**CODE:**

**Department.cs**

namespace EmployeeWebApi.Models

{

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

}

}

**Employee.cs**

using System;

using System.Collections.Generic;

namespace EmployeeWebApi.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 EmployeeWebApi.Models

{

public class Skill

{

public int Id { get; set; }

public string Name { get; set; }

}

}

**EmployeeController.cs**

using EmployeeWebApi.Filters;

using EmployeeWebApi.Models;

using Microsoft.AspNetCore.Mvc;

namespace EmployeeWebApi.Controllers

{

[ApiController]

[Route("[controller]")]

[ServiceFilter(typeof(CustomAuthFilter))]

public class EmployeeController : ControllerBase

{

private readonly List<Employee> \_employees;

public EmployeeController()

{

\_employees = GetStandardEmployeeList();

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 1,

Name = "John",

Salary = 50000,

Permanent = true,

DateOfBirth = new DateTime(1990, 5, 1),

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

Skills = new List<Skill>

{

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

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

}

}

};

}

[HttpGet("standard")]

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

public ActionResult<List<Employee>> GetStandard()

{

return \_employees;

}

[HttpPost]

public IActionResult AddEmployee([FromBody] Employee employee)

{

\_employees.Add(employee);

return Ok(\_employees);

}

[HttpPut("{id}")]

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

{

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

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

employee.Name = updated.Name;

employee.Salary = updated.Salary;

employee.Permanent = updated.Permanent;

employee.DateOfBirth = updated.DateOfBirth;

employee.Department = updated.Department;

employee.Skills = updated.Skills;

return Ok(employee);

}

[HttpGet("error")]

[ProducesResponseType(500)]

public IActionResult ThrowError()

{

throw new Exception("This is a test exception.");

}

}

}

**CustomAuthFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace EmployeeWebApi.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");

}

}

}

}

**CustomExceptionFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace EmployeeWebApi.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

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

File.AppendAllText("errors.log", errorMessage);

context.Result = new ObjectResult("Internal server error occurred.")

{

StatusCode = 500

};

}

}

}

**Program.cs**

using EmployeeWebApi.Filters;

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers(options =>

{

options.Filters.Add<CustomAuthFilter>();

options.Filters.Add<CustomExceptionFilter>();

});

builder.Services.AddScoped<CustomAuthFilter>();

builder.Services.AddScoped<CustomExceptionFilter>();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo { Title = "Employee API", Version = "v1" });

c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

In = ParameterLocation.Header,

Description = "Please insert Bearer token",

Name = "Authorization",

Type = SecuritySchemeType.ApiKey

});

c.AddSecurityRequirement(new OpenApiSecurityRequirement {

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

new string[] { }

}});

});

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseAuthorization();

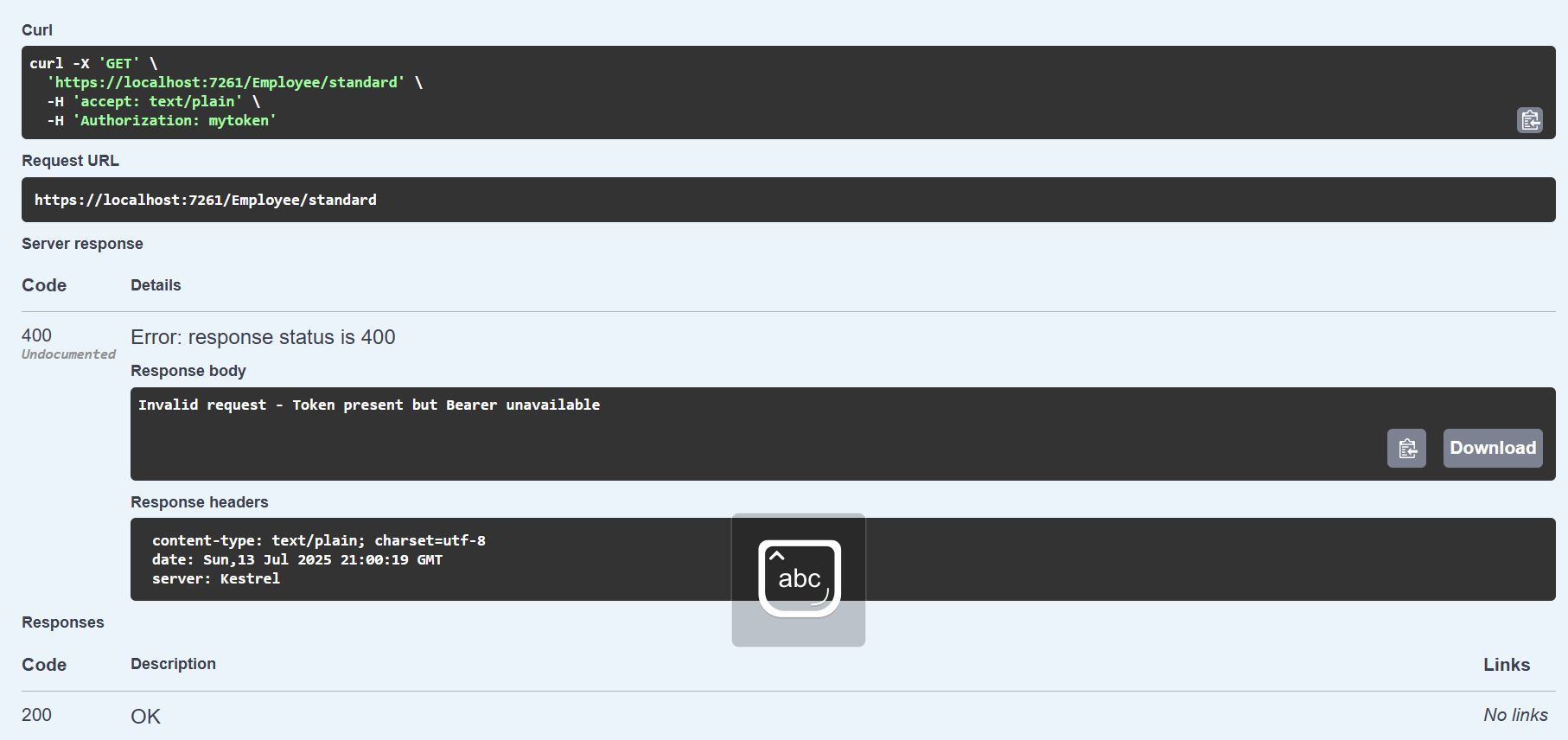
app.MapControllers();

app.Run();

**OUTPUT**

**When not Authorized**

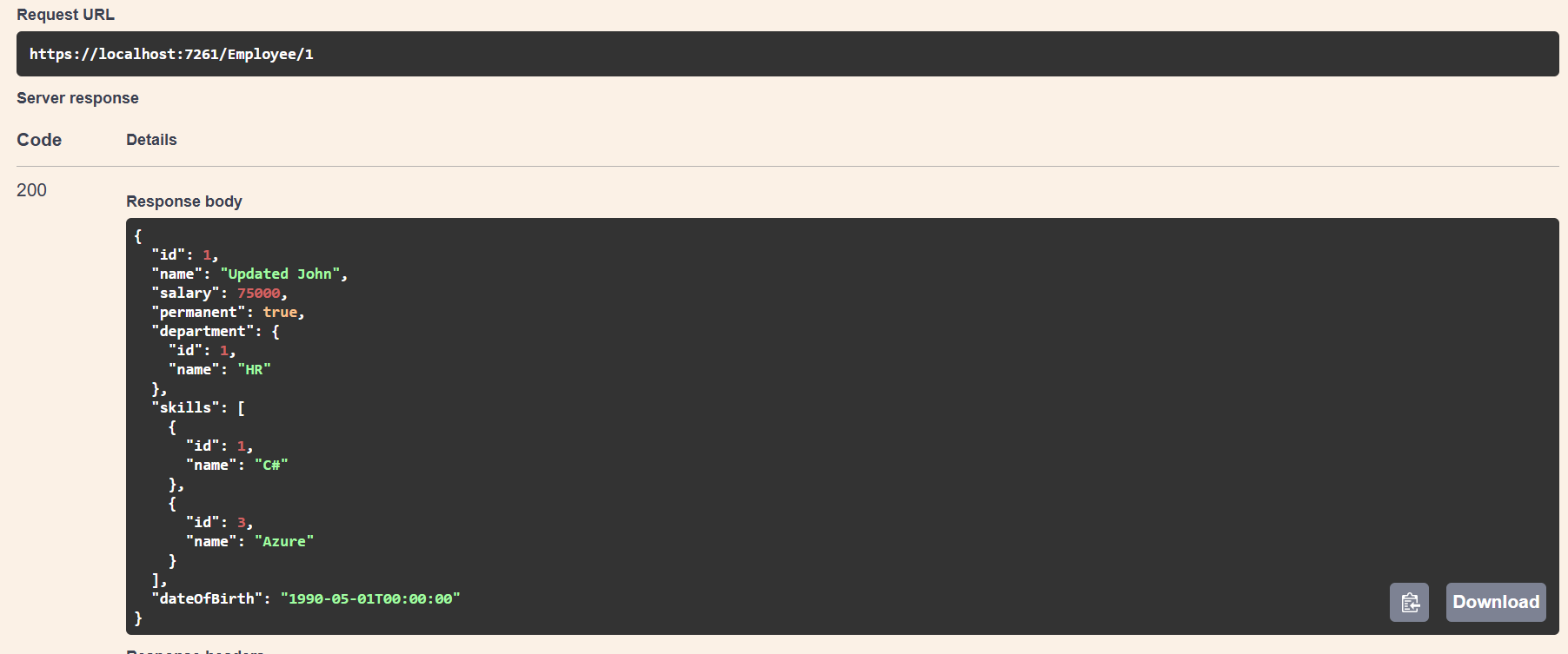
****

**When Authorized but does not contain ‘Bearer’**

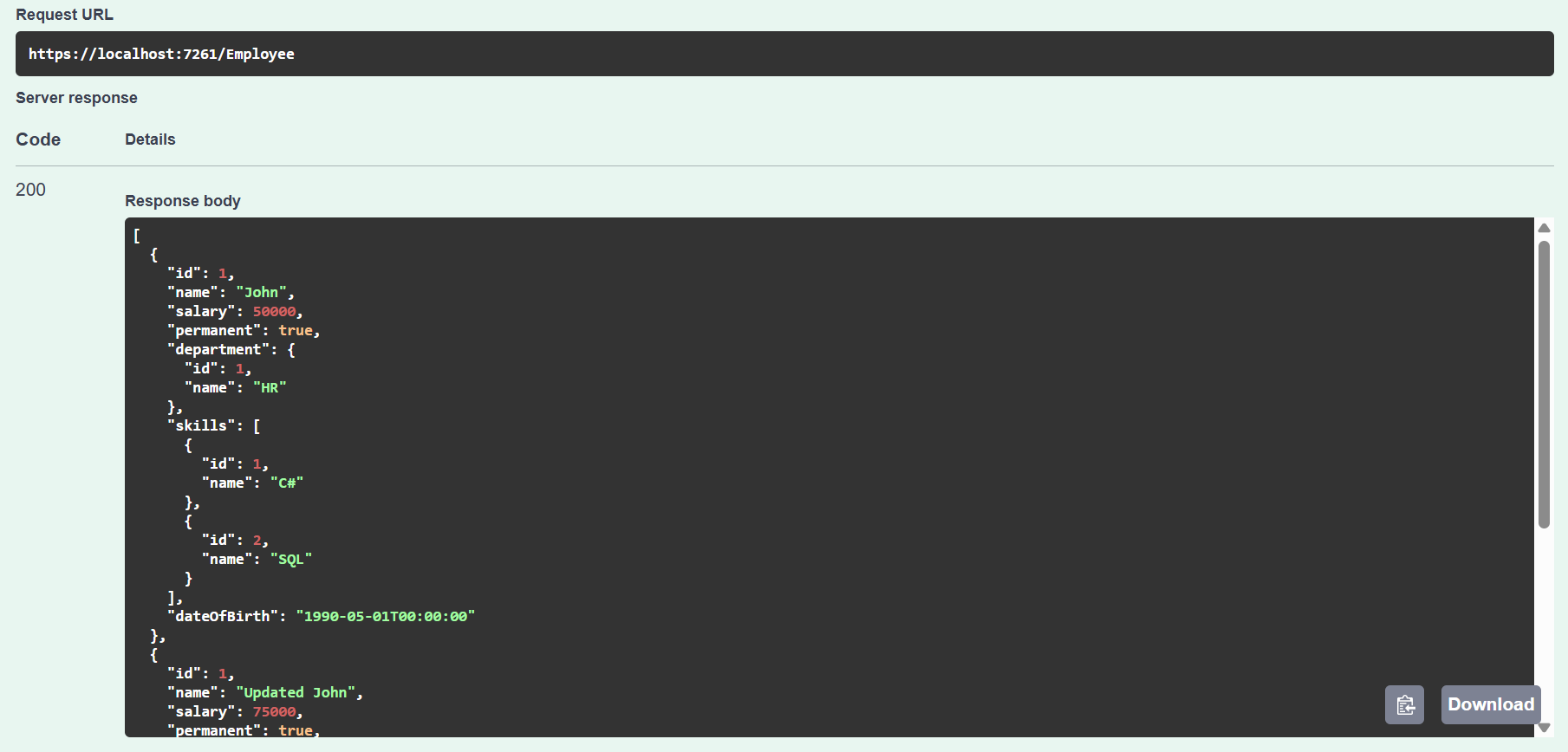
**GET Operation**

****

**PUT Operation**

****

**Post Opertation**

****