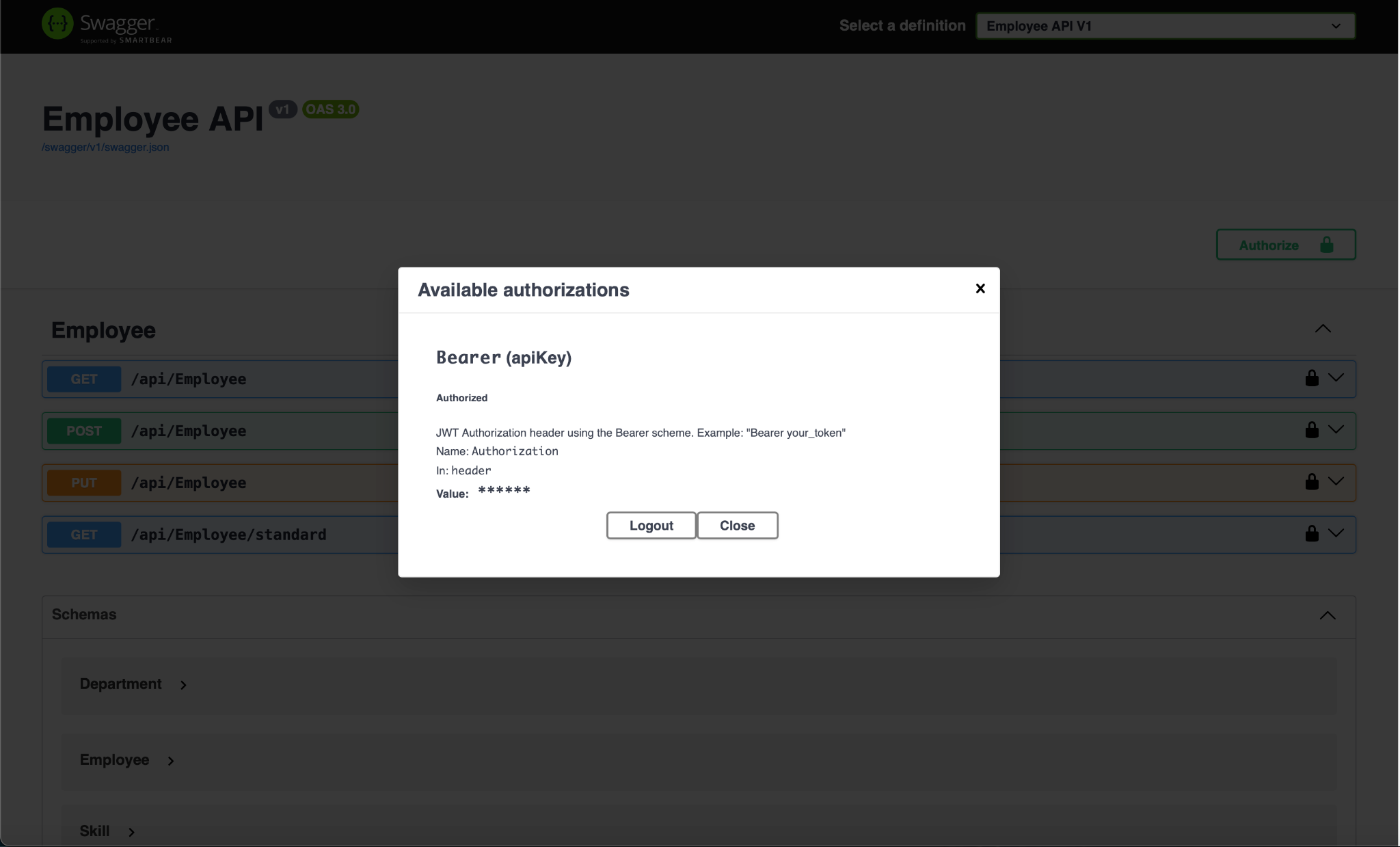
}

}

}

****

****

****

**Task 3:**

Create a Custom exception filter for Authorization.

**Solution:**

**CustomExceptionFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System.IO;

**namespace** WebApi.Filters

{

**public** **class** CustomExceptionFilter : IExceptionFilter

{

**public** void OnException(ExceptionContext context)

{

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

Directory.CreateDirectory(errorLogPath);

string filePath = Path.Combine(errorLogPath, "errors.txt");

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

File.AppendAllText(filePath, exceptionMessage);

context.Result = new ObjectResult("An error occurred. Please contact support.")

{

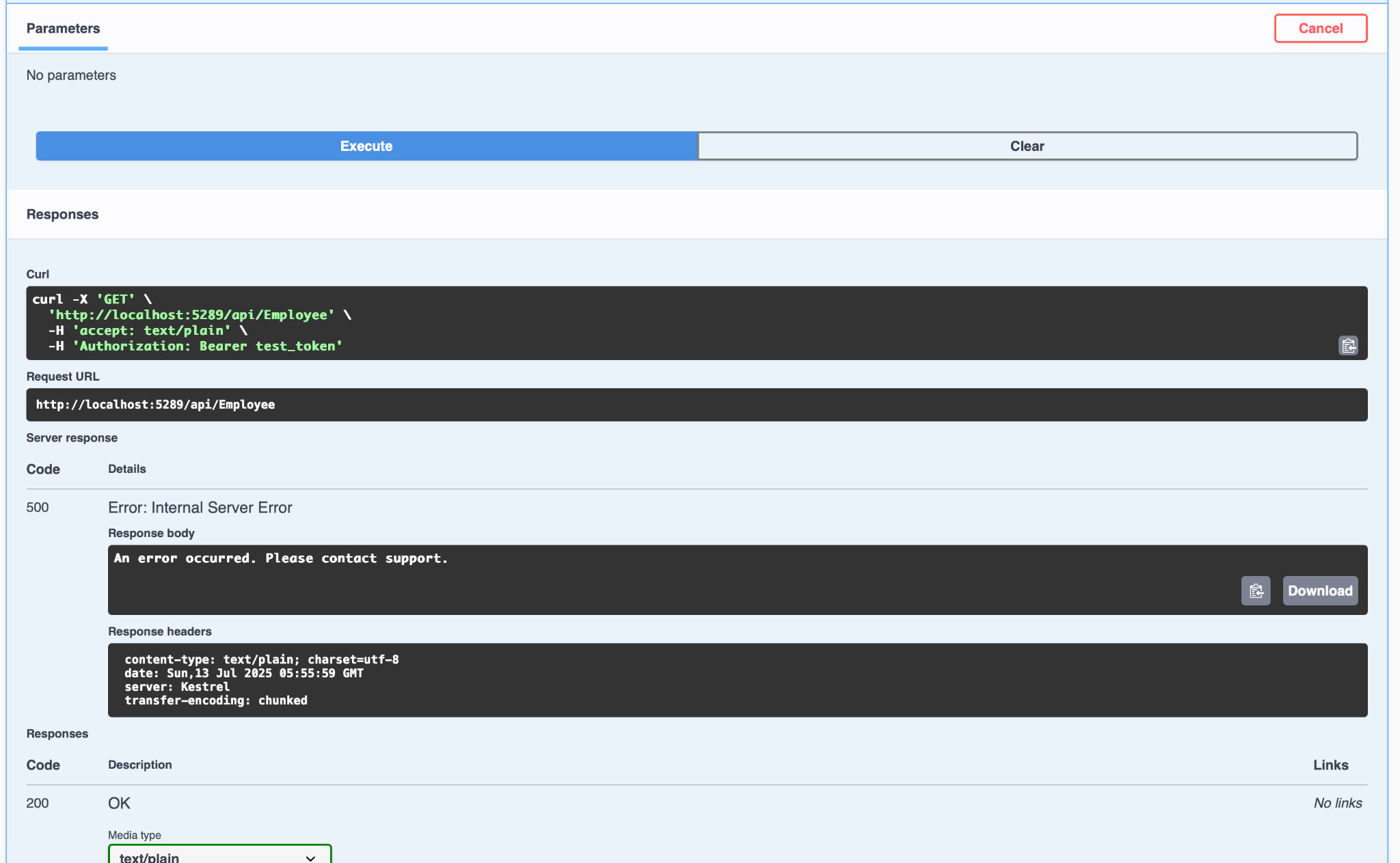
StatusCode = 500

};

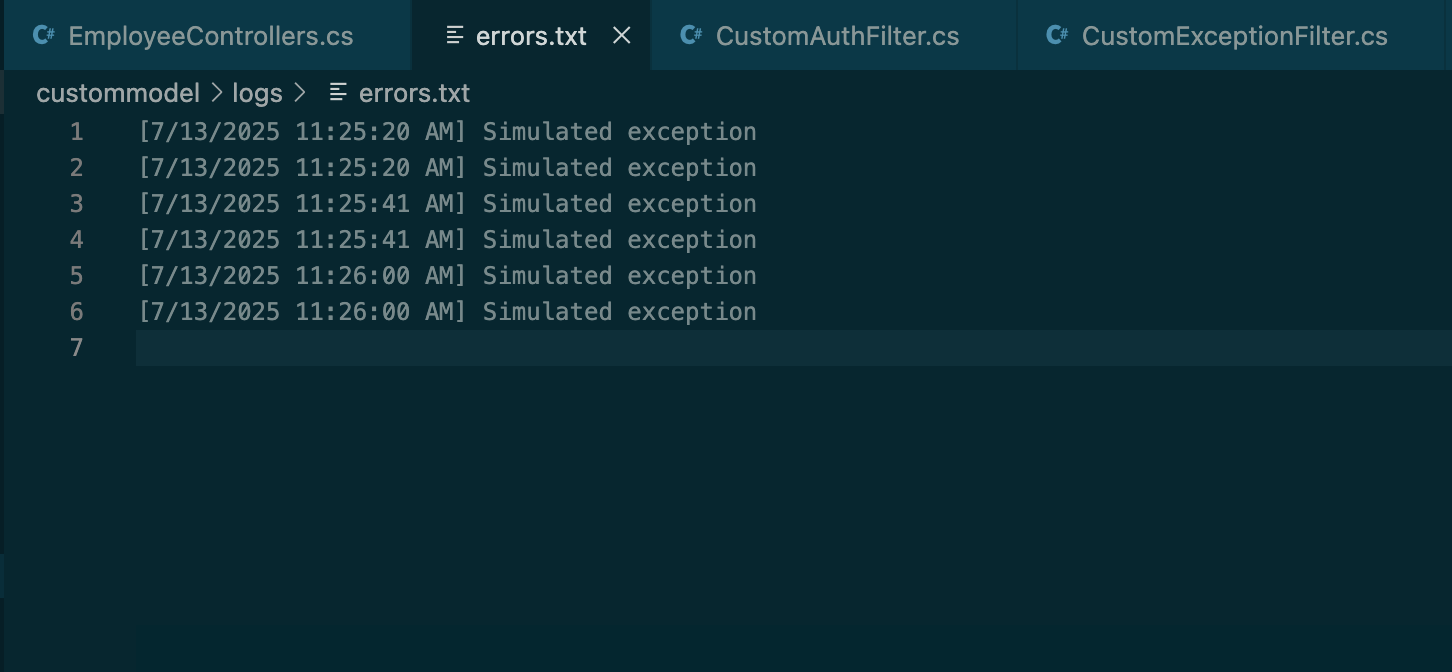
}

}

}



**errors.txt**

****