**WEBAPI 04**

**Aim:**

Web Api CRUD operation

**Department.cs:**

namespace EmployeeCrudApi.Models

{

    public class Department

    {

        public int Id { get; set; }

        public string? Name { get; set; }

    }

}

**Employee.cs:**

using System;

using System.Collections.Generic;

namespace EmployeeCrudApi.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; }

    }

}

**Skills.cs:**

namespace EmployeeCrudApi.Models

{

    public class Skill

    {

        public int Id { get; set; }

        public string? Name { get; set; }

    }

}

**EmployeeController.cs:**

using EmployeeCrudApi.Models;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

namespace EmployeeCrudApi.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    public class EmployeeController : ControllerBase

    {

        private static List<Employee> \_employees = new List<Employee>

        {

            new Employee

            {

                Id = 1,

                Name = "Alice",

                Salary = 60000,

                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, 5, 21)

            },

            new Employee

            {

                Id = 2,

                Name = "Bob",

                Salary = 55000,

                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(1988, 11, 3)

            }

        };

        [HttpGet]

        [ProducesResponseType(StatusCodes.Status200OK)]

        public ActionResult<List<Employee>> GetAll()

        {

            return Ok(\_employees);

        }

        [HttpGet("{id}")]

        [ProducesResponseType(StatusCodes.Status200OK)]

  [ProducesResponseType(StatusCodes.Status404NotFound)]

        public ActionResult<Employee> GetById(int id)

        {

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

            if (employee == null)

                return NotFound();

            return Ok(employee);

        }

        [HttpPost]

 [ProducesResponseType(StatusCodes.Status201Created)]  [ProducesResponseType(StatusCodes.Status400BadRequest)]

        public ActionResult<Employee> Create([FromBody] Employee newEmployee)

        {

            if (newEmployee == null)

                return BadRequest("Employee data is required.");

            newEmployee.Id = \_employees.Max(e => e.Id) + 1;

            \_employees.Add(newEmployee);

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

        }

        [HttpPut("{id}")]

        [ProducesResponseType(StatusCodes.Status200OK)]

        [ProducesResponseType(StatusCodes.Status400BadRequest)]

        public ActionResult<Employee> Update(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);

        }

        [HttpDelete("{id}")]

 [ProducesResponseType(StatusCodes.Status204NoContent)]      [ProducesResponseType(StatusCodes.Status400BadRequest)]

        public IActionResult Delete(int id)

        {

            if (id <= 0)

                return BadRequest("Invalid employee id");

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

            if (employee == null)

                return BadRequest("Invalid employee id");

            \_employees.Remove(employee);

            return NoContent();

        }

    }

**Program.cs:**

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.MapControllers();

app.Run();

**Startup.cs:**

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.OpenApi.Models;

namespace EmployeeCrudApi

{

    public class Startup

    {

        // This method gets called by the runtime. Use this method to add services to the container.

        public void ConfigureServices(IServiceCollection services)

        {

            services.AddControllers();

            // Register Swagger generator

            services.AddSwaggerGen(c =>

            {

                c.SwaggerDoc("v1", new OpenApiInfo

                {

                    Title = "Employee CRUD API",

                    Version = "v1",

                    Description = "API to perform CRUD operations on Employee data"

                });

            });

        }

        // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

        {

            if (env.IsDevelopment())

            {

                app.UseDeveloperExceptionPage();

            }

            // Enable middleware to serve generated Swagger as a JSON endpoint.

            app.UseSwagger();

            // Enable middleware to serve Swagger UI (HTML, JS, CSS, etc.)

            app.UseSwaggerUI(c =>

            {

                c.SwaggerEndpoint("/swagger/v1/swagger.json", "Employee CRUD API v1");

                c.RoutePrefix = string.Empty; // Serve Swagger UI at app's root (http://localhost:<port>/)

            });

            app.UseRouting();

            app.UseAuthorization();

            app.UseEndpoints(endpoints =>

            {

                endpoints.MapControllers();

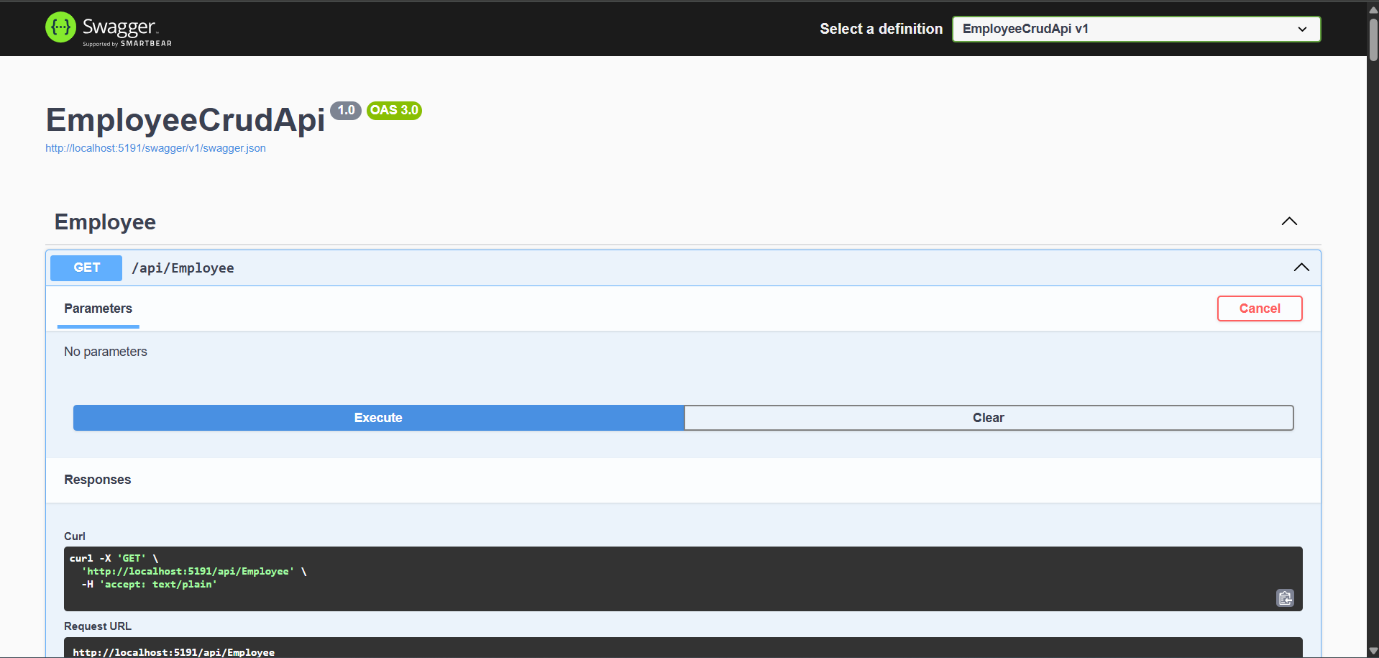
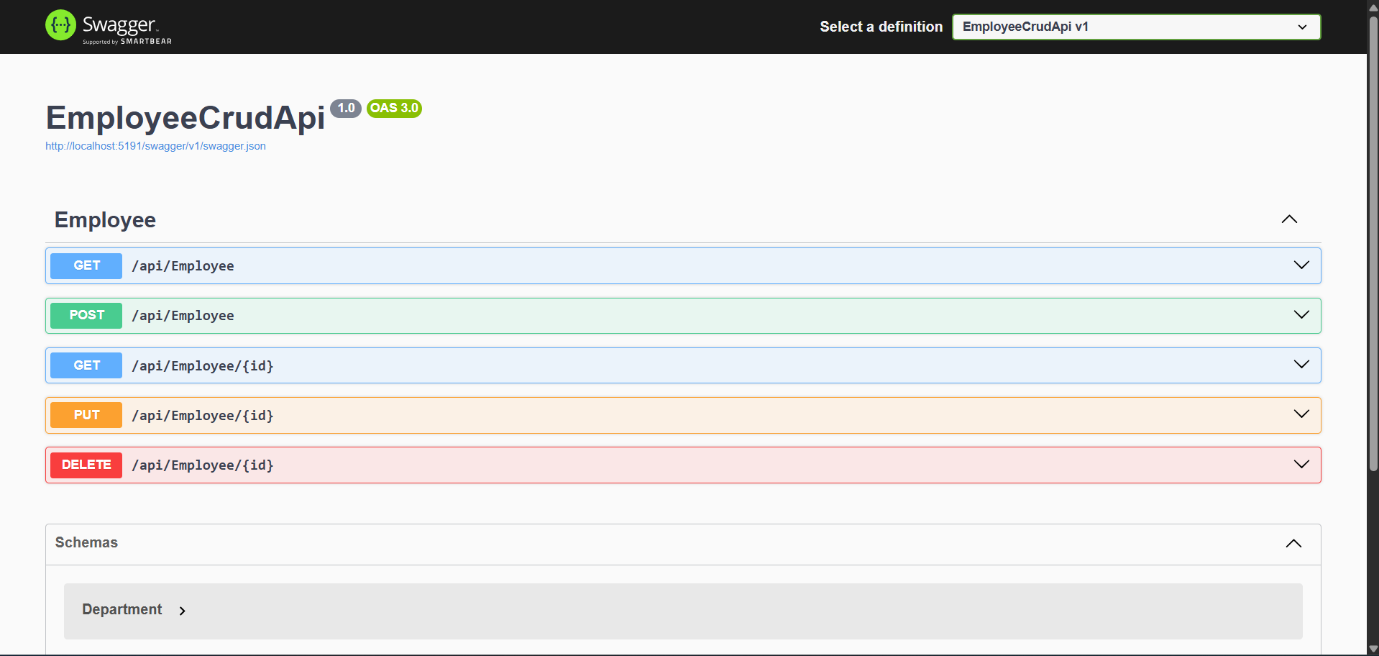
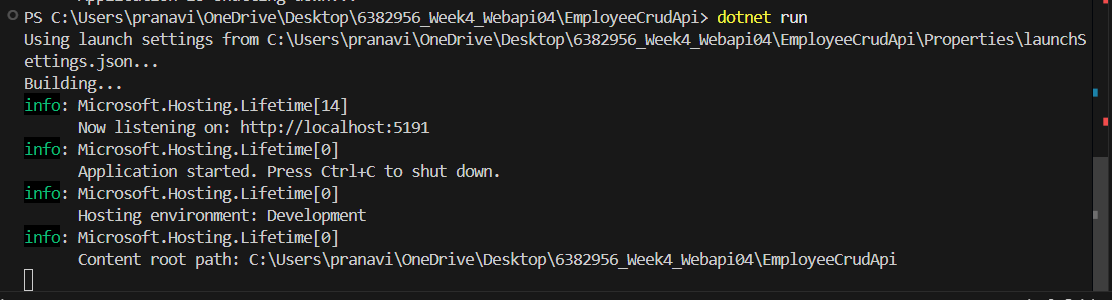
            });

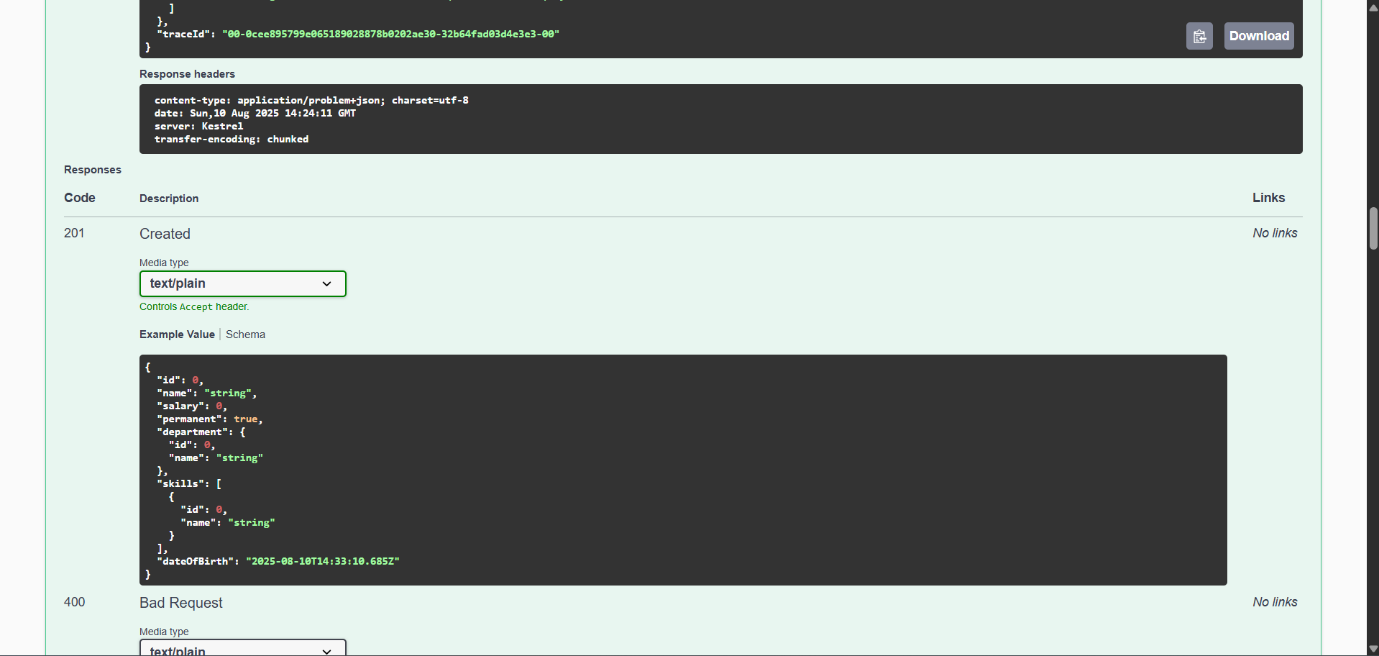
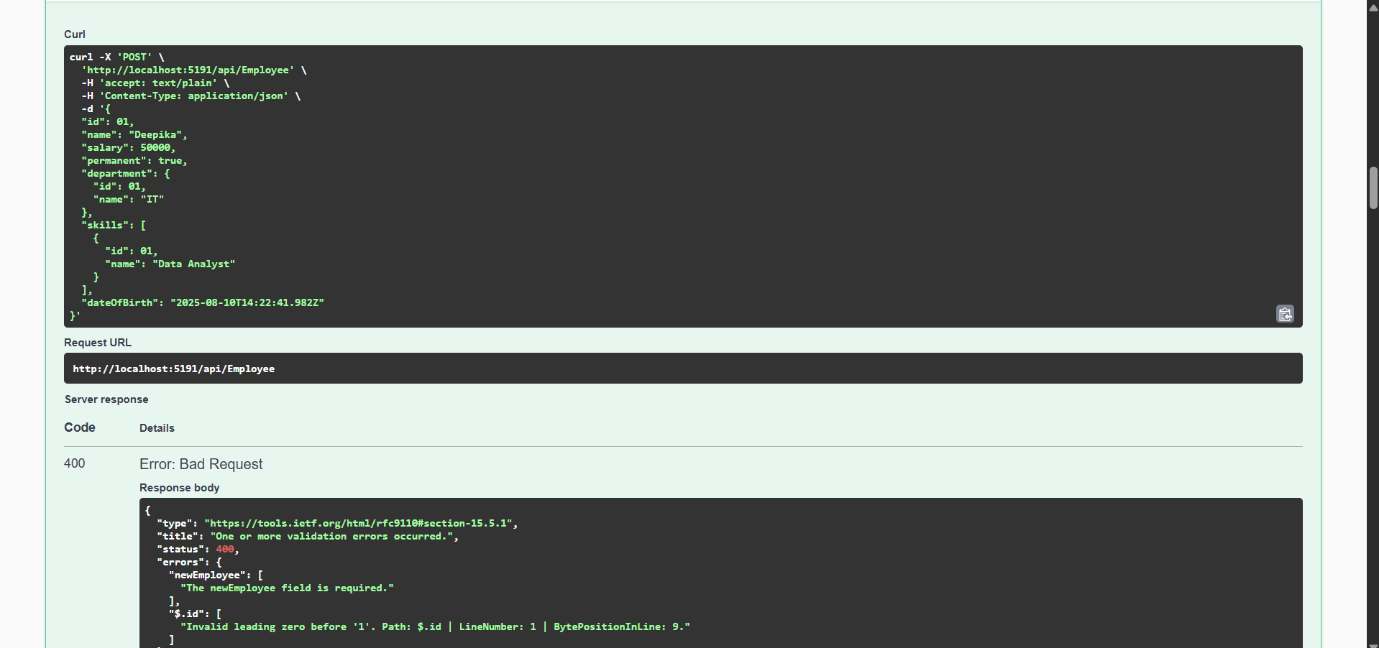
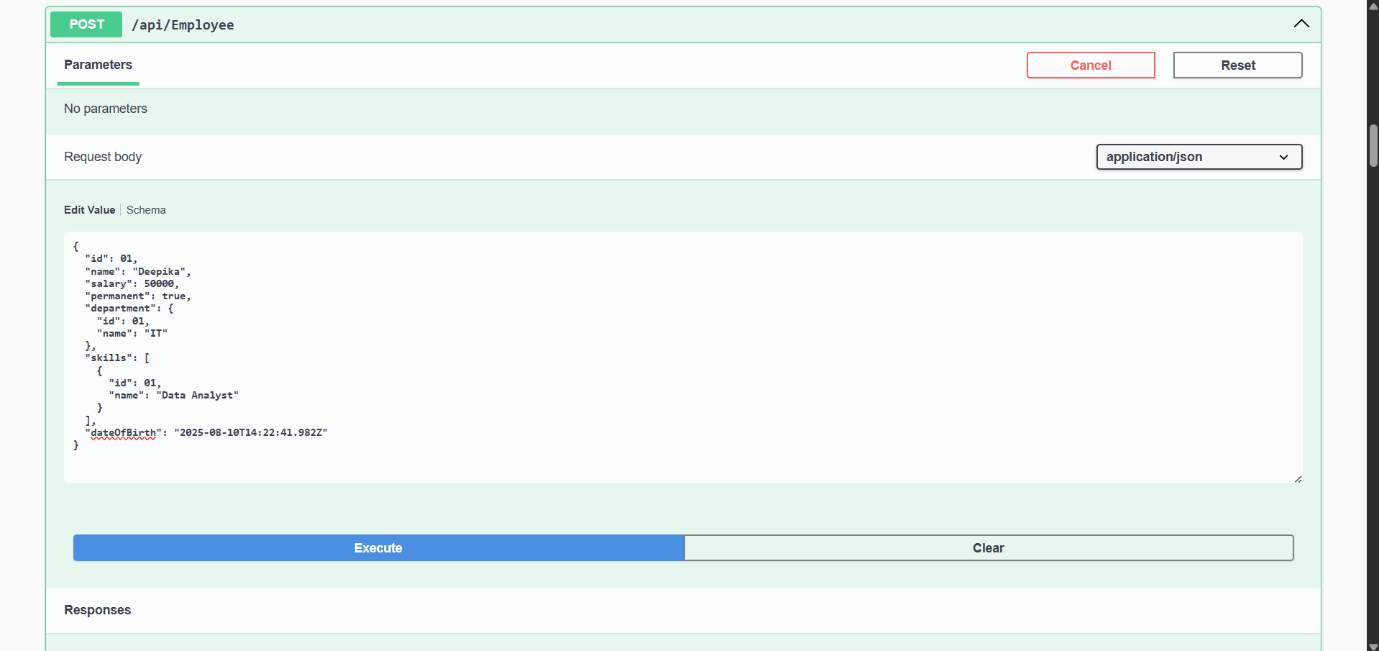
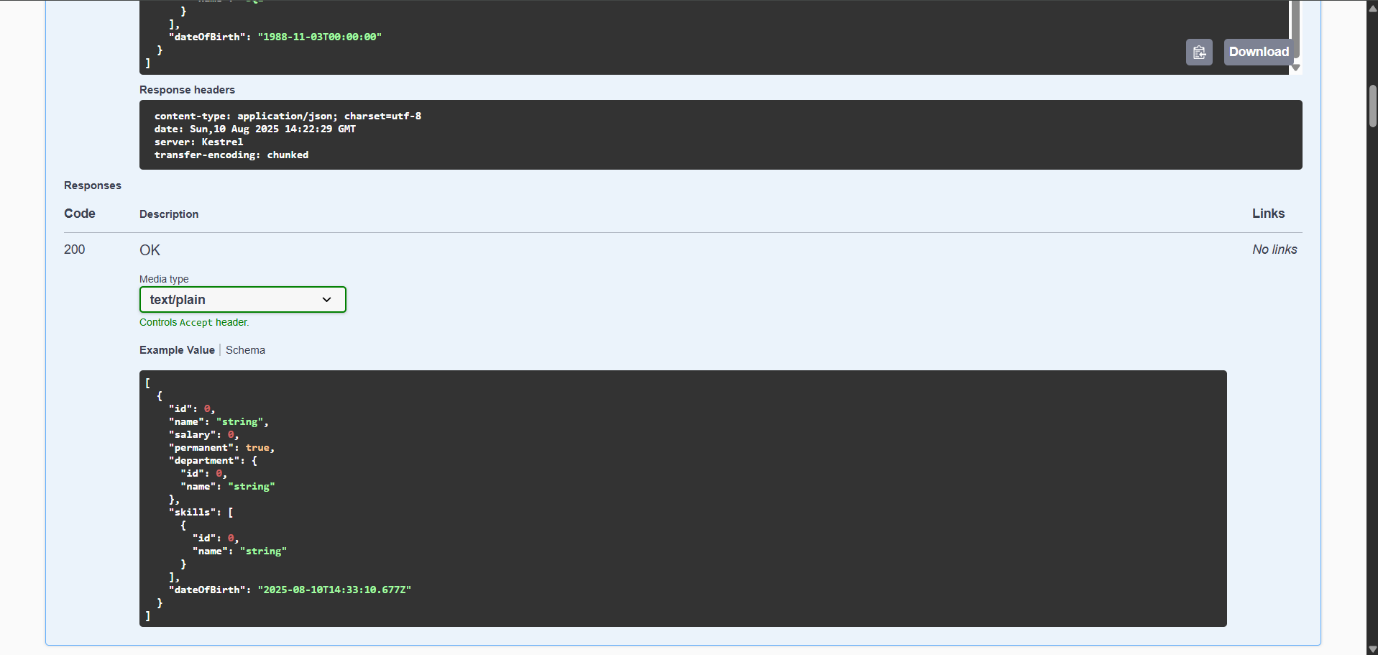
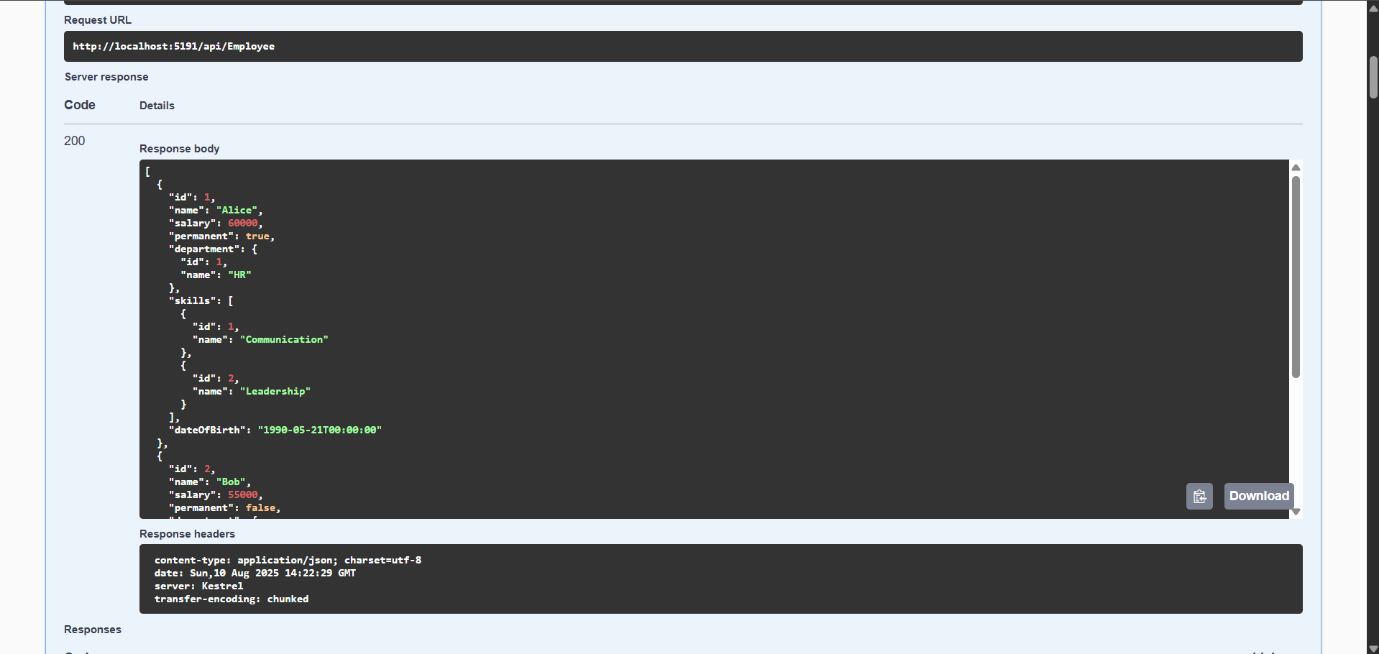
        }

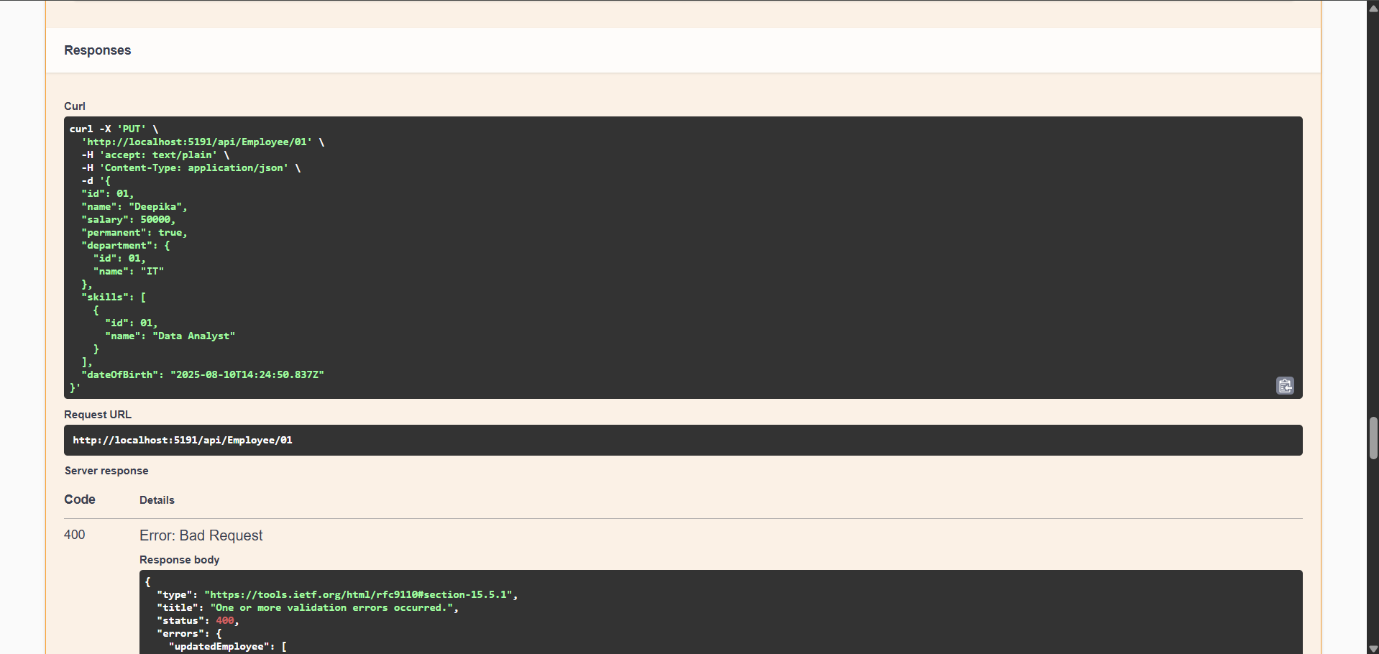
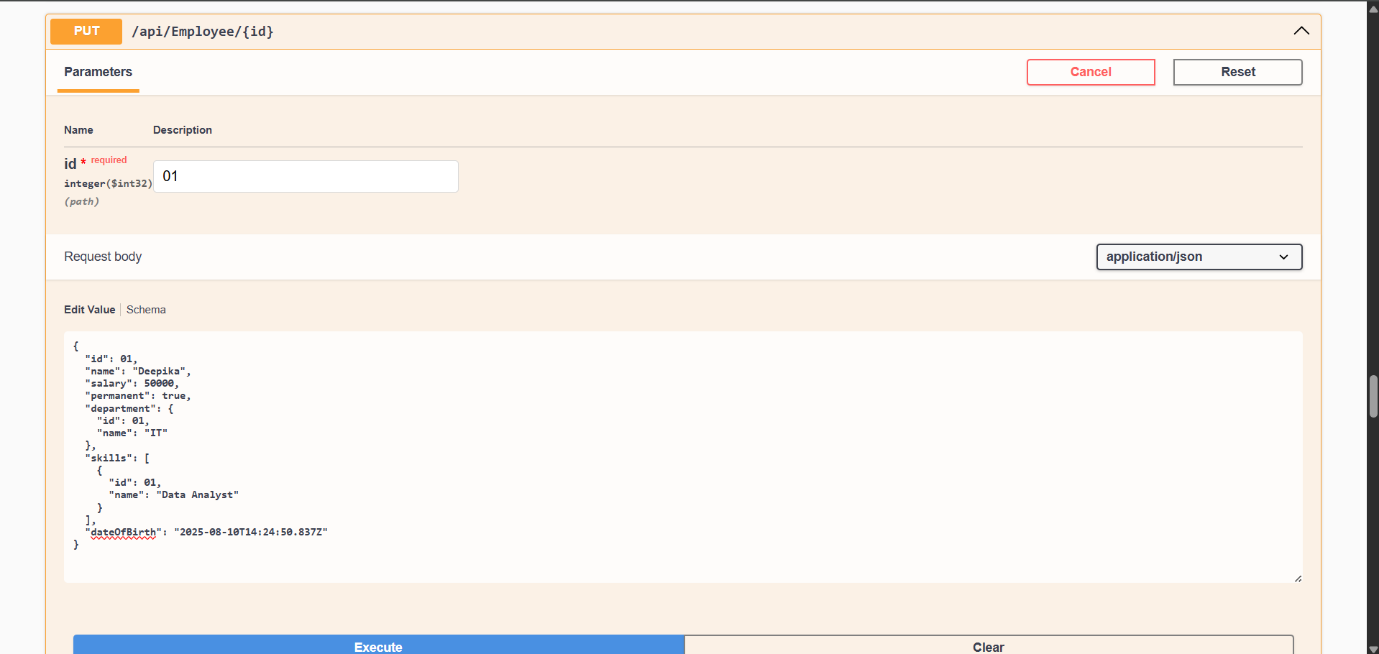
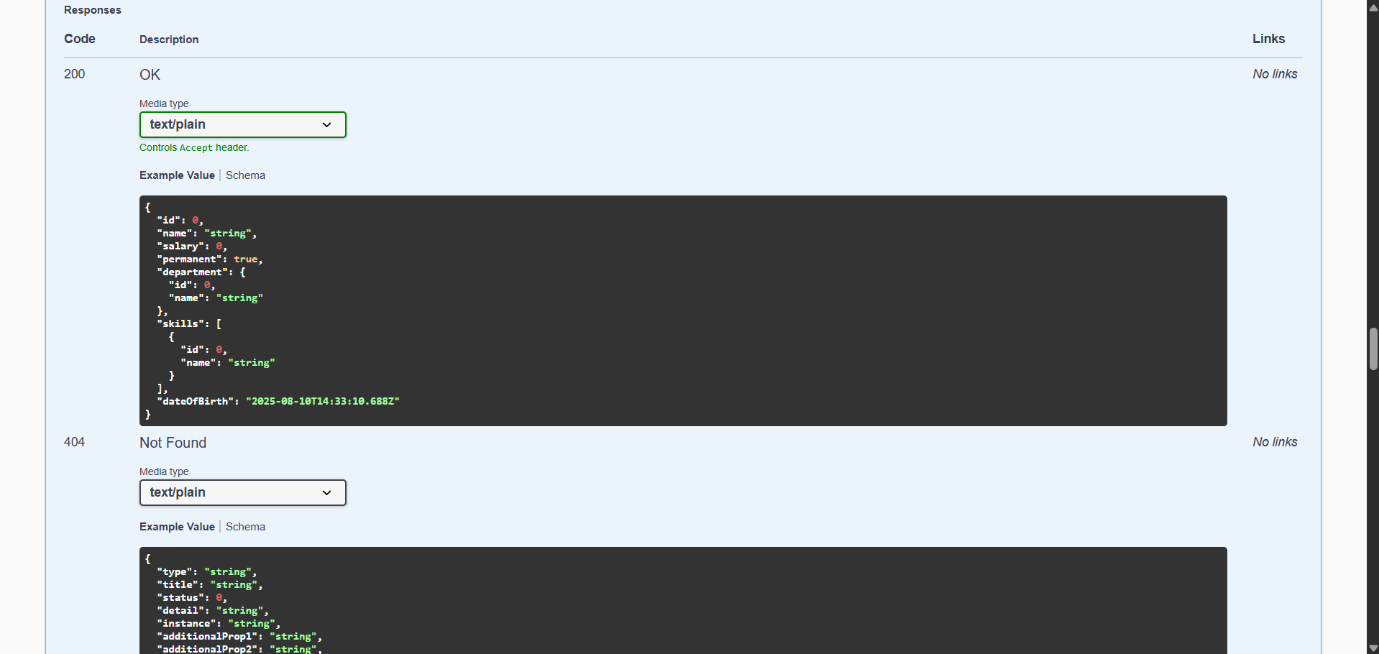
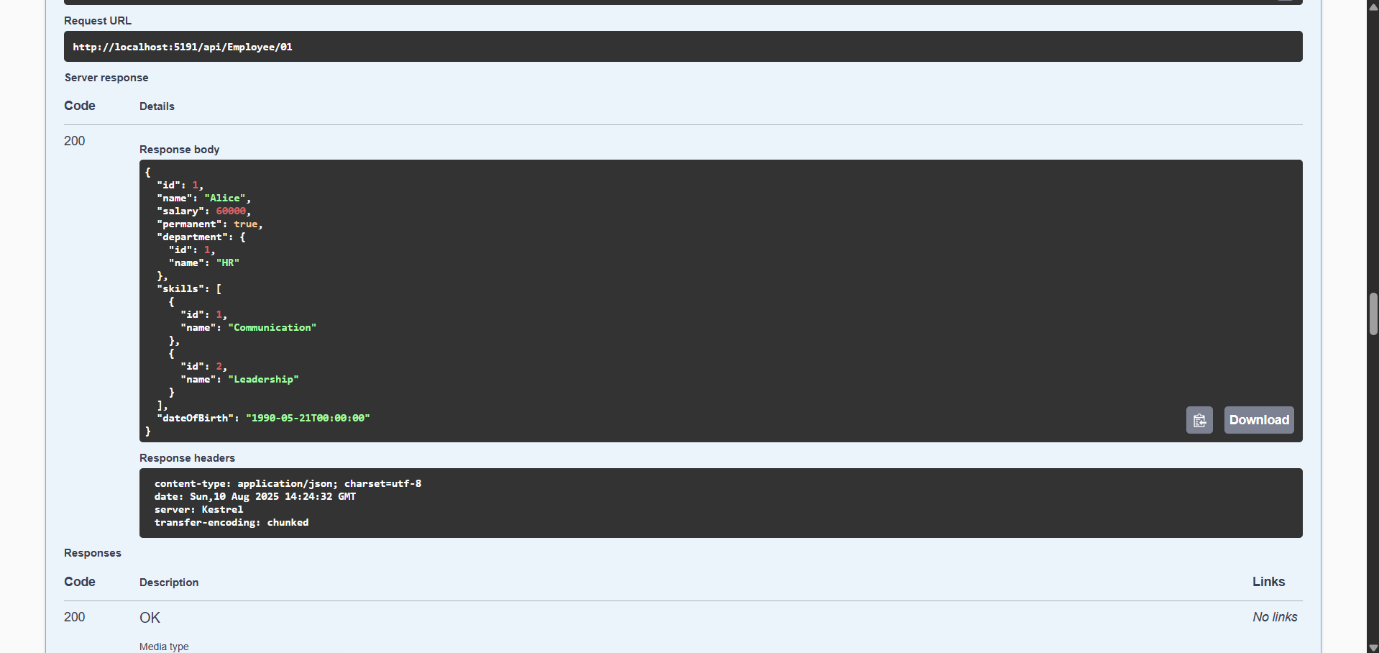
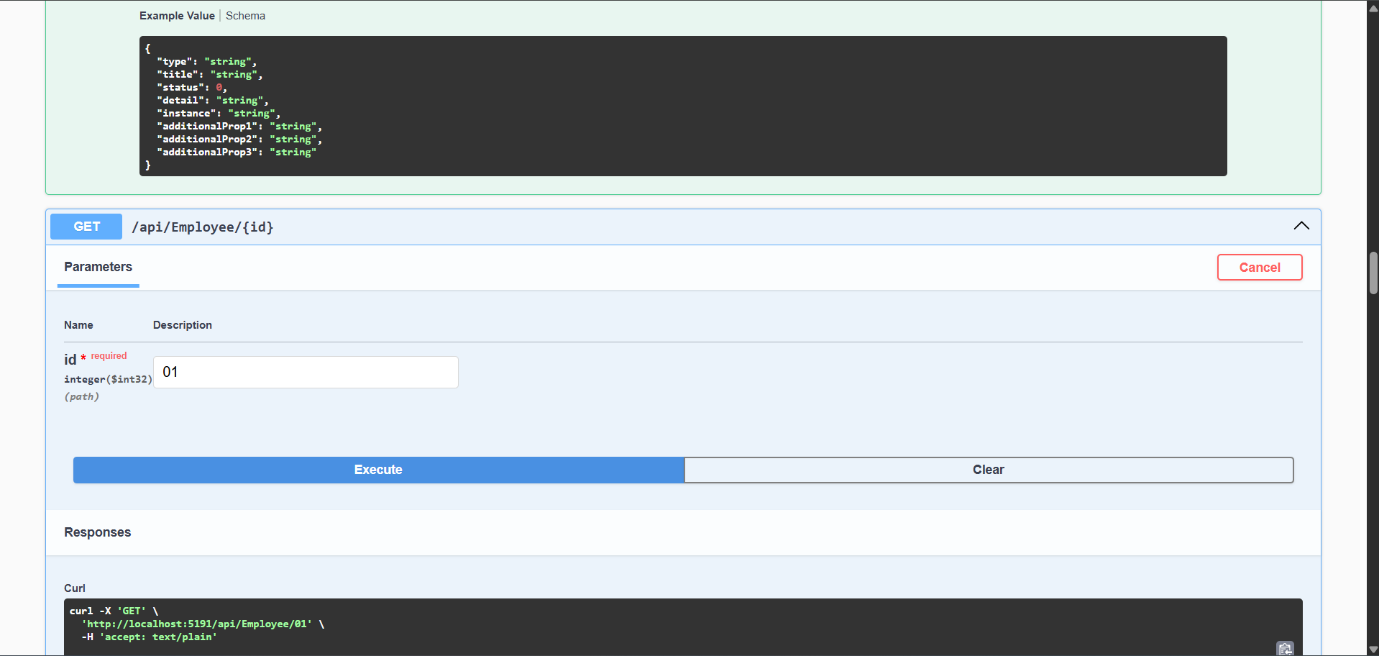
    }

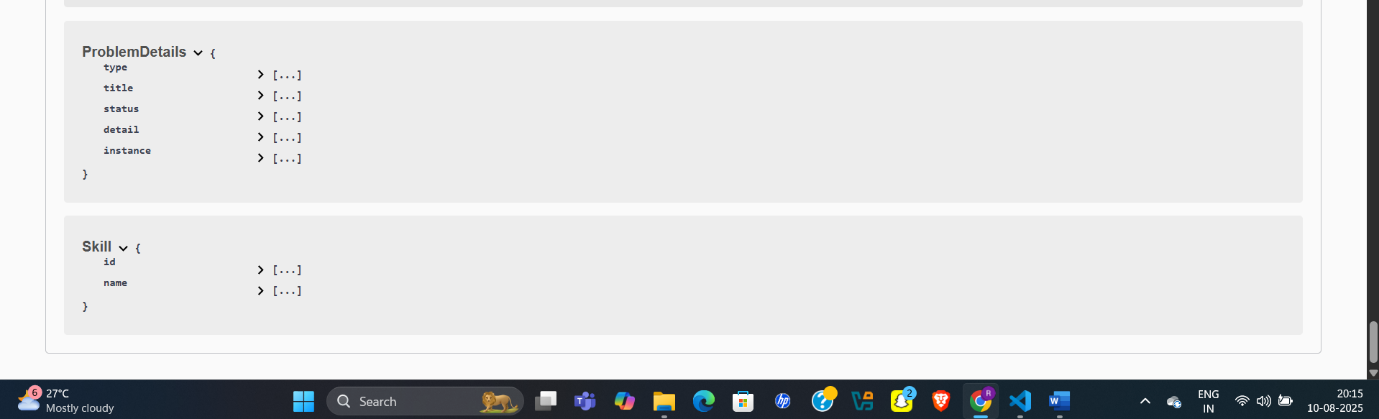
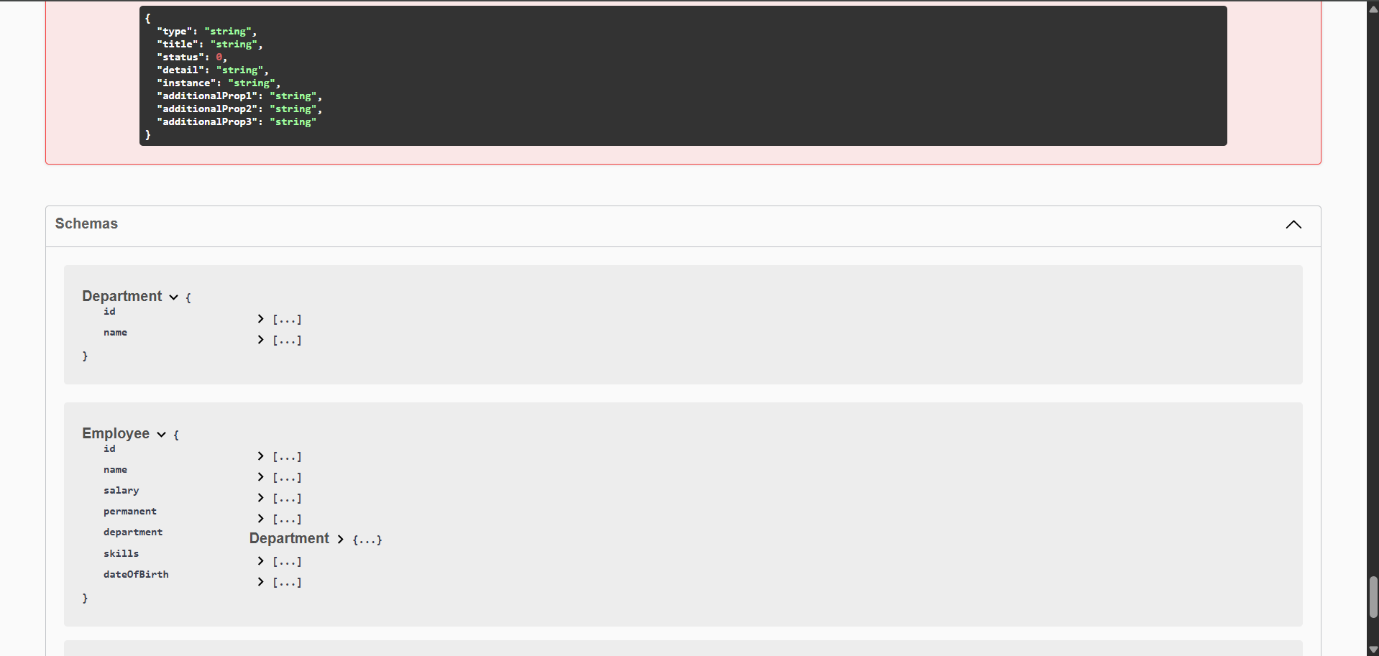
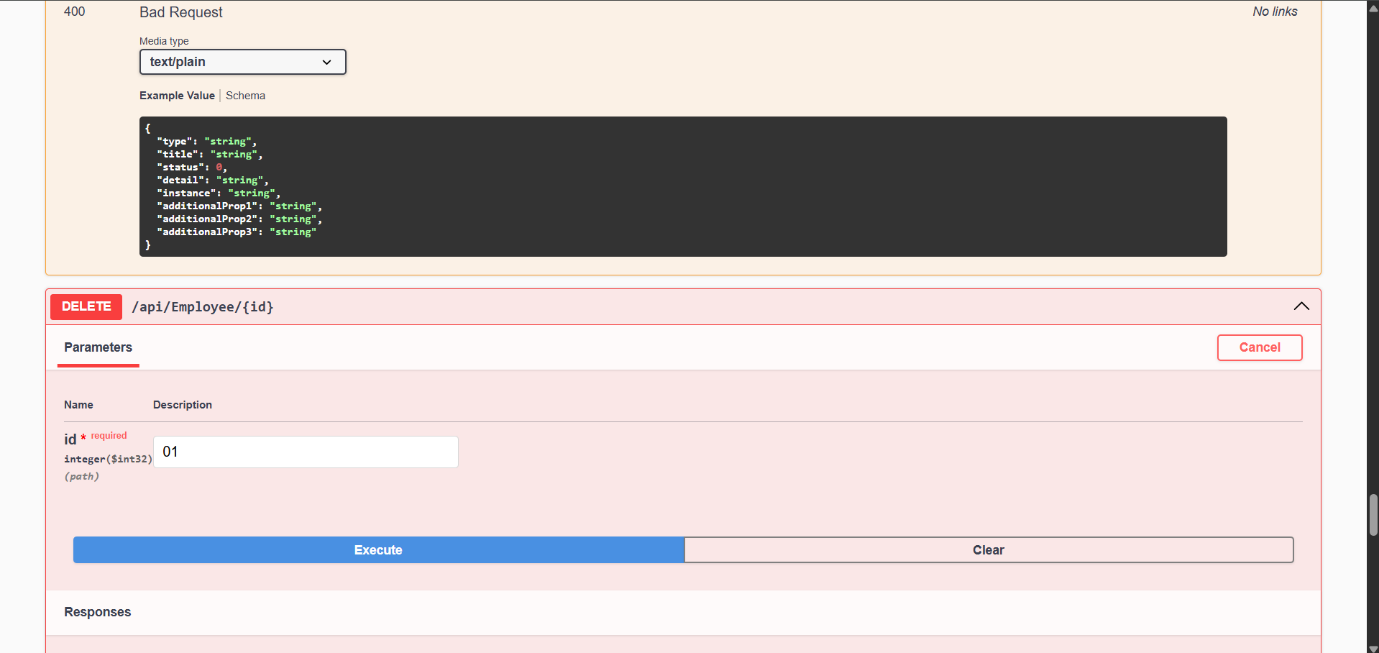
}

**Output:**

****

****

****

****