**WEEK 4: ASP.NET Core 8.0 Web API**

**1.WebAPI HandsOn**

**RESTFUL (Representational State Transfer) Web Service**

* It is a stateless, resource based service uses HTTP verbs.

**Web Service**

* Exposes RESTful services via HTTP and supports multiple data formats (JSON, XML).

**Microservices:**

* It is small, independent, focused services, loosely coupled and scalable.

**HTTPRequest:** Incoming data from client.

**HTTPResponse:** Outgoing data to client.

**ActionVerbs:**

* [HttpGet]: Read data
* [HttpPost]: Create
* [HttpPut]: Update
* [HttpDelete]: Delete

**HTTP Status Codes:**

Ok() - 200

BadRequest() – 400

Unauthorized() – 401

StatusCode(500) – Internal Server Error

**Configuration Files in Web API:**

* **.NET Core**: Startup.cs, appSettings.json, launchSettings.json
* **.NET 4.5**: WebApi.config, Route.config

**1.First Web API using .Net Core**

**EmployeeController.cs**

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

namespace FirstWebAPI.Controllers

{

[ApiController]

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

public class EmployeeController : ControllerBase

{

private static readonly List<string> Employees = new()

{

"Kaviin", "Kaushik", "Kavirajan"

};

[HttpGet]

public ActionResult<IEnumerable<string>> Get()

{

return Employees;

}

[HttpGet("{id}")]

public ActionResult<string> Get(int id)

{

if (id < 0 || id >= Employees.Count)

return NotFound("Employee not found");

return Employees[id];

}

[HttpPost]

public IActionResult Post([FromBody] string name)

{

Employees.Add(name);

return Ok("Employee added successfully");

}

[HttpPut("{id}")]

public IActionResult Put(int id, [FromBody] string name)

{

if (id < 0 || id >= Employees.Count)

return NotFound("Employee not found");

Employees[id] = name;

return Ok("Employee updated successfully");

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

if (id < 0 || id >= Employees.Count)

return NotFound("Employee not found");

Employees.RemoveAt(id);

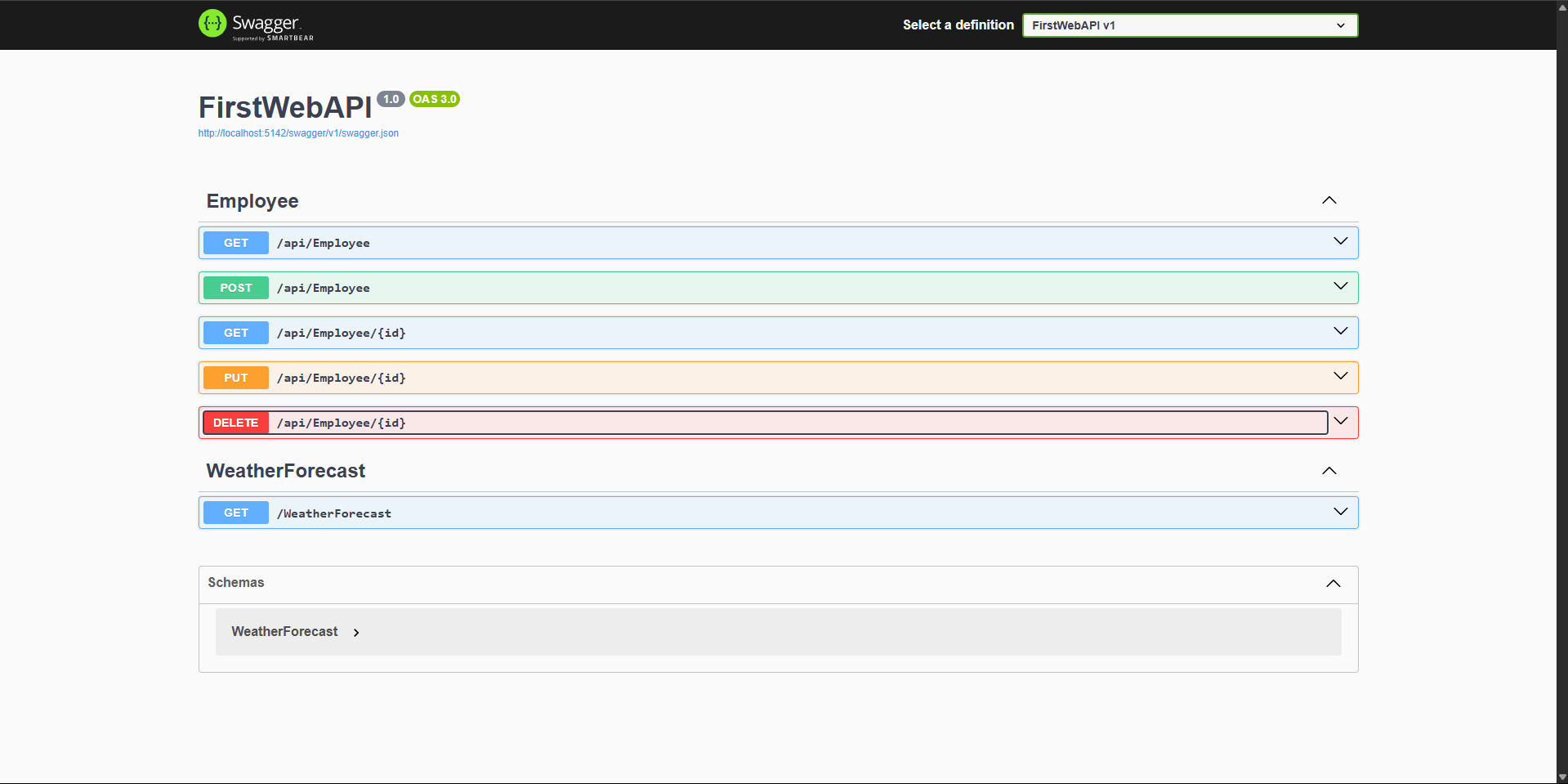
return Ok("Employee deleted successfully");

}

}

}

**OUTPUT:**



**2.WebApi HandsOn**

**1. Web Api using .Net core with Swagger**

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 OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "Employee API Demo",

Contact = new OpenApiContact { Name = "John Doe", Email = "john@example.com" }

});

});

var app = builder.Build();

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

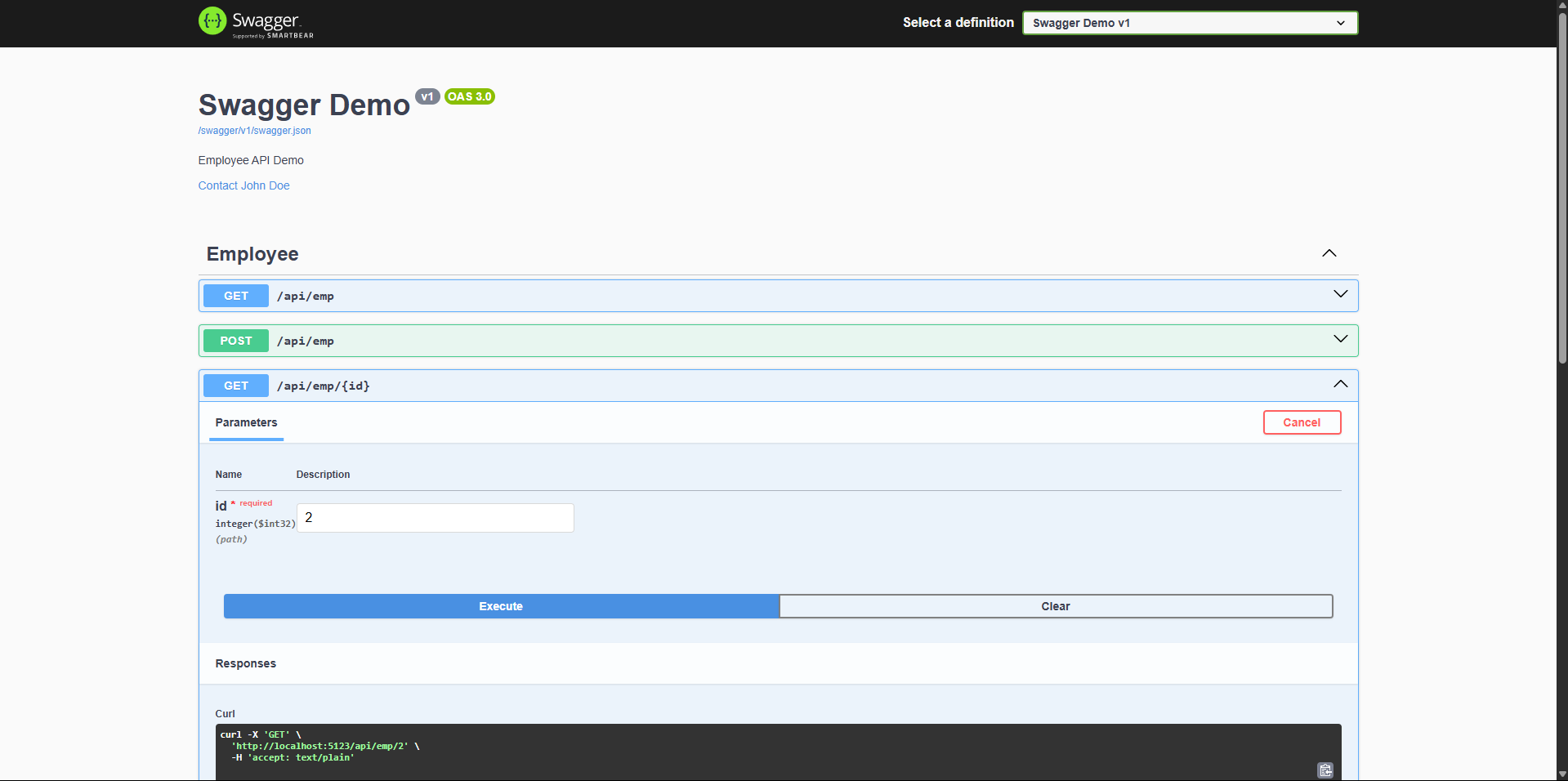
});

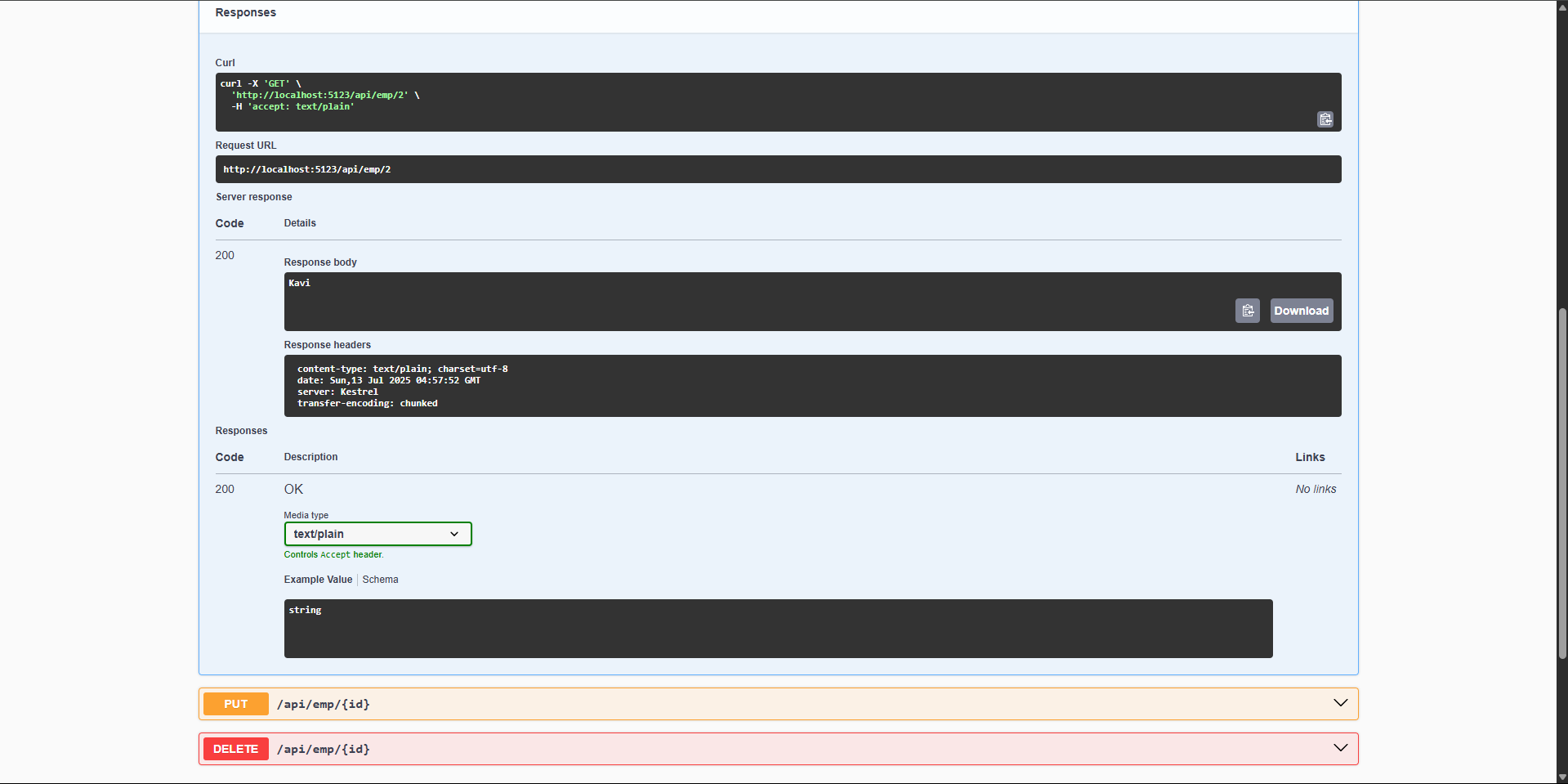
app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

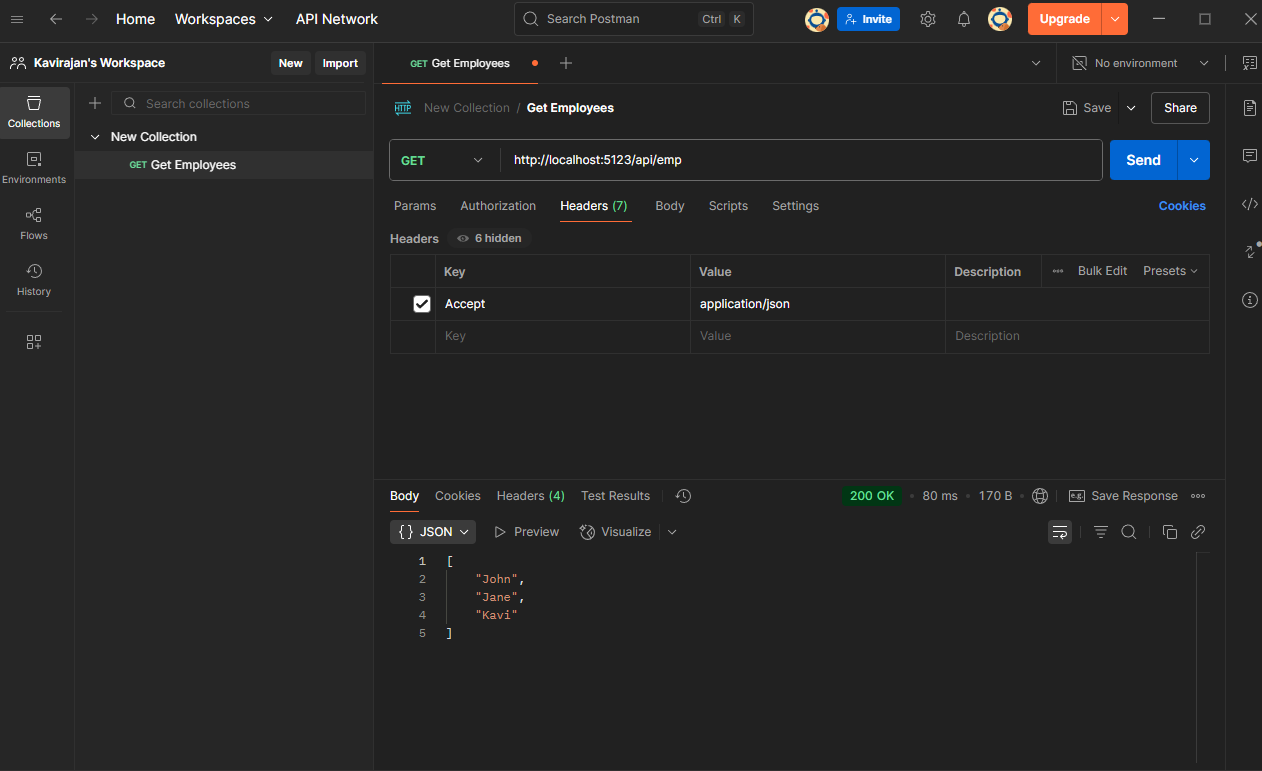
app.Run();

**Output:** 



**2.Using Postman Tool**

**Output:**

****

**3.Modify the Controller name in the Route attribute of the Employee controller to ‘Emp’ and check its access thru POSTMAN**

using Microsoft.AspNetCore.Mvc;

[ApiController]

[Route("api/emp")]

public class EmployeeController : ControllerBase

{

private static readonly List<string> Employees = new() { "Kaviin", "Kaushik", "Kavirajan" };

[HttpGet]

public ActionResult<IEnumerable<string>> Get() => Employees;

[HttpGet("{id}")]

public ActionResult<string> Get(int id) =>

(id >= 0 && id < Employees.Count) ? Employees[id] : NotFound();

[HttpPost]

public IActionResult Post([FromBody] string name)

{

Employees.Add(name);

return Ok("Added");

}

[HttpPut("{id}")]

public IActionResult Put(int id, [FromBody] string name)

{

if (id < 0 || id >= Employees.Count) return NotFound();

Employees[id] = name;

return Ok("Updated");

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

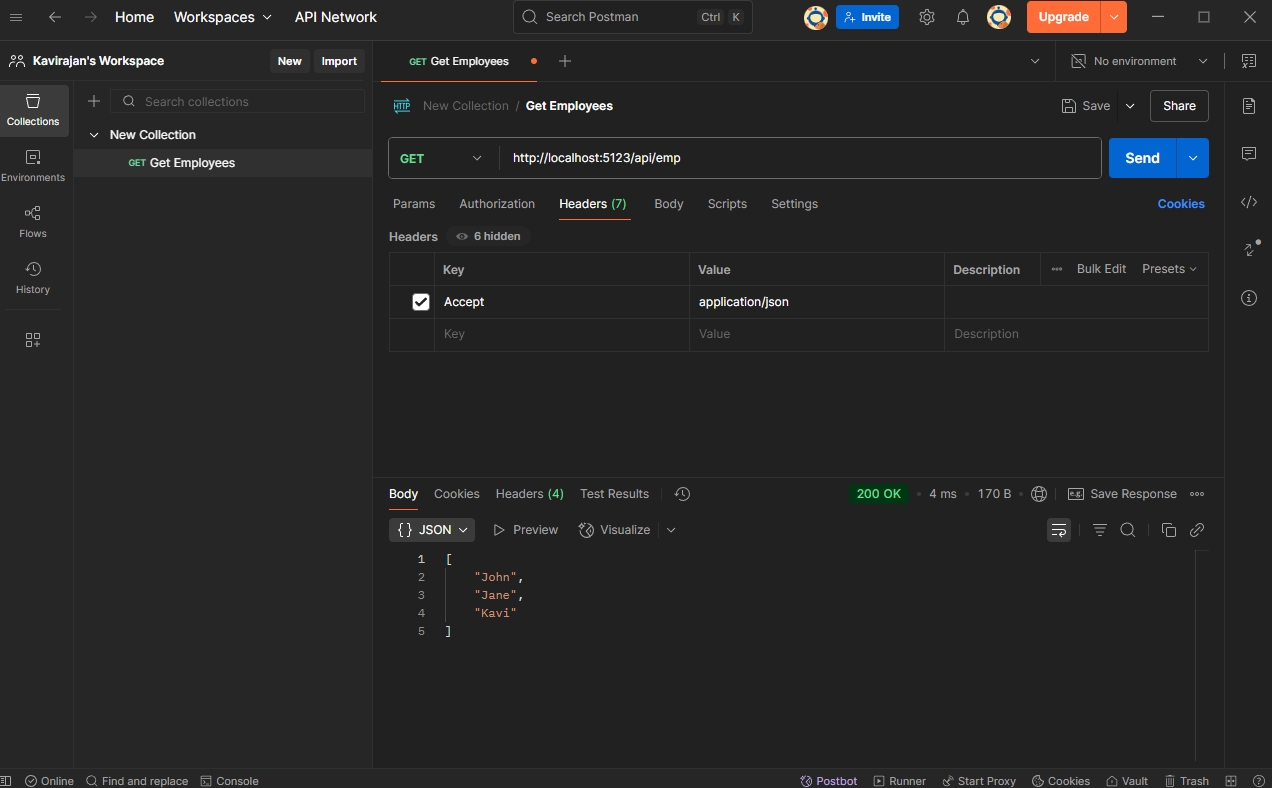
if (id < 0 || id >= Employees.Count) return NotFound();

Employees.RemoveAt(id);

return Ok("Deleted");

}

}

**Output: **

**3. WebAPI HandsOn**

**Employee.cs**

using System;

using System.Collections.Generic;

namespace FirstWebAPI.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; }

}

public class Department

{

public int Id { get; set; }

public string DeptName { get; set; }

}

public class Skill

{

public int Id { get; set; }

public string SkillName { get; set; }

}

}

**EmployeeController.cs**

using FirstWebAPI.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using YourNamespace.Filters;

namespace YourNamespace.Controllers

{

[ApiController]

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

[ServiceFilter(typeof(CustomAuthFilter))]

public class EmployeeController : ControllerBase

{

private readonly List<Employee> \_employees;

public EmployeeController()

{

\_employees = GetStandardEmployeeList();

}

private List<Employee> GetStandardEmployeeList() => new()

{

new Employee

{

Id = 1,

Name = "Kavi Rajan",

Salary = 40000,

Permanent = true,

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

Skills = new List<Skill> { new Skill { Id = 1, SkillName = "C#" } },

DateOfBirth = new DateTime(2000, 1, 1)

}

};

[HttpGet("GetStandard")]

[ProducesResponseType(200)]

[ProducesResponseType(500)]

public ActionResult<List<Employee>> GetStandard()

{

return Ok(\_employees);

}

[HttpPost]

public IActionResult AddEmployee([FromBody] Employee emp)

{

\_employees.Add(emp);

return Ok(emp);

}

}

}

**CustomAuthFilter.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System.Linq;

namespace YourNamespace.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.StartsWith("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;

using System;

using System.IO;

namespace YourNamespace.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

var exceptionMessage = $"Exception: {context.Exception.Message} | Date: {DateTime.Now}";

File.AppendAllText("logs.txt", exceptionMessage + Environment.NewLine);

context.Result = new ObjectResult("An error occurred. Please try again later.")

{

StatusCode = 500

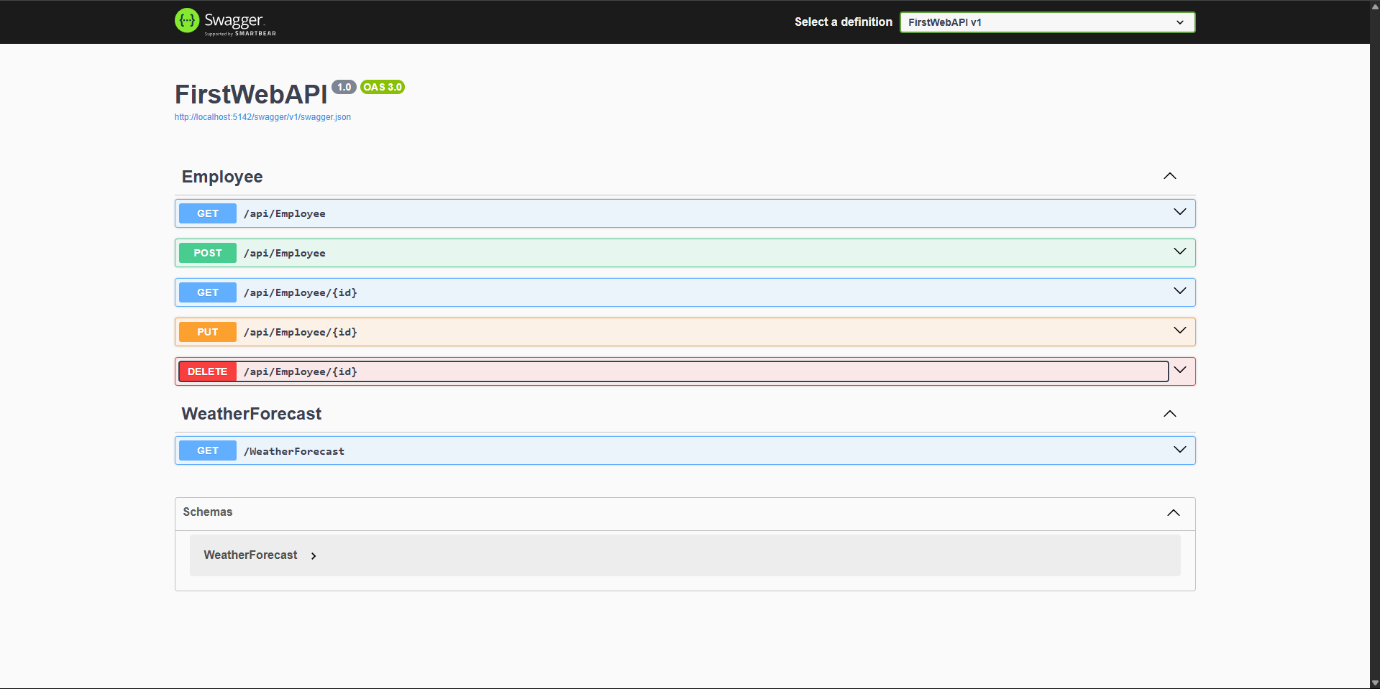
};

}

}

}

**Output:**



**4.WebApi HandsOn**

**Testing with Swagger**

1. Open https://localhost:/swagger
2. Try POST, PUT, DELETE operations directly from the UI.
3. Swagger will auto-generate input form for your Product model.

**Testing with Postman**

* Create:
  + Type: POST
  + URL: https://localhost:/api/products
  + Body → raw → JSON:

**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.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

**EmployeeController.cs**

using Microsoft.AspNetCore.Mvc;

using EmployeeCrudApi.Models;

namespace EmployeeCrudApi.Controllers

{

[ApiController]

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

public class EmployeeController : ControllerBase

{

private static readonly List<Employee> Employees = new()

{

new Employee

{

Id = 1,

Name = "Kavi",

Salary = 60000,

Permanent = true,

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

DateOfBirth = new DateTime(1999, 01, 15)

},

new Employee

{

Id = 2,

Name = "Rajan",

Salary = 45000,

Permanent = false,

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

DateOfBirth = new DateTime(2000, 04, 10)

}

};

[HttpPut("{id}")]

public ActionResult<Employee> UpdateEmployee(int id, [FromBody] Employee updated)

{

if (id <= 0)

return BadRequest("Invalid employee id");

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

if (emp == null)

return BadRequest("Invalid employee id");

emp.Name = updated.Name;

emp.Salary = updated.Salary;

emp.Permanent = updated.Permanent;

emp.Department = updated.Department;

emp.DateOfBirth = updated.DateOfBirth;

return Ok(emp);

}

[HttpDelete("{id}")]

public IActionResult DeleteEmployee(int id)

{

if (id <= 0)

return BadRequest("Invalid employee id");

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

if (emp == null)

return NotFound("Employee not found");

Employees.Remove(emp);

return Ok($"Employee with ID {id} deleted successfully");

}

}

}

**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 DateTime DateOfBirth { get; set; }

}

}

**Department.cs**

namespace EmployeeCrudApi.Models

{

public class Department

{

public int Id { get; set; }

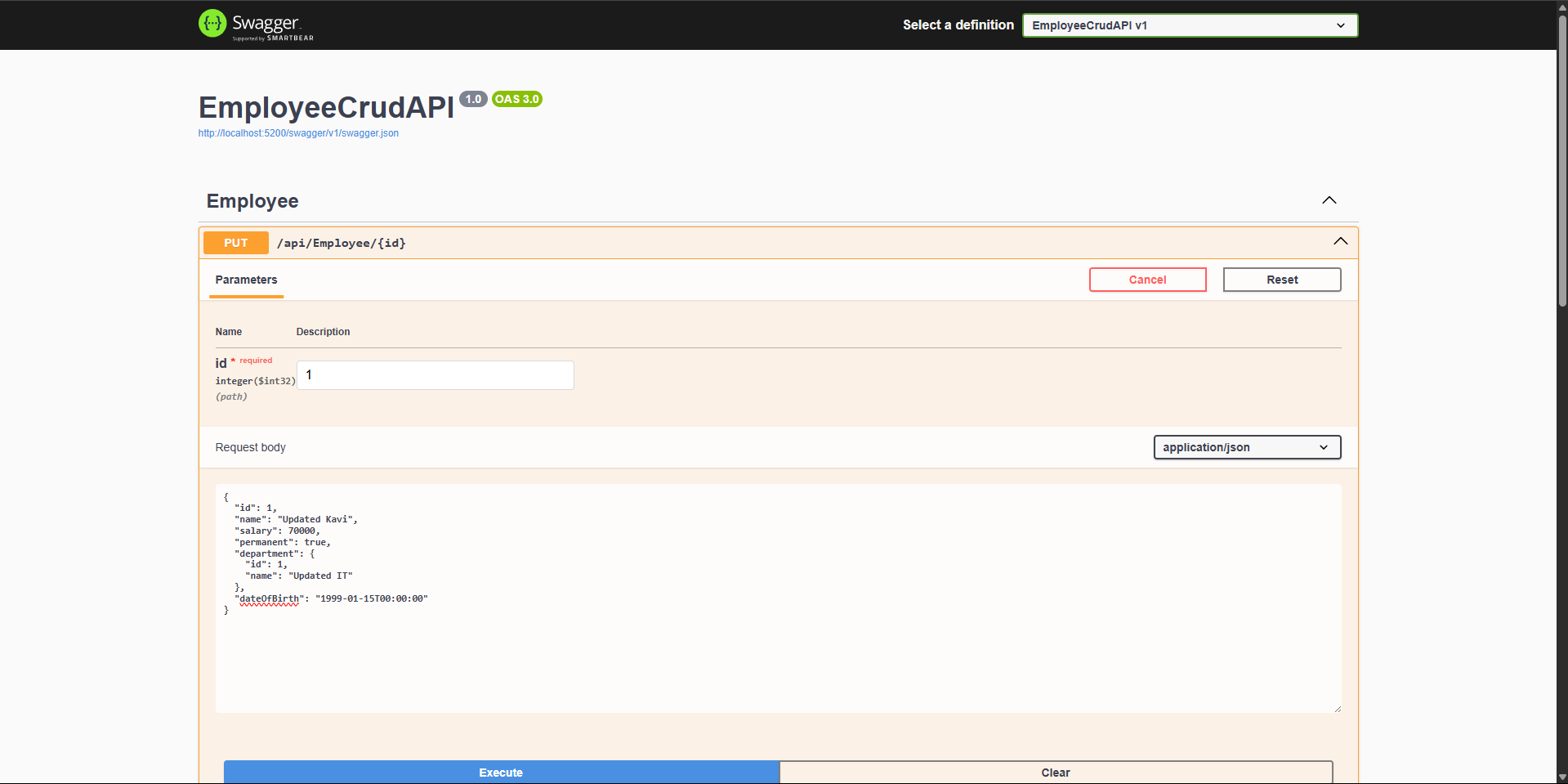
public string Name { get; set; }

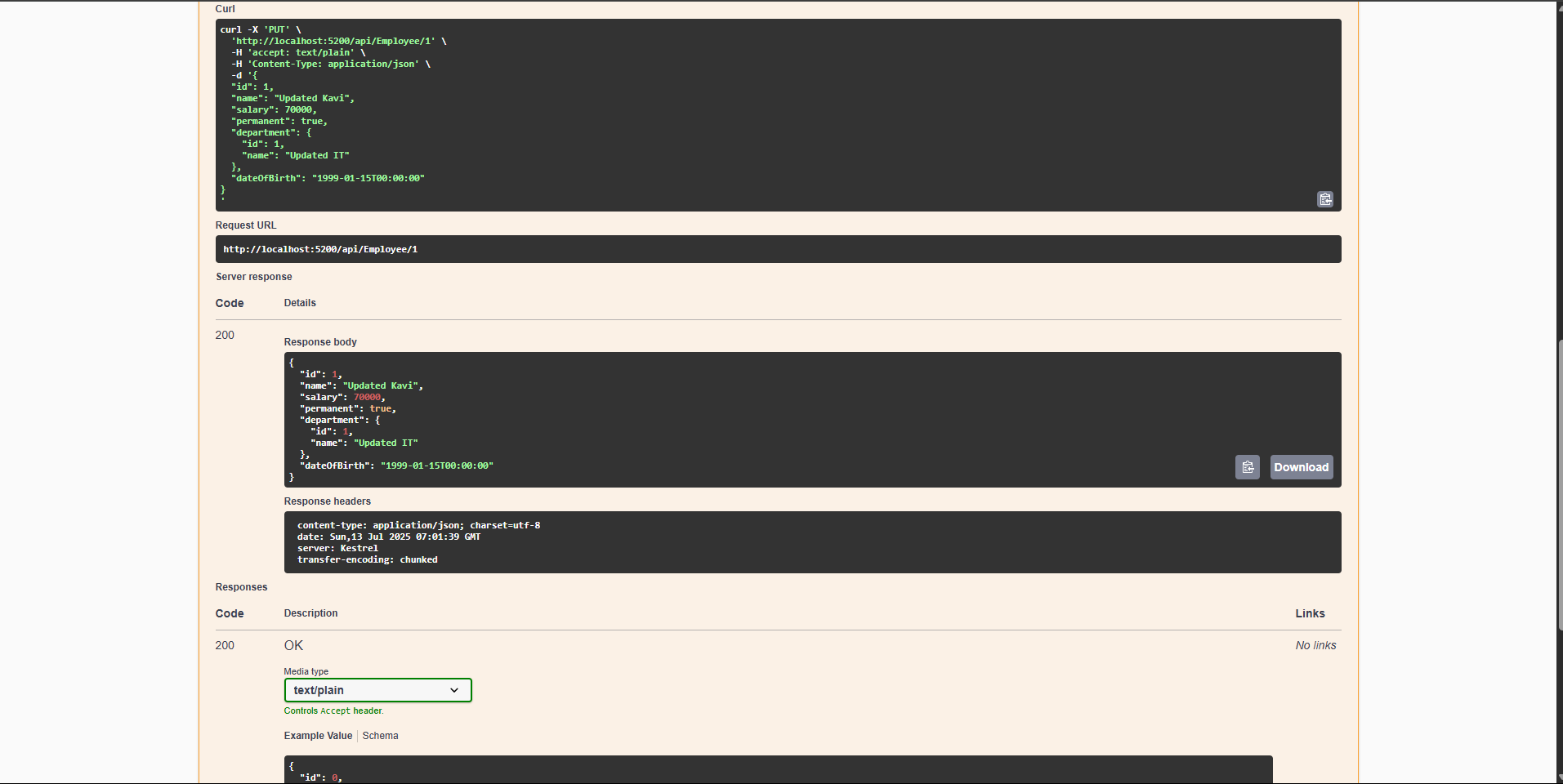
}

}

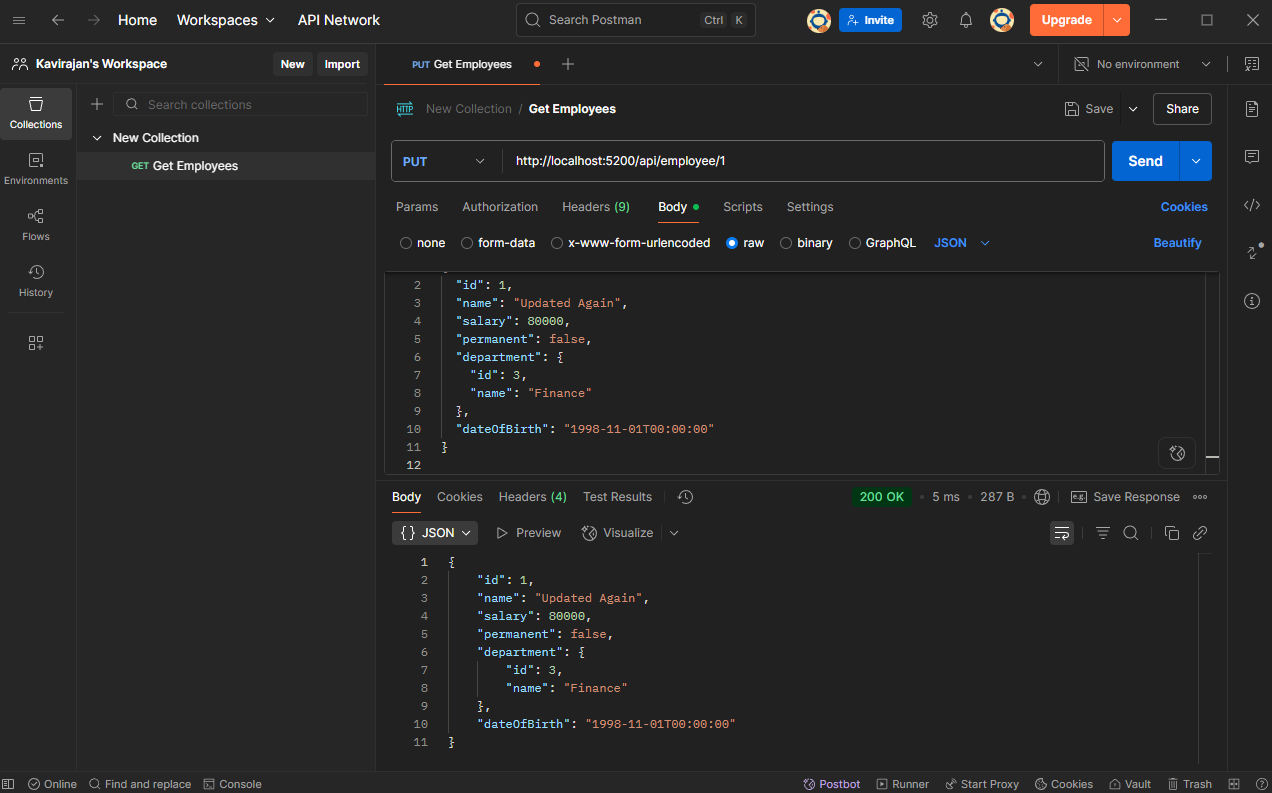
**WebApi Crud Operation: PUT Method**

**Using Swagger:**



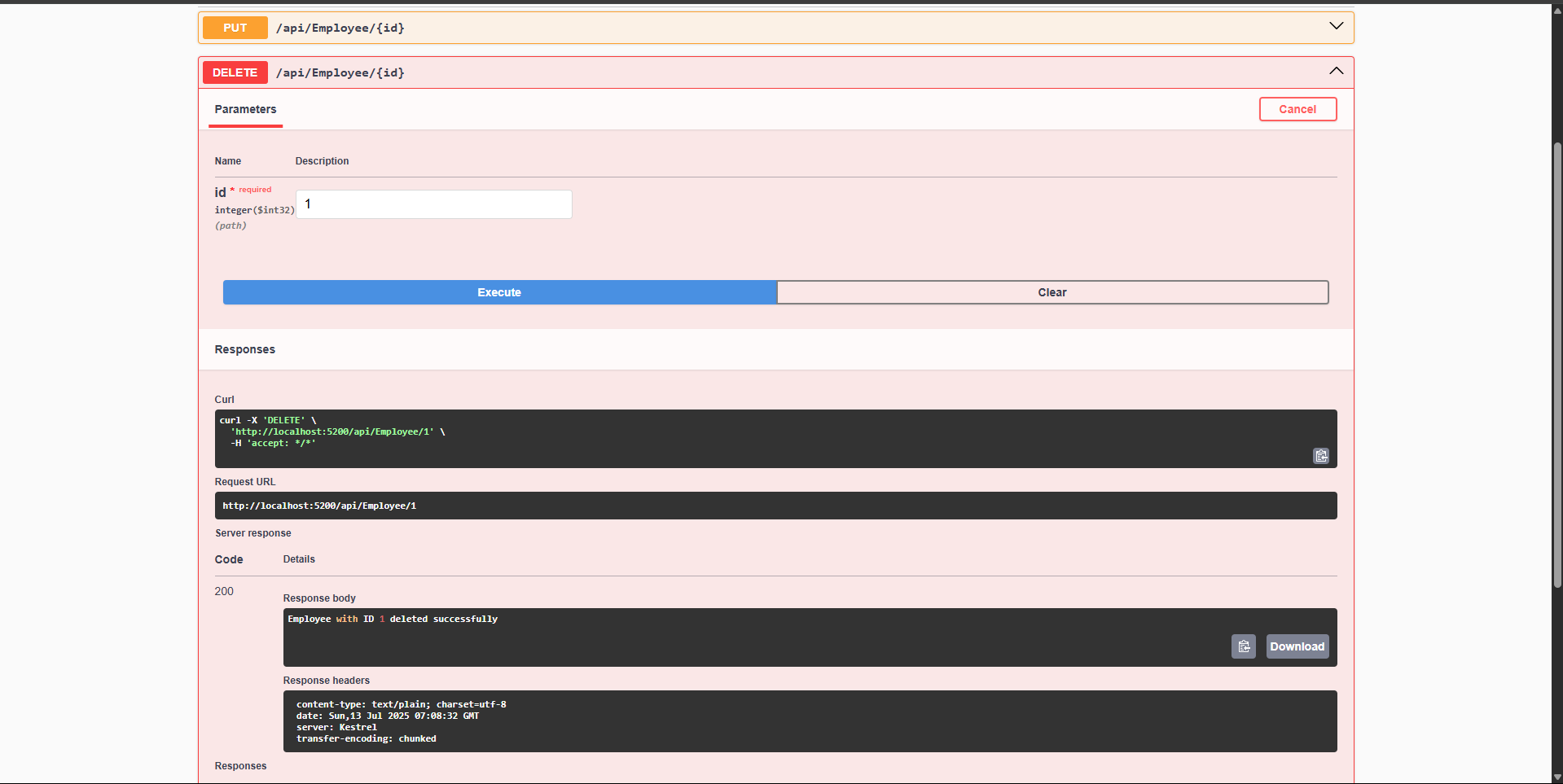


**Using Postman:**

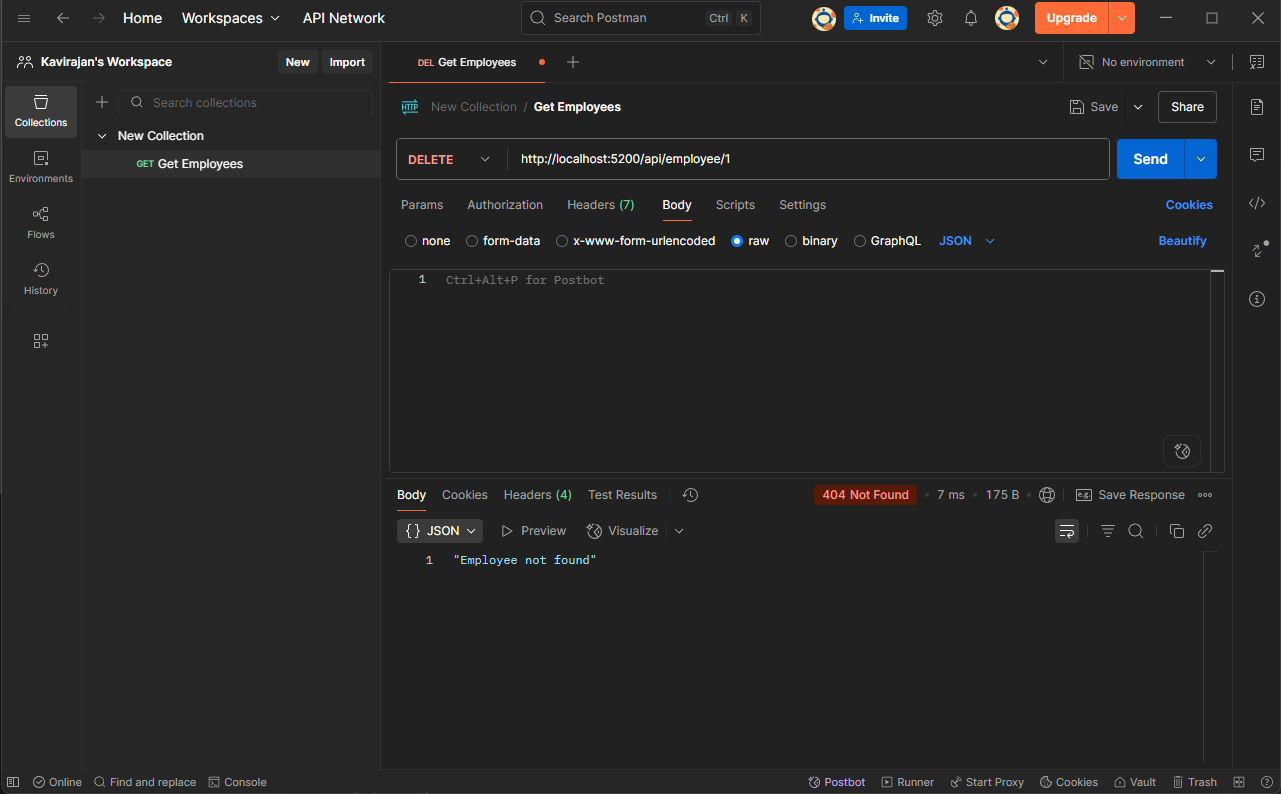


**Crud Operation: Delete Method**

**Using Swagger:**

****

**Using Postman:**

****

**5.JSON Web Token(JWT)**

**CORS in WebAPI:**

CORS (Cross-Origin Resource Sharing) allows your Web API to be accessed from a different domain. By default, browsers block cross-origin requests for security**.**

**Web API Security with JWT Authentication**

* JWT (JSON Web Token): Compact token with claims.
* Bearer Token: Sent in Authorization header (Bearer eyJhbGciOi...).
* Authorize: Restricts access to authenticated users.
* AllowAnonymous: Used for login/token generation endpoint.

**Program.cs**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

using System.Security.Claims;

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddCors(options =>

{

options.AddPolicy("AllowAll", policy =>

{

policy.AllowAnyOrigin()

.AllowAnyMethod()

.AllowAnyHeader();

});

});

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

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

c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

Description = "JWT Authorization header using the Bearer scheme (Example: 'Bearer token')",

Name = "Authorization",

In = ParameterLocation.Header,

Type = SecuritySchemeType.ApiKey,

Scheme = "Bearer"

});

c.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

new string[] {}

}

});

});

var key = Encoding.UTF8.GetBytes("mysuperdupersecret");

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddJwtBearer(options =>

{

options.TokenValidationParameters = new()

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = "mySystem",

ValidAudience = "myUsers",

IssuerSigningKey = new SymmetricSecurityKey(key)

};

});

builder.Services.AddAuthorization();

var app = builder.Build();

app.UseCors("AllowAll");

app.UseSwagger();

app.UseSwaggerUI();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

**Models/Employee.cs**

namespace JwtAuthApiDemo.Models

{

public class Employee

{

public int Id { get; set; }

public string Name { get; set; }

}

}

**Controllers/AuthController.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace JwtAuthApiDemo.Controllers

{

[ApiController]

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

public class AuthController : ControllerBase

{

[HttpGet("token")]

[AllowAnonymous]

public IActionResult GetToken()

{

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecret"));

var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);

var claims = new[]

{

new Claim(ClaimTypes.Name, "User1"),

new Claim(ClaimTypes.Role, "Admin"),

new Claim("UserId", "1")

};

var token = new JwtSecurityToken(

issuer: "mySystem",

audience: "myUsers",

claims: claims,

expires: DateTime.Now.AddMinutes(2),

signingCredentials: credentials);

return Ok(new { token = new JwtSecurityTokenHandler().WriteToken(token) });

}

}

}

**Controllers/EmployeeControler.cs**

using JwtAuthApiDemo.Models;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace JwtAuthApiDemo.Controllers

{

[ApiController]

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

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

public class EmployeeController : ControllerBase

{

private static List<Employee> Employees = new()

{

new Employee { Id = 1, Name = "Kavi" },

new Employee { Id = 2, Name = "Rajan" }

};

[HttpGet]

public ActionResult<List<Employee>> GetAll()

{

return Employees;

}

[HttpPost]

public IActionResult Create([FromBody] Employee emp)

{

Employees.Add(emp);

return Ok("Created");

}

[HttpPut("{id}")]

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

{

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

if (existing == null) return BadRequest("Invalid employee id");

existing.Name = emp.Name;

return Ok(existing);

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

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

if (existing == null) return BadRequest("Invalid employee id");

Employees.Remove(existing);

return Ok("Deleted");

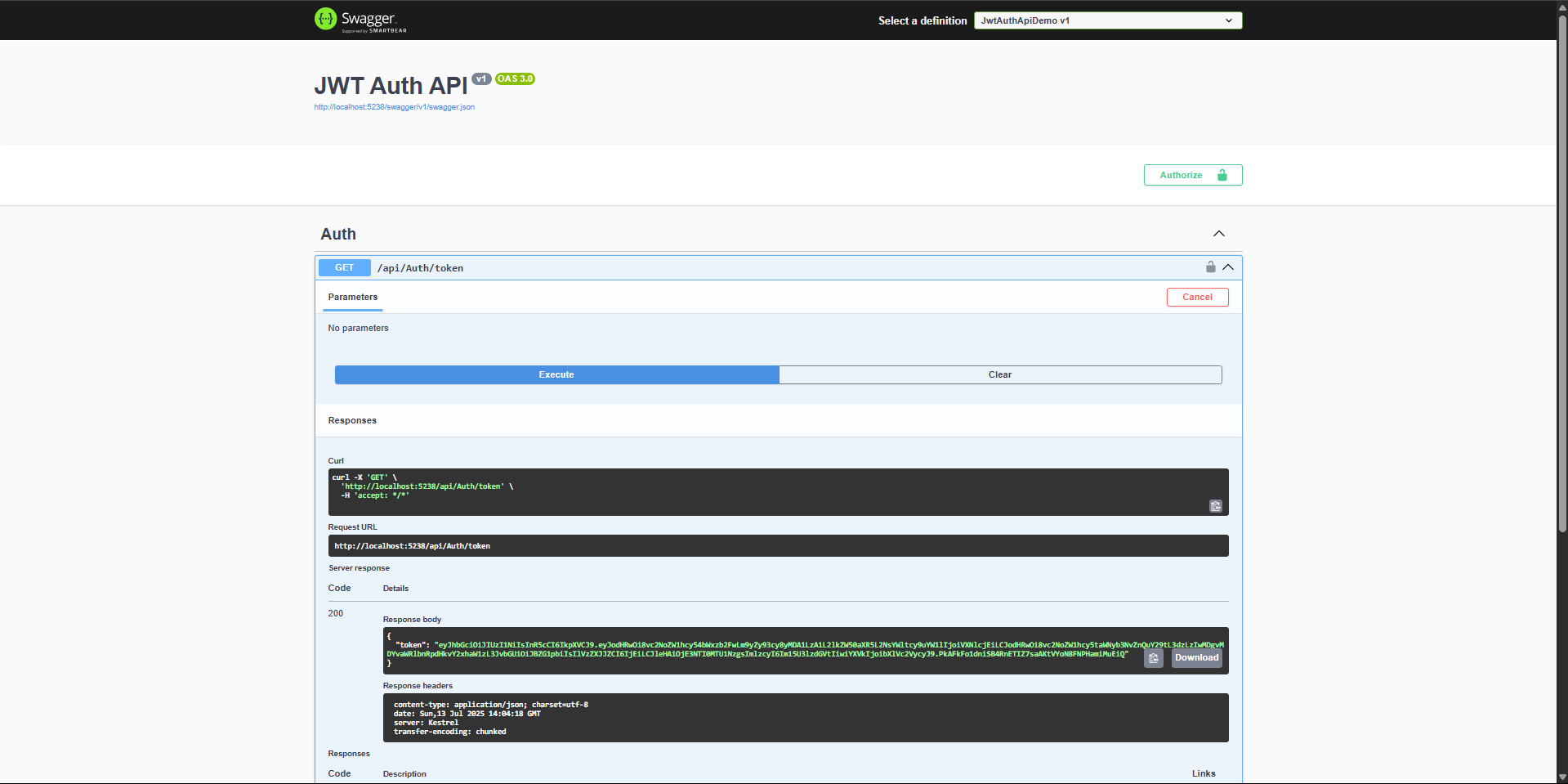
}

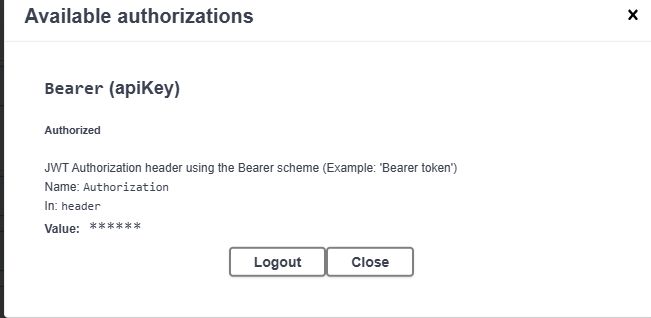
}

}

**Output:**

**Using Swagger:**





**Using Postman:**

