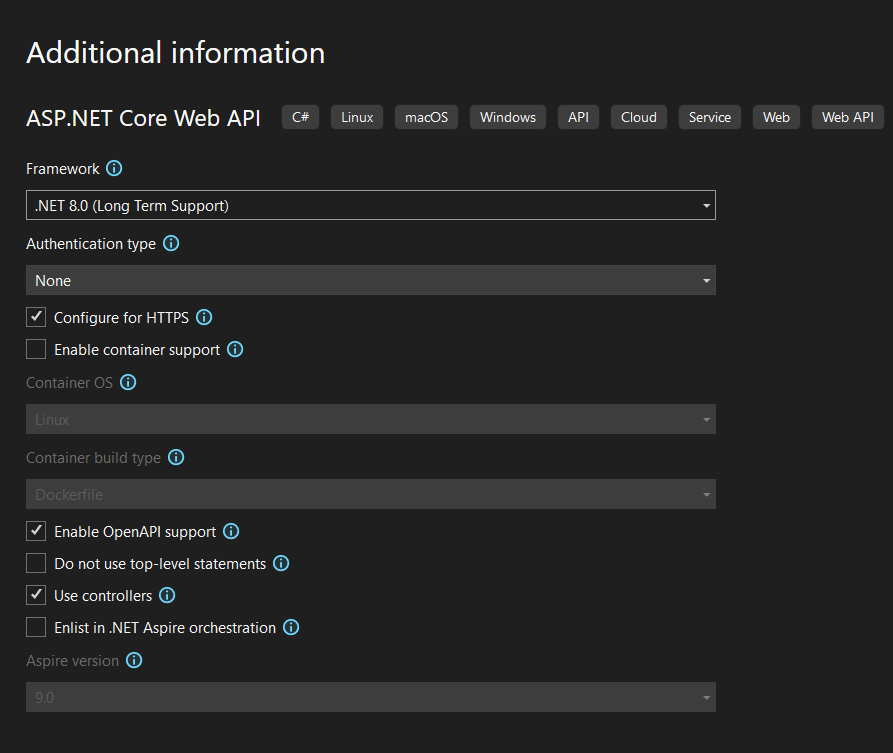
WEEK-3

ASP.NET Core 8.0 Web API

1. WebApi\_Handson

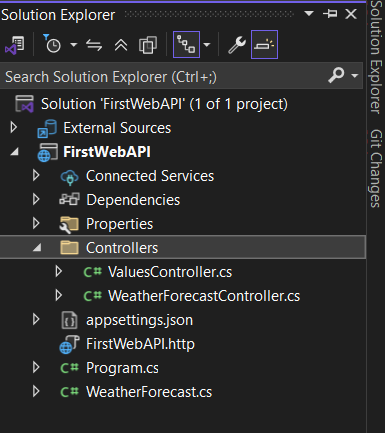
1. **First Web Api using .Net core**

**1. Project Creation:-**



**2. Generated Files:-**

After creation, the following files were generated:



**3. Adding a New Controller**

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

namespace FirstWebAPI.Controllers

{

public class ValuesController : Controller

{

// GET: ValuesController

public ActionResult Index()

{

return View();

}

// GET: ValuesController/Details/5

public ActionResult Details(int id)

{

return View();

}

// GET: ValuesController/Create

public ActionResult Create()

{

return View();

}

// POST: ValuesController/Create

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create(IFormCollection collection)

{

try

{

return RedirectToAction(nameof(Index));

}

catch

{

return View();

}

}

// GET: ValuesController/Edit/5

public ActionResult Edit(int id)

{

return View();

}

// POST: ValuesController/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit(int id, IFormCollection collection)

{

try

{

return RedirectToAction(nameof(Index));

}

catch

{

return View();

}

}

// GET: ValuesController/Delete/5

public ActionResult Delete(int id)

{

return View();

}

// POST: ValuesController/Delete/5

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Delete(int id, IFormCollection collection)

{

try

{

return RedirectToAction(nameof(Index));

}

catch

{

return View();

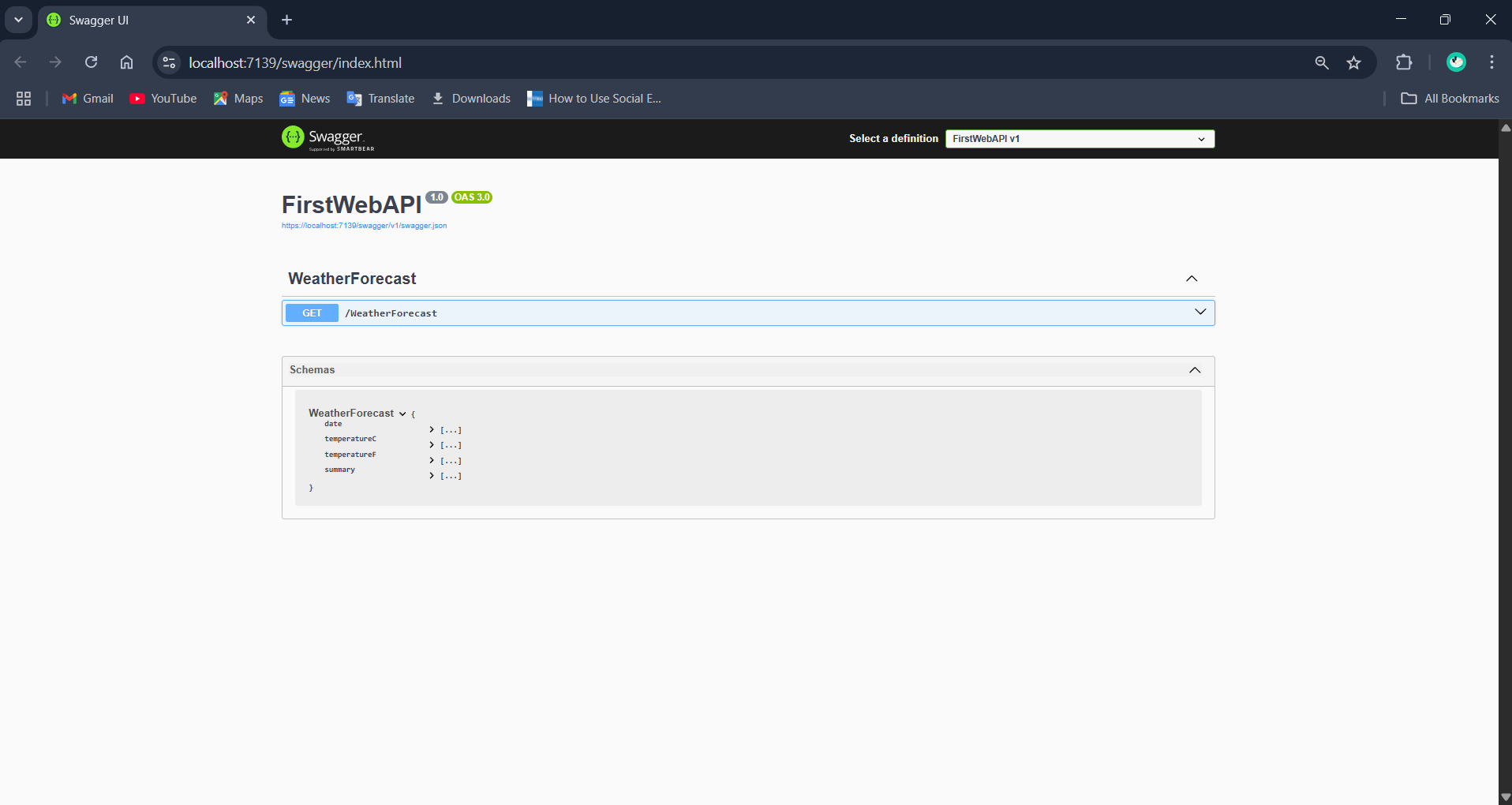
}

}

}

}

**4. Running the Application**



**5. Testing the GET Method**

**A screenshot of a computer

AI-generated content may be incorrect.**

2. WebApi\_Handson

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

Program.cs:-

using Microsoft.OpenApi.Models;

using System.ComponentModel;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "TBD",

TermsOfService = new Uri("https://example.com/terms"),

Contact = new OpenApiContact

{

Name = "John Doe",

Email = "john@xyzmail.com",

Url = new Uri("https://www.example.com")

},

License = new OpenApiLicense

{

Name = "License Terms",

Url = new Uri("https://www.example.com")

}

});

});

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

// specifying the Swagger JSON endpoint.

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

});

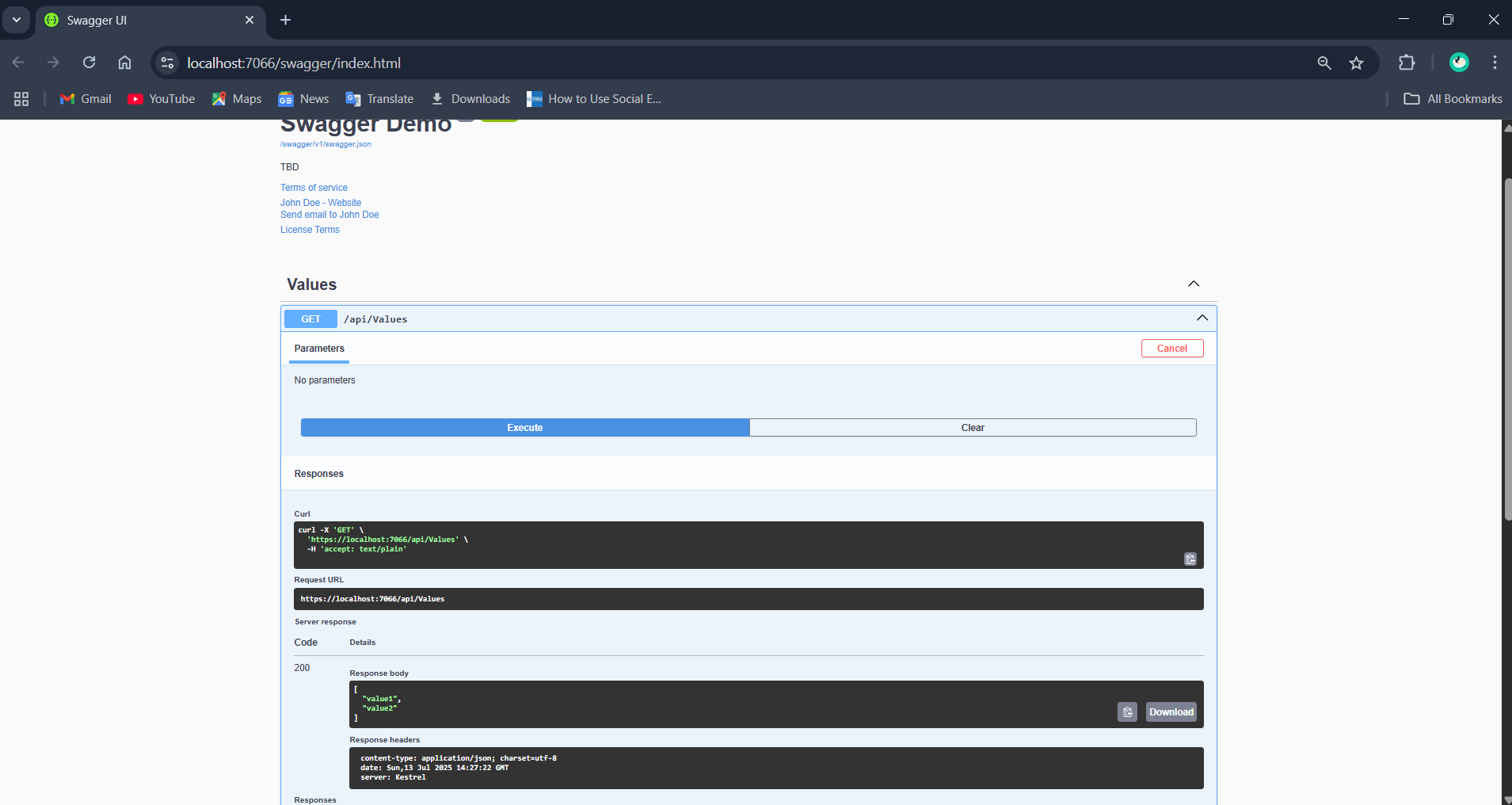
}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();



**2.Postman to Test Your Local Web API:-**

Employeecontroller.cs:-

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

namespace webAPI.Controllers

{

[ApiController]

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

public class EmployeeController : ControllerBase

{

[HttpGet]

public ActionResult<IEnumerable<string>> Get()

{

var employees = new List<string>

{

"Alice Johnson",

"Bob Smith",

"Charlie Brown"

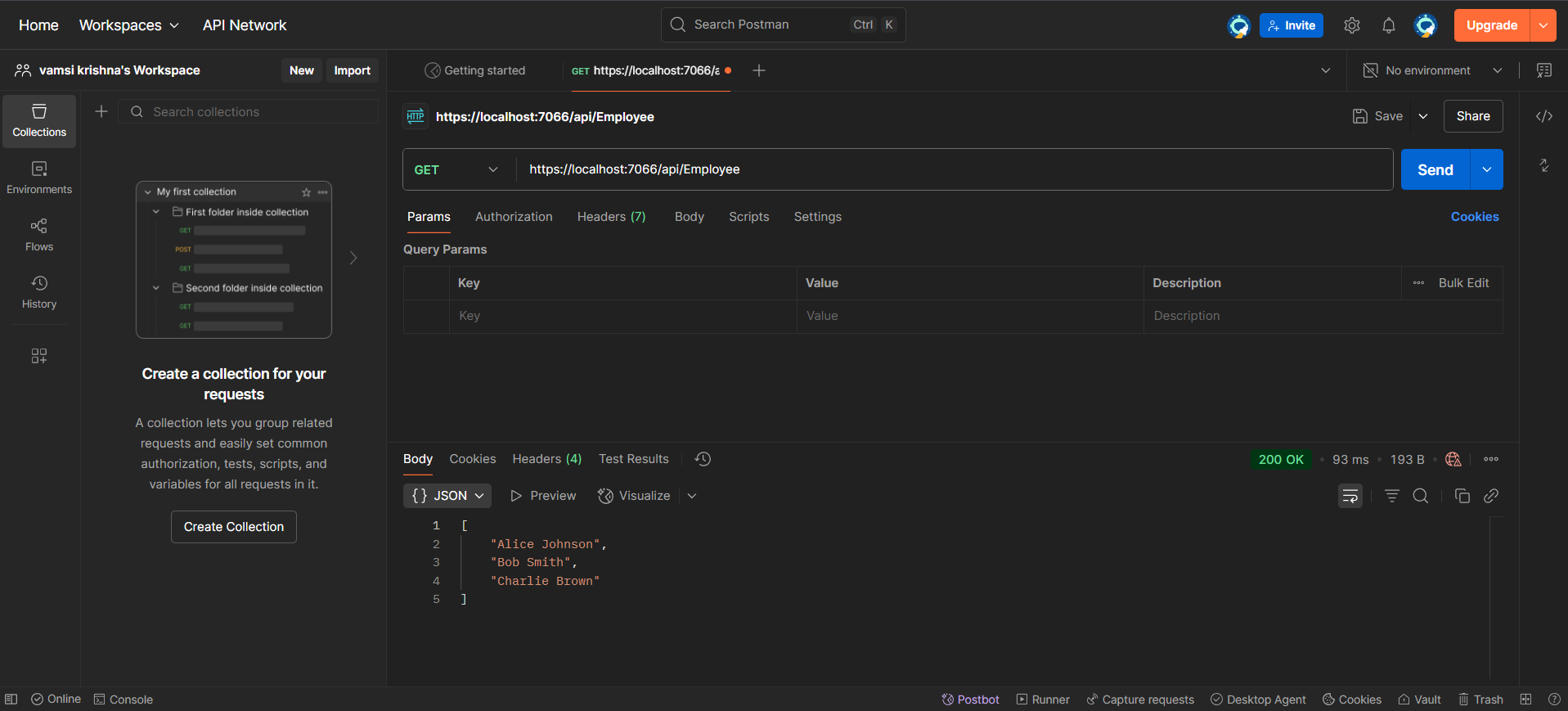
};

return Ok(employees);

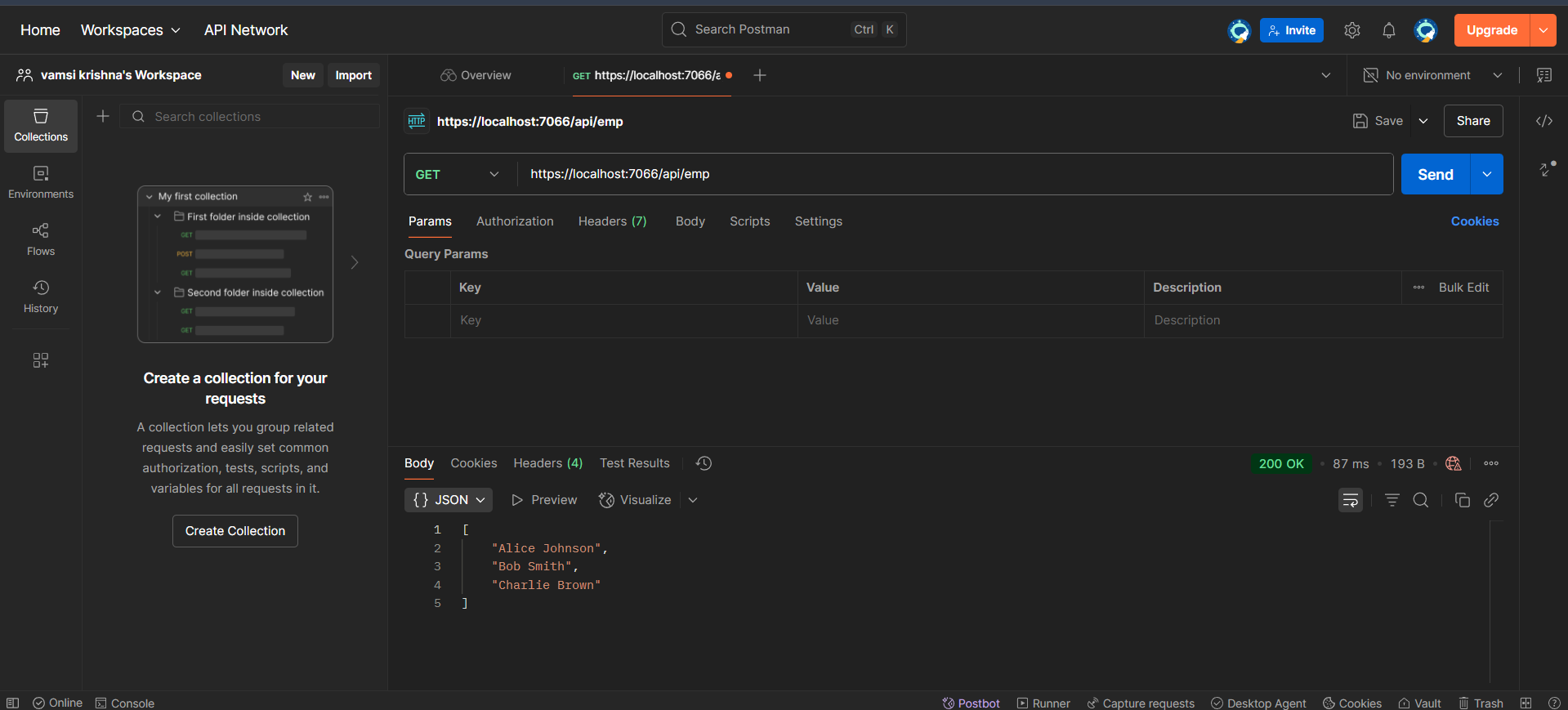
}

}

}



**3.Modify Controller Route to Emp and Test in Postman:-**



3. WebApi\_Handson

1. **Web Api using custom model class:-**

Employee.cs:-

using System;

using System.Collections.Generic;

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.Models;

namespace webAPI.Controllers

{

[ApiController]

[Route("api/emp")]

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 = "Alice",

Salary = 50000,

Permanent = true,

Department = new Department { Id = 101, Name = "HR" },

Skills = new List<Skill>

{

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

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

},

DateOfBirth = new DateTime(1990, 5, 23)

},

new Employee

{

Id = 2,

Name = "Bob",

Salary = 45000,

Permanent = false,

Department = new Department { Id = 102, Name = "Finance" },

Skills = new List<Skill>

{

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

},

DateOfBirth = new DateTime(1985, 3, 17)

}

};

}

[HttpGet]

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

public ActionResult<List<Employee>> Get()

{

return Ok(\_employees);

}

[HttpGet("standrad")]

public ActionResult<Employee> GetStandrad()

{

return \_employees[0];

}

[HttpPost]

public IActionResult Post([FromBody] Employee employee)

{

\_employees.Add(employee);

return CreatedAtAction(nameof(GetStandrad), new { id = employee.Id }, employee);

}

[HttpPut("{id}")]

public IActionResult Put(int id, [FromBody] Employee employee)

{

var index = \_employees.FindIndex(e => e.Id == id);

if (index == -1) return NotFound();

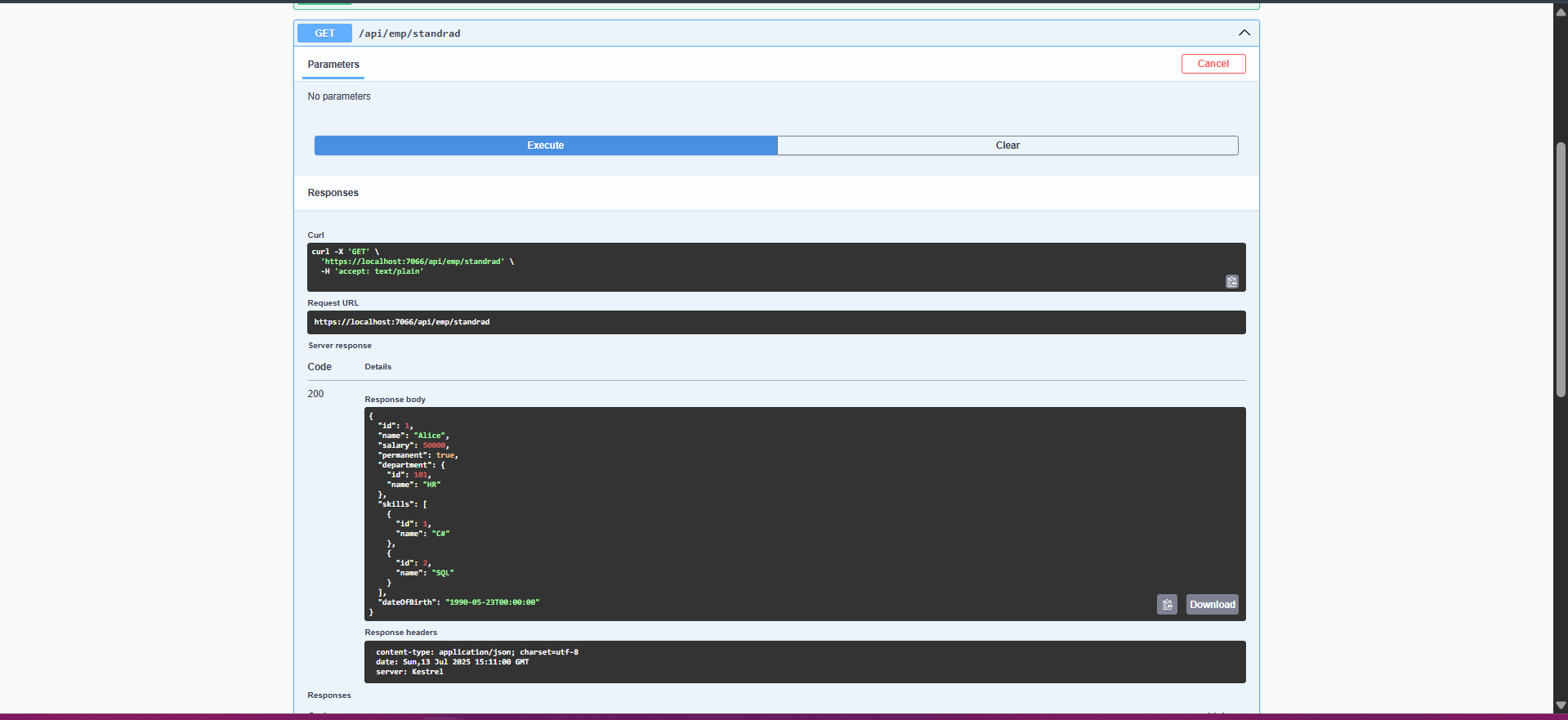
\_employees[index] = employee;

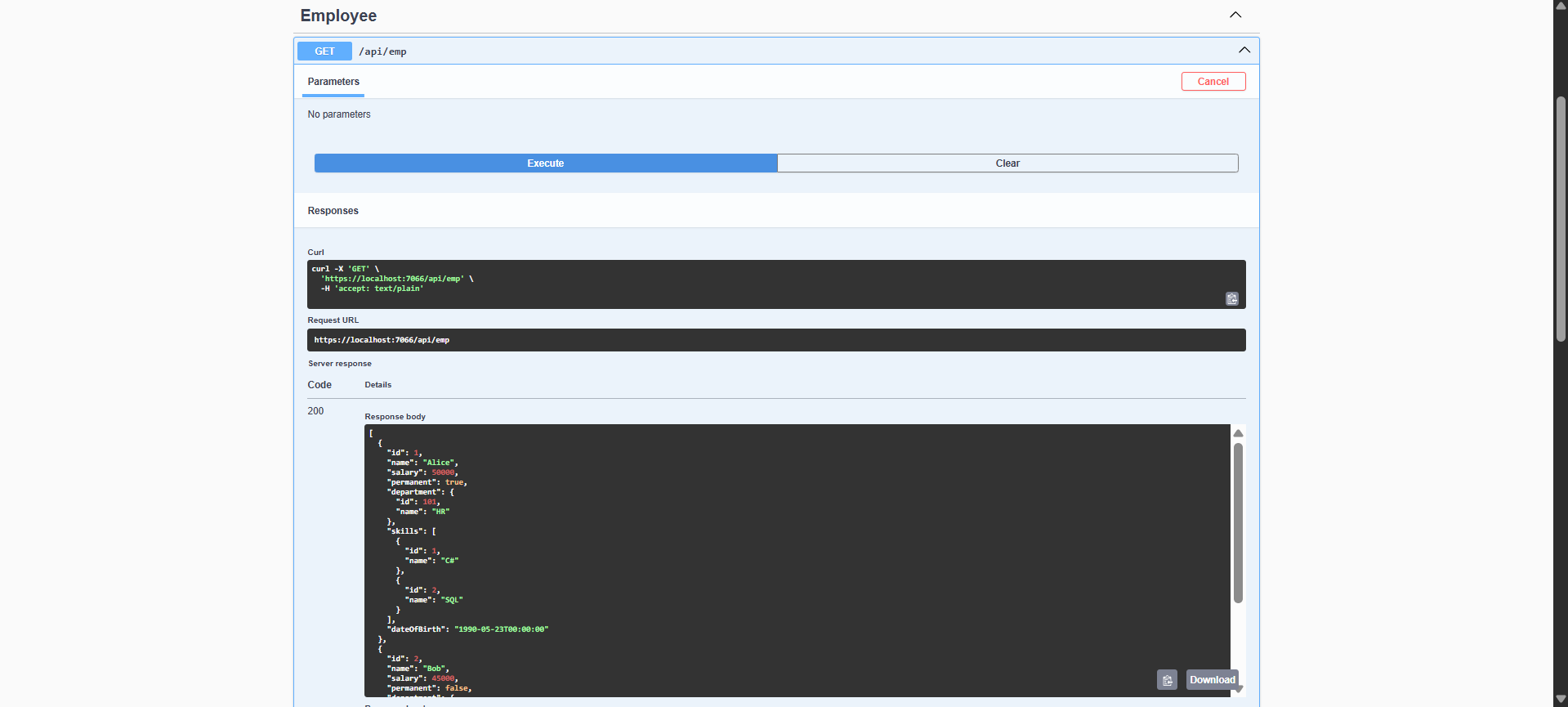
return NoContent();

}

}

}





1. **Create a Custom action filter for Authorization.**

**CustomAuthFilter.cs:-**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System.Linq;

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");

}

base.OnActionExecuting(context);

}

}

}

1. **Custom Exception filter**

CustomExceptionFilter.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

using System;

using System.IO;

namespace webAPI.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

var exception = context.Exception;

var logPath = "exception\_log.txt";

File.AppendAllText(logPath, $"[{DateTime.Now}] {exception.Message}{Environment.NewLine}");

context.Result = new ObjectResult("Something went wrong, please try again later.")

{

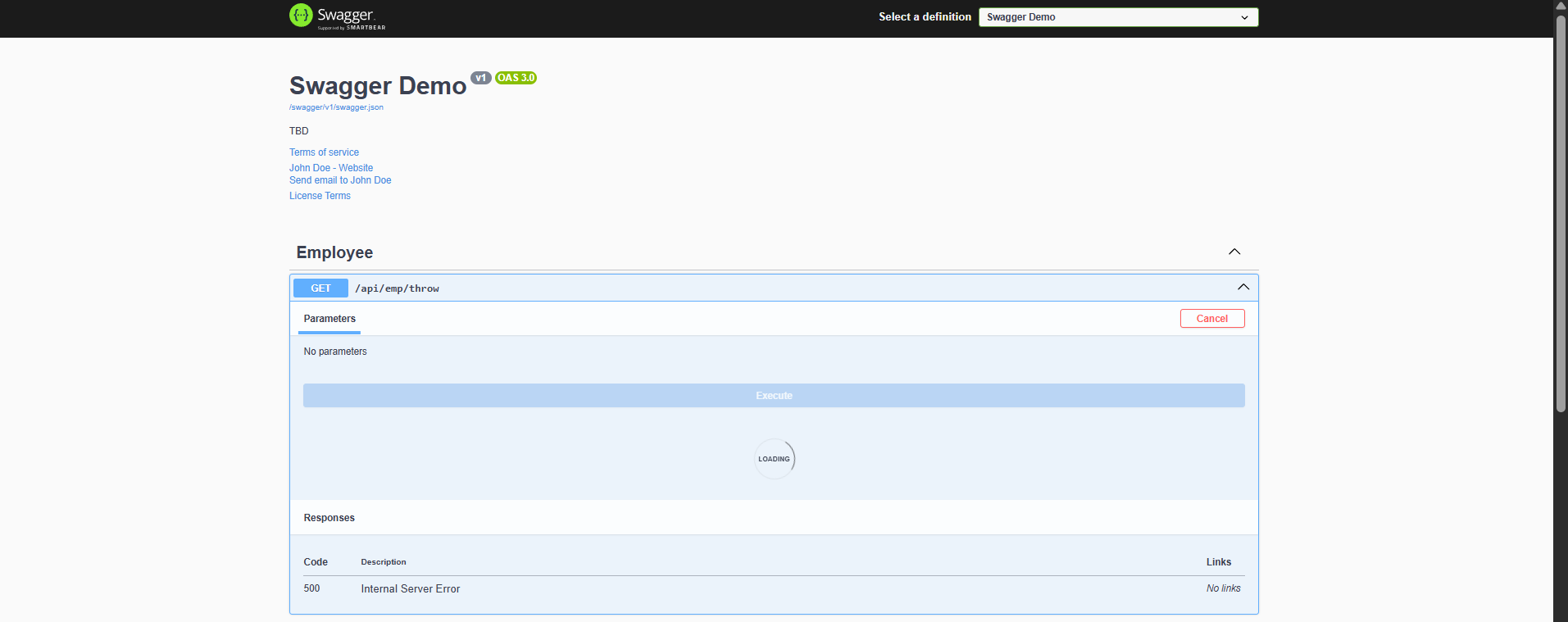
StatusCode = 500

};

}

}

}



**4. WebApi\_Handson**

**1.Web Api CRUD operation**

Replace your existing Put() with this full implementation:

[HttpPut("{id}")]

[ProducesResponseType(typeof(Employee), 200)]

[ProducesResponseType(400)]

public ActionResult<Employee> Put(int id, [FromBody] Employee employee)

{

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 = employee.Name;

emp.Salary = employee.Salary;

emp.Permanent = employee.Permanent;

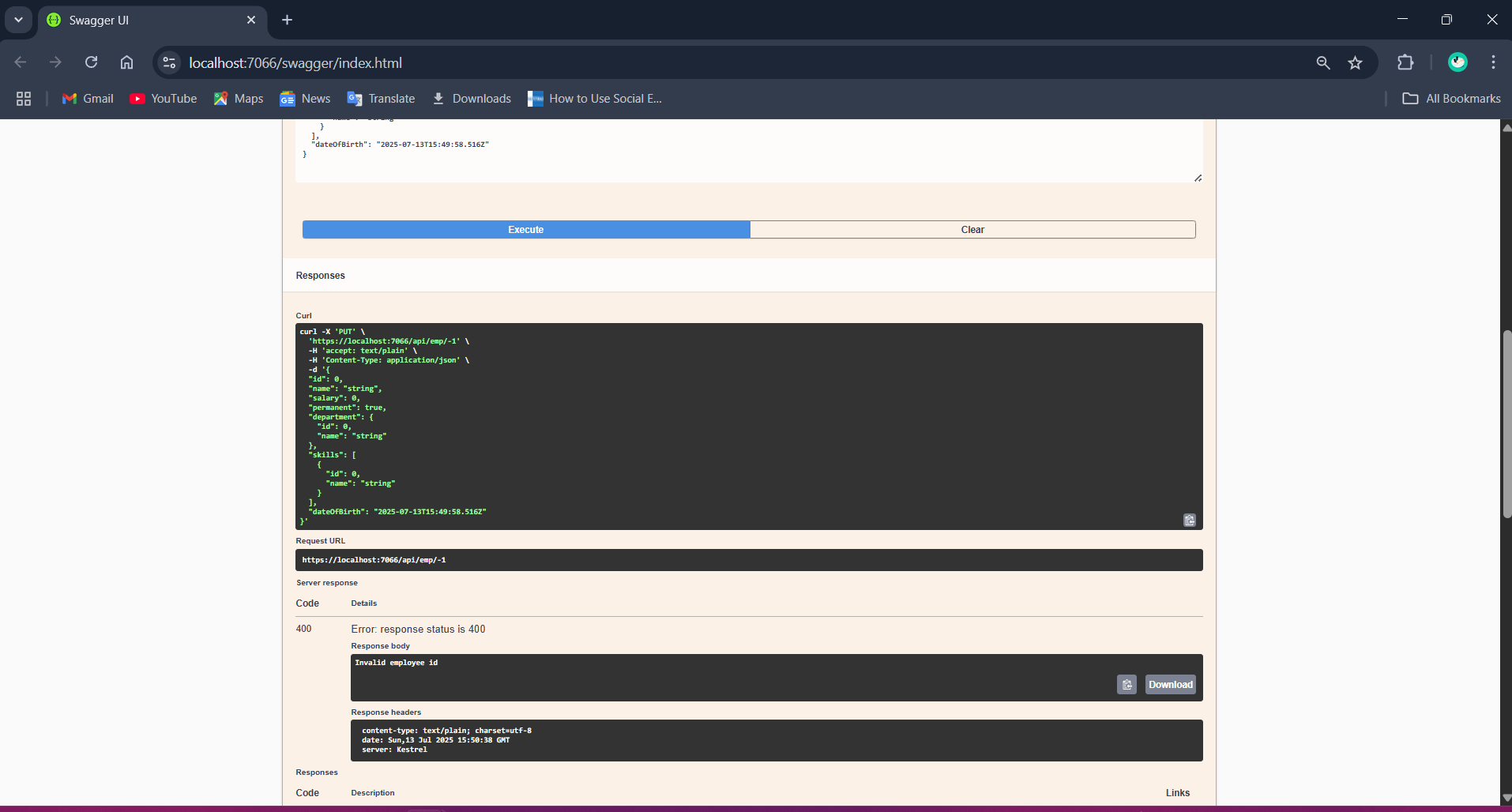
emp.Department = employee.Department;

emp.Skills = employee.Skills;

emp.DateOfBirth = employee.DateOfBirth;

return Ok(emp); // Return updated employee object

}



**5. WebApi\_Handson**

**1.JsonWebToken:**

**Program.cs:-**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using System.Text;

using webAPI.Filters;

var builder = WebApplication.CreateBuilder(args);

string securityKey = "mysuperdupersecretkey1234567890123456";

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

builder.Services.AddAuthentication(x =>

{

x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

x.DefaultSignInScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(JwtBearerDefaults.AuthenticationScheme, x =>

{

x.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = "mySystem",

ValidAudience = "myUsers",

IssuerSigningKey = symmetricSecurityKey

};

});

builder.Services.AddControllers(options =>

{

options.Filters.Add<CustomExceptionFilter>();

});

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "TBD",

TermsOfService = new Uri("https://example.com/terms"),

Contact = new OpenApiContact

{

Name = "John Doe",

Email = "john@xyzmail.com",

Url = new Uri("https://www.example.com")

},

License = new OpenApiLicense

{

Name = "License Terms",

Url = new Uri("https://www.example.com")

}

});

c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

Description = "JWT Authorization header using the Bearer scheme. Example: \"Bearer abc123\"",

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 app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

}

app.UseHttpsRedirection();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

**Authcontroller.cs:-**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace webAPI.Controllers

{

[AllowAnonymous]

[ApiController]

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

public class AuthController : ControllerBase

{

private string GenerateJSONWebToken(int userId, string userRole)

{

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("mysuperdupersecretkey1234567890123456"));

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

var claims = new List<Claim>

{

new Claim(ClaimTypes.Role, userRole),

new Claim("UserId", userId.ToString())

};

var token = new JwtSecurityToken(

issuer: "mySystem",

audience: "myUsers",

claims: claims,

expires: DateTime.Now.AddMinutes(10),

signingCredentials: credentials

);

return new JwtSecurityTokenHandler().WriteToken(token);

}

[HttpGet("token")]

public IActionResult GetToken()

{

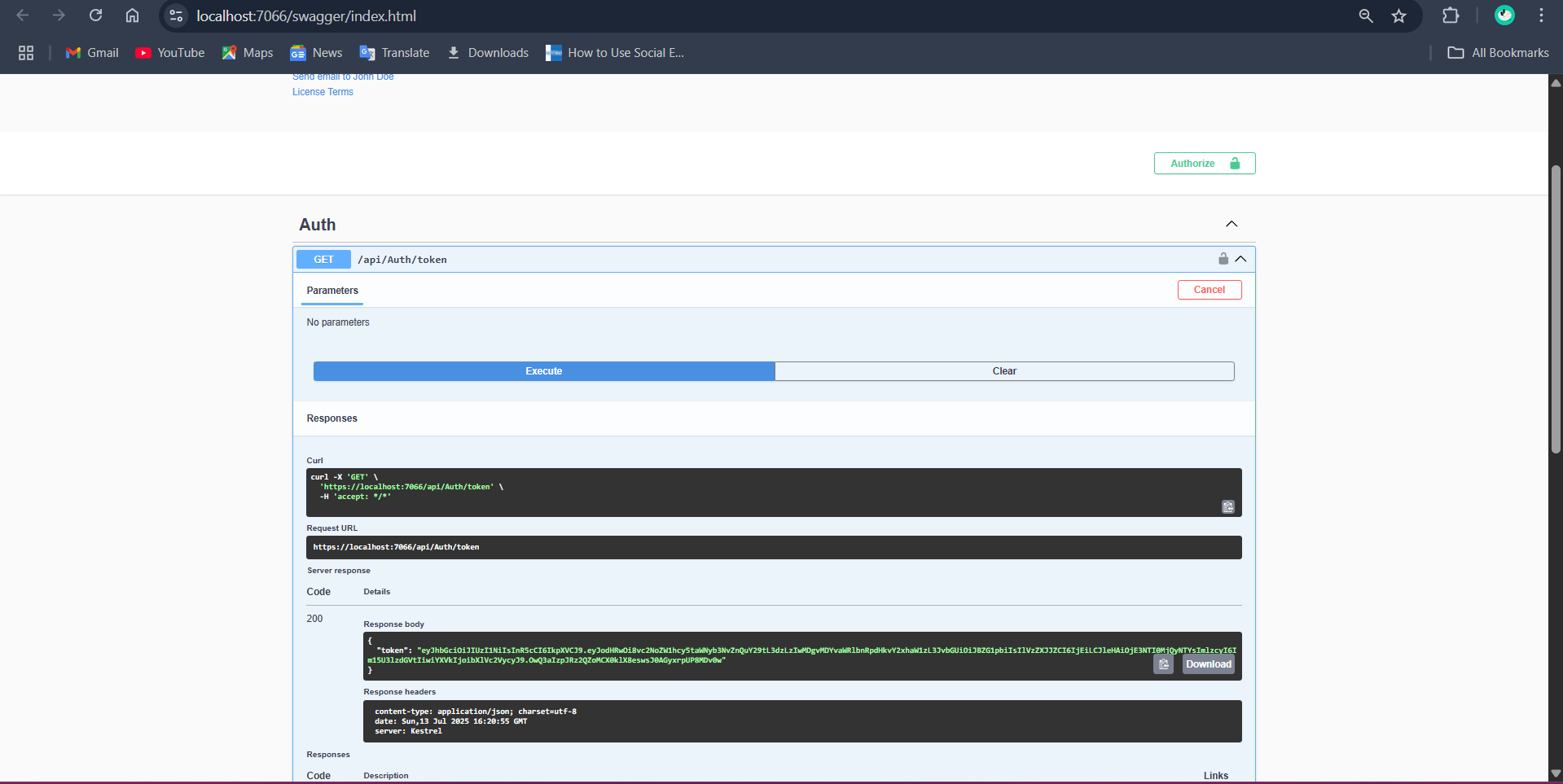
var token = GenerateJSONWebToken(1, "Admin");

return Ok(new { Token = token });

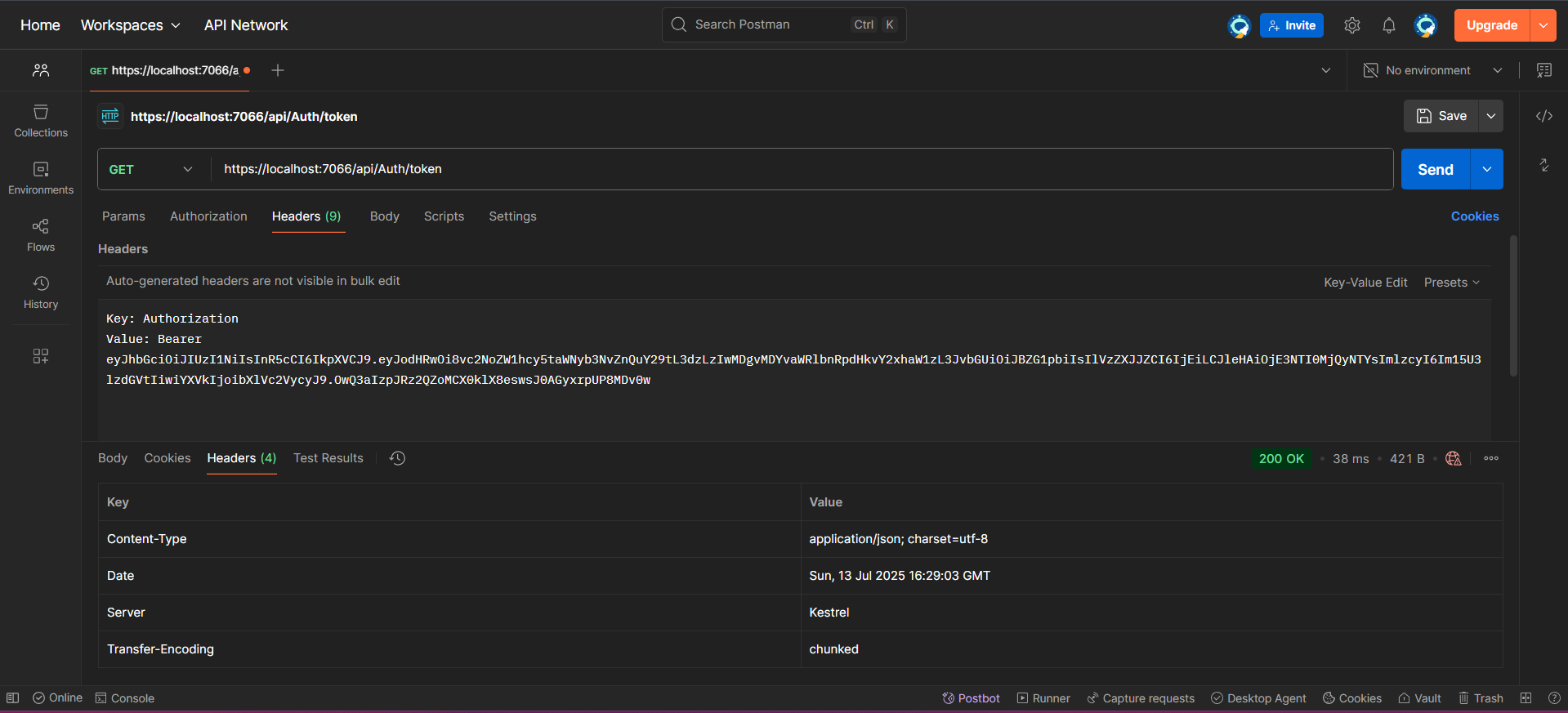
}

}

}



**2.Use the JWT generated thru the AuthController to be used in POSTMAN request.**

****

**3.Check for JWT expiration:**

Modify GenerateJSONWebToken() Method in AuthController

expires: DateTime.Now.AddMinutes(2),

