Week 4 : ASP.Net core 8.0 Web API

Lab 1 :

API Schema:

namespace MyFirstWebAPI.model

{

public class Student

{

public int Id { get; set; }

public string Name { get; set; }

public string Department { get; set; }

}

}

AppDbContext :

using Microsoft.EntityFrameworkCore;

using MyFirstWebAPI.model;

namespace MyFirstWebAPI.data

{

public class AppDbContext : DbContext

{

public AppDbContext(DbContextOptions<AppDbContext> options)

: base(options) { }

public DbSet<Student> Students { get; set; }

}

}

Program.cs

using Microsoft.EntityFrameworkCore;

using MyFirstWebAPI.data;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

// Add Swagger

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

// Add DbContext

builder.Services.AddDbContext<AppDbContext>(options =>

options.UseInMemoryDatabase("StudentDb")); // Ensure 'Microsoft.EntityFrameworkCore.InMemory' package is installed

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

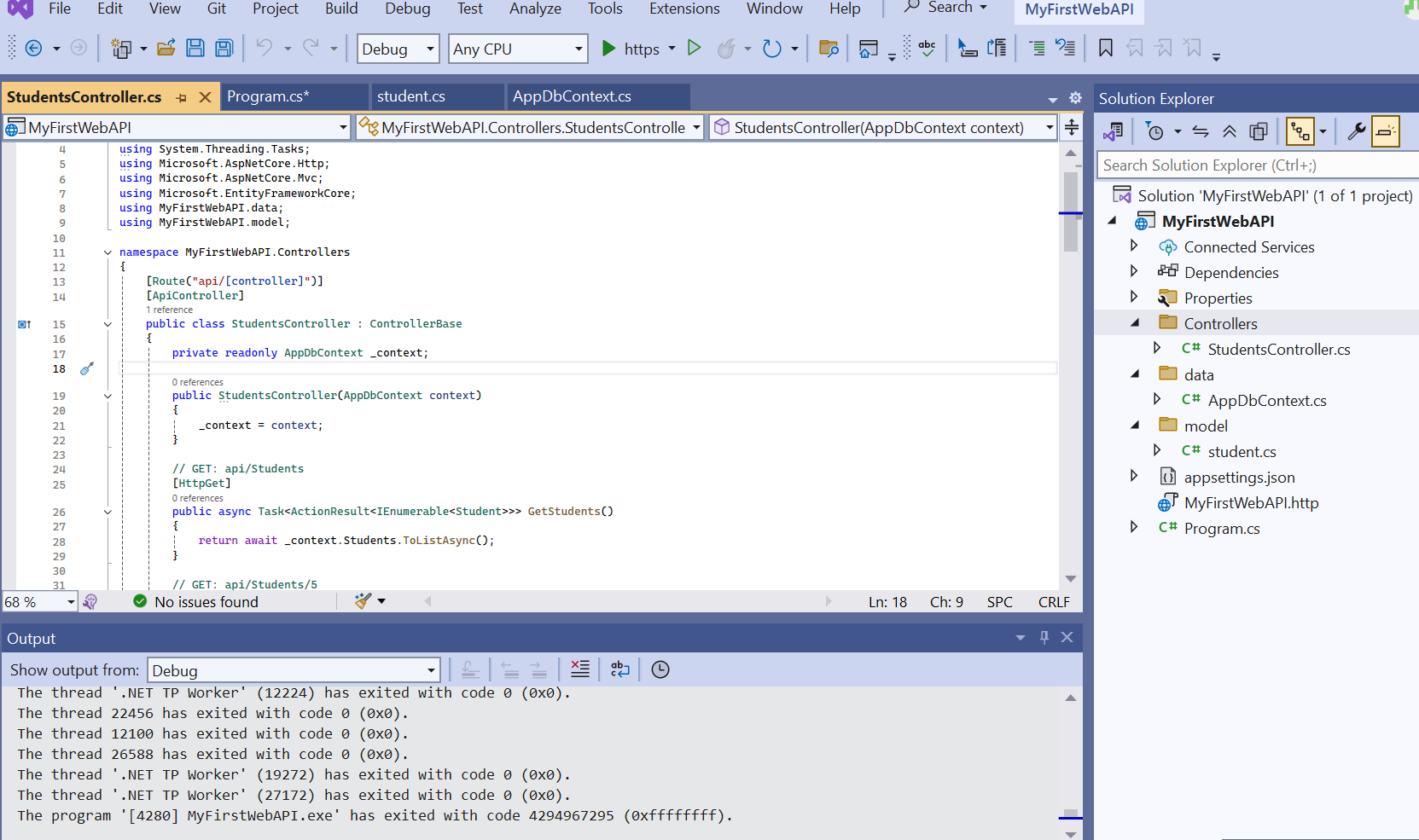
app.UseHttpsRedirection();

app.UseAuthorization();

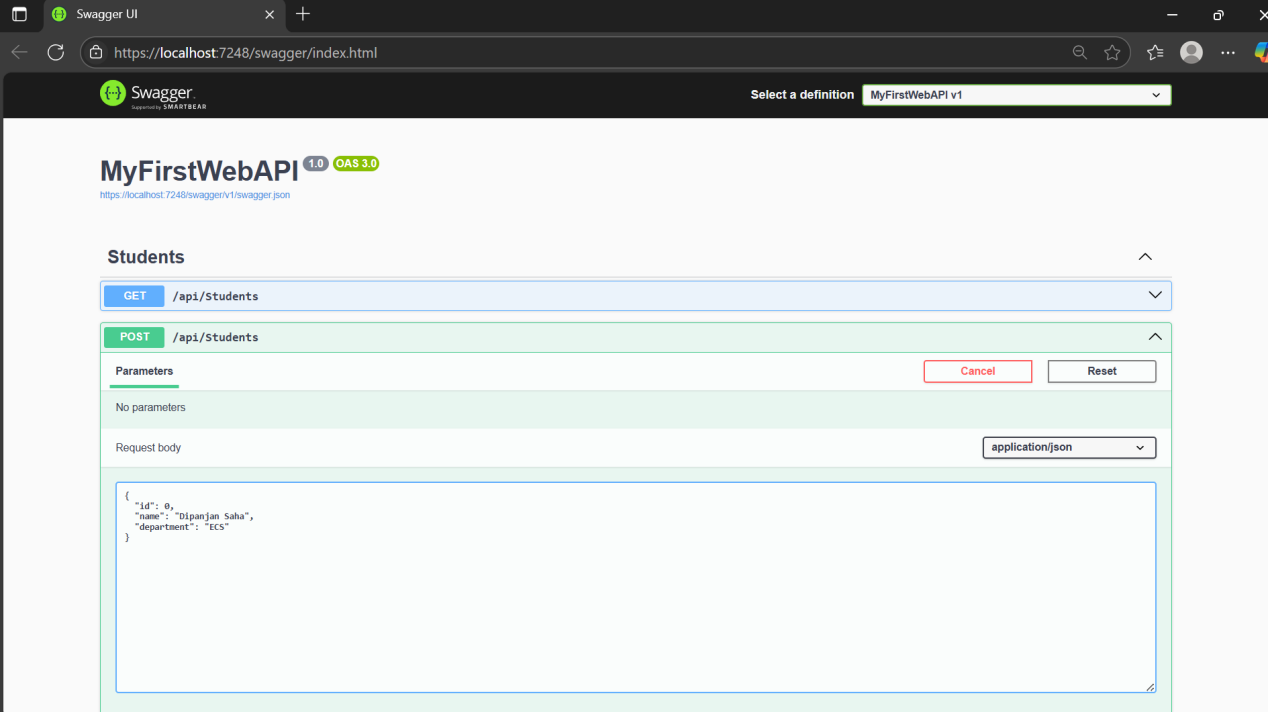
app.MapControllers();

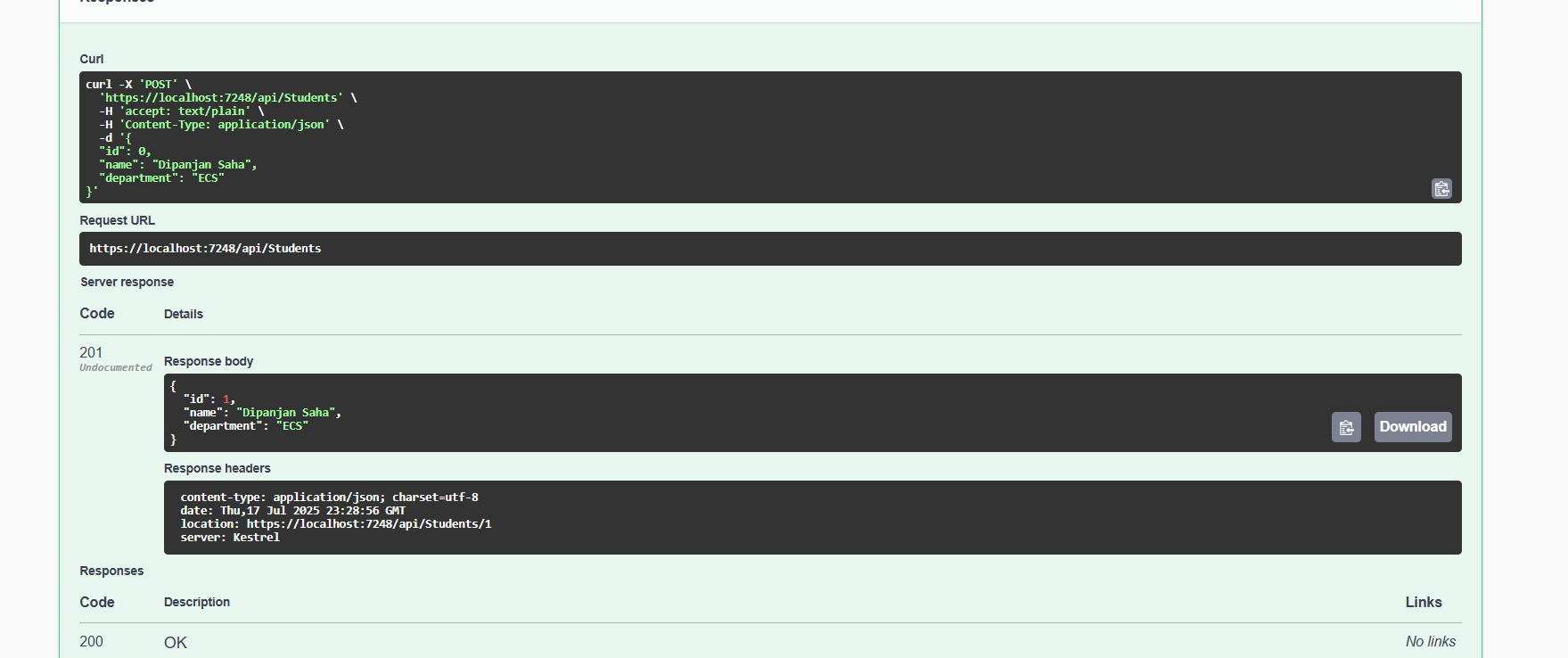
app.Run();

In controller add a scafold item with read/write permission.



The final output :





Lab 2

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

Program.cs

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

// Add Swagger

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 = "Dipanjan Saha",

Email = "dipanjan1604@gmail.com",

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

},

License = new OpenApiLicense

{

Name = "License Terms",

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

}

});

});

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

});

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

Controller code:

using Microsoft.AspNetCore.Mvc;

// For more information on enabling Web API for empty projects, visit https://go.microsoft.com/fwlink/?LinkID=397860

namespace SwaggerDemoAPI.Controllers

{

[Route("api/Employee")]

[ApiController]

public class EmployeeController : ControllerBase

{

// GET: api/<EmployeeController>

[HttpGet]

public IEnumerable<string> Get()

{

return new string[] {

"Dipanjan Saha",

"Dipprovash Saha"

};

}

// GET api/<EmployeeController>/5

[HttpGet("{id}")]

public string Get(int id)

{

return "value";

}

// POST api/<EmployeeController>

[HttpPost]

public IActionResult GetEmployees()

{

var employees = new[]

{

new { Id = 1, Name = "Dipanjan Saha", Department = "dotnet developer" },

new { Id = 2, Name = "Ashis Patra", Department = "IT" }

};

return Ok(employees);

}

// PUT api/<EmployeeController>/5

[HttpPut("{id}")]

public void Put(int id, [FromBody] string value)

{

}

// DELETE api/<EmployeeController>/5

[HttpDelete("{id}")]

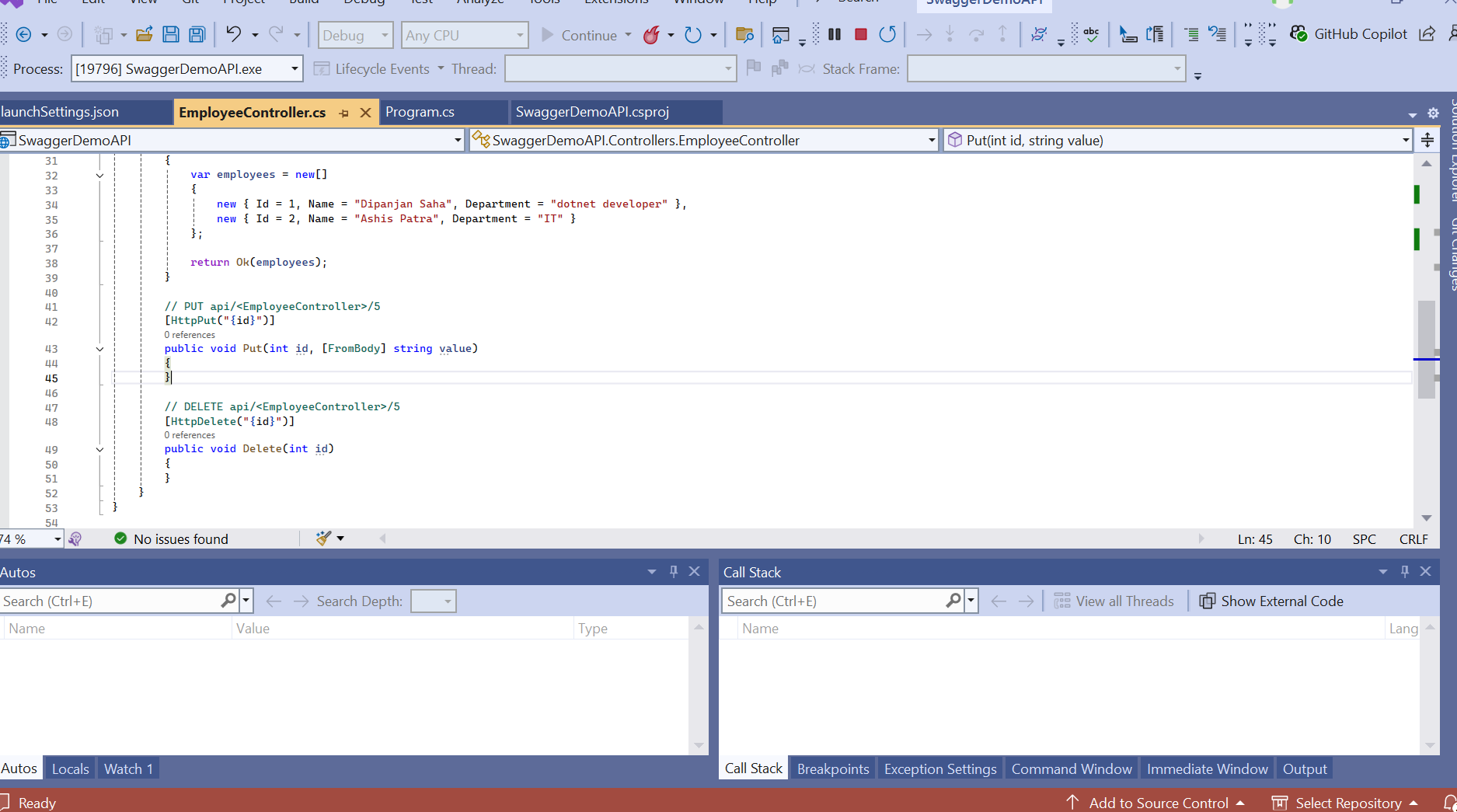
public void Delete(int id)

{

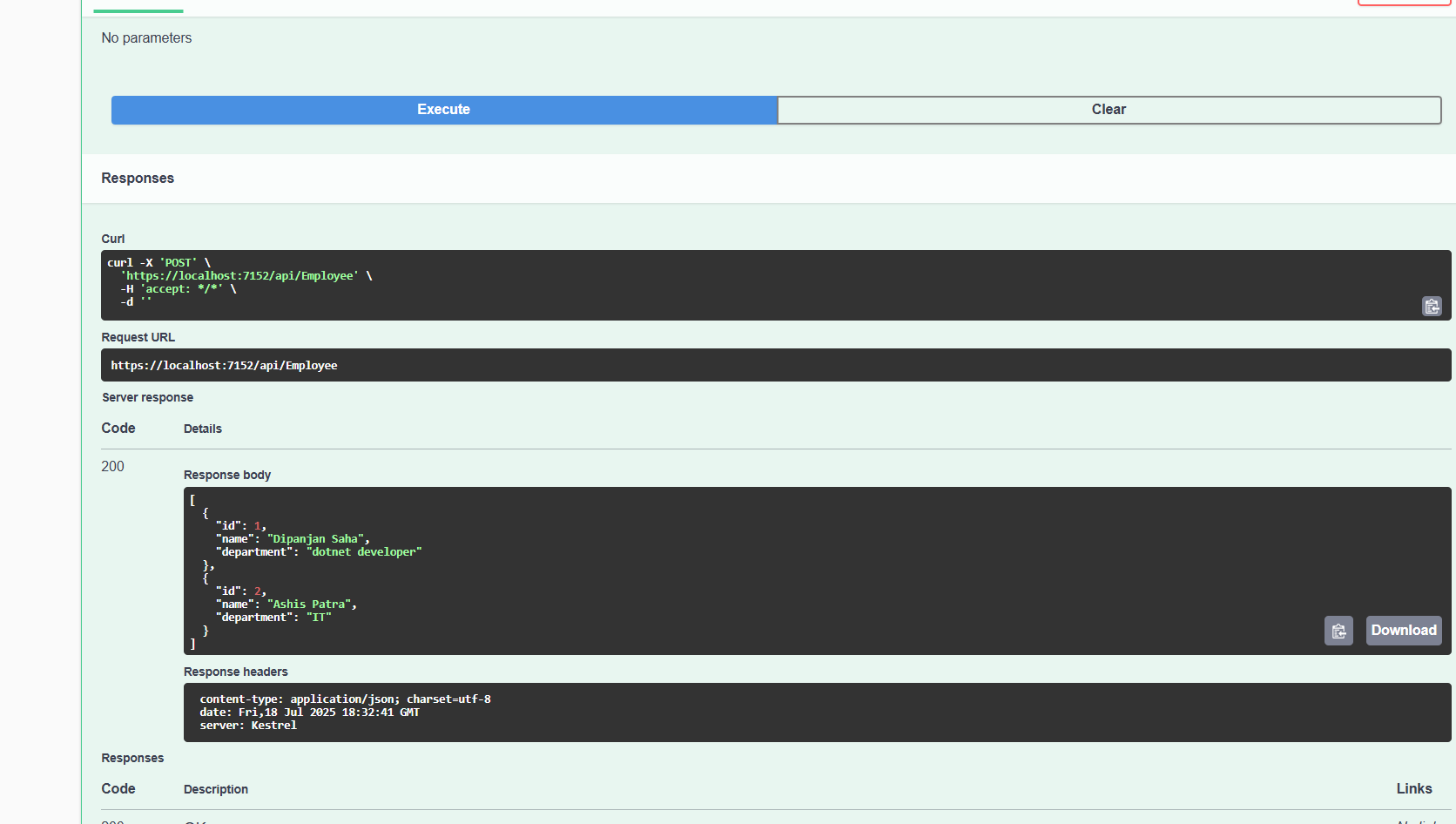
}

}

}

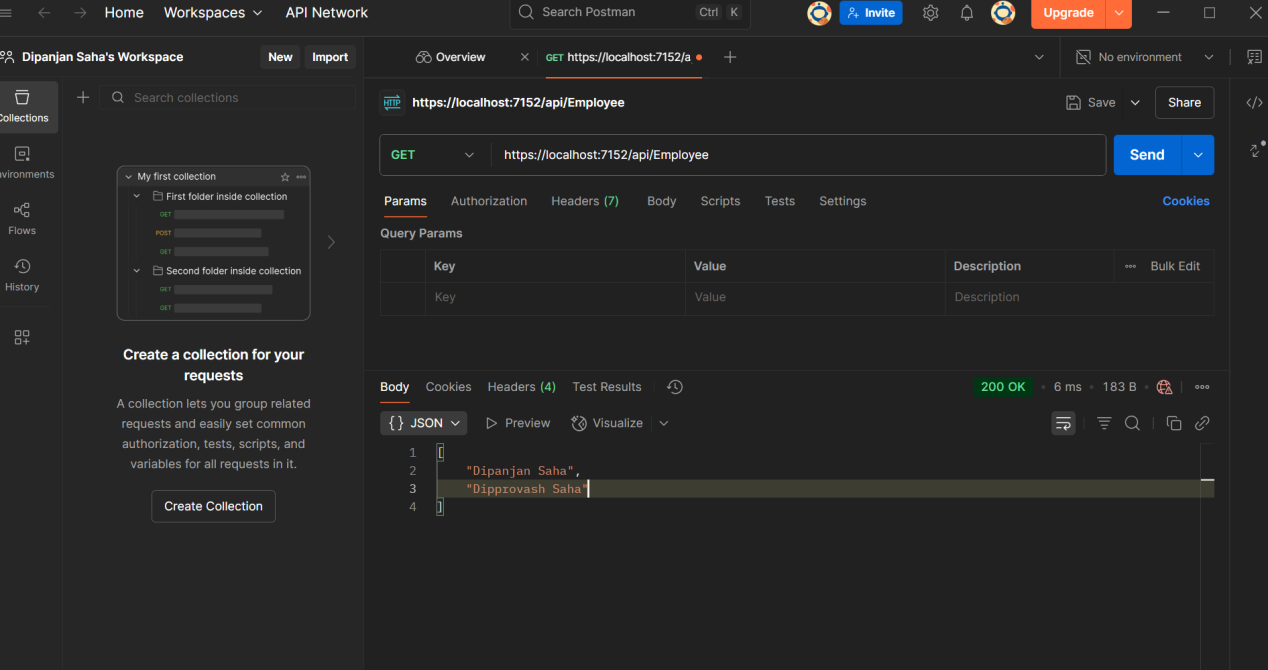


Swagger UI

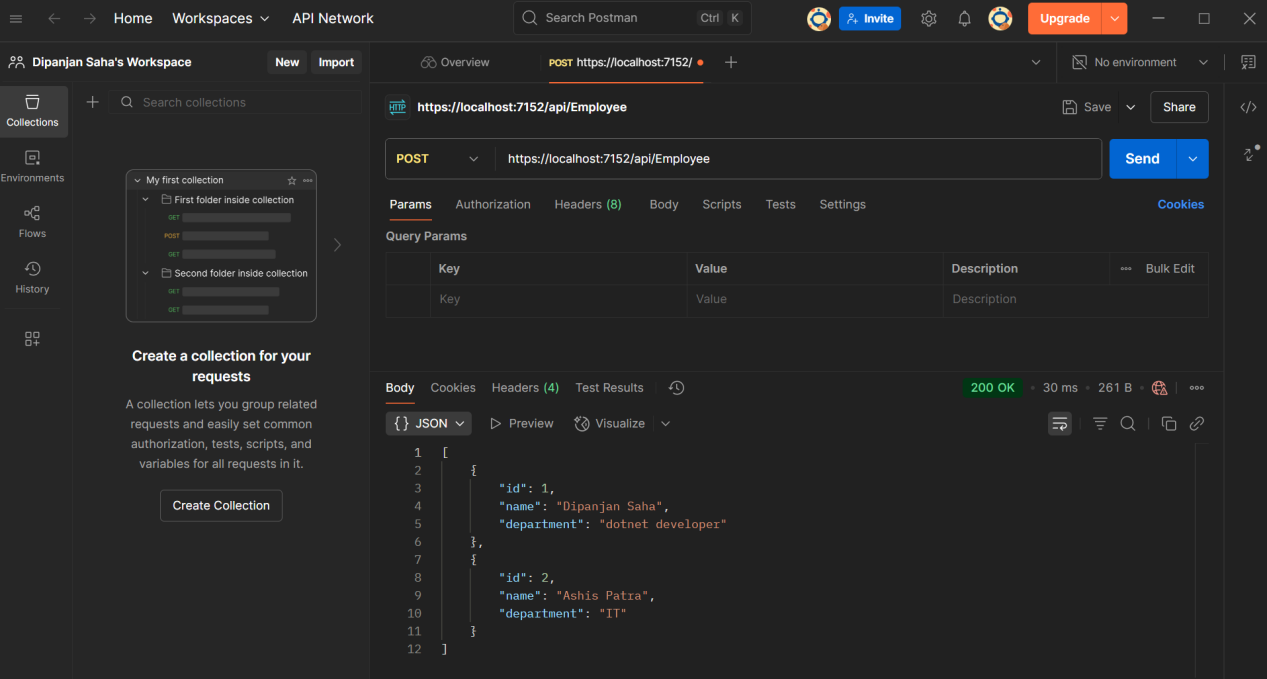


Postman:

Get:



Post:



Lab 3

1. Web Api using custom model class

namespace SwaggerDemoAPI.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 Name { get; set; }

}

public class Skill

{

public int Id { get; set; }

public string Name { get; set; }

}

}

Controller

namespace SwaggerDemoAPI.Controllers

{

[ApiController]

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

[CustomAuthFilter] // Attach the custom auth filter

public class EmployeeController : ControllerBase

{

private static List<Employee> employees = new List<Employee>();

public EmployeeController()

{

if (employees.Count == 0)

employees = GetStandardEmployeeList();

}

[HttpGet]

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

[ProducesResponseType(500)]

public ActionResult<List<Employee>> GetStandard()

{

// Simulate an error to test exception filter

// throw new Exception("Test exception");

return Ok(employees);

}

[HttpPost]

public IActionResult AddEmployee([FromBody] Employee emp)

{

employees.Add(emp);

return Ok(emp);

}

[HttpPut("{id}")]

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

{

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

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

emp.Name = updatedEmp.Name;

emp.Salary = updatedEmp.Salary;

emp.Permanent = updatedEmp.Permanent;

emp.Department = updatedEmp.Department;

emp.Skills = updatedEmp.Skills;

emp.DateOfBirth = updatedEmp.DateOfBirth;

return Ok(emp);

}

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(1995, 5, 12)

}

};

}

}

}

1. Created a Custom action filter for Authorization.

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace SwaggerDemoAPI.Filters

{

public class CustomAuthFilter : ActionFilterAttribute

{

public override void OnActionExecuting(ActionExecutingContext context)

{

var hasAuth = context.HttpContext.Request.Headers.TryGetValue("Authorization", out var token);

if (!hasAuth)

{

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");

return;

}

base.OnActionExecuting(context);

}

}

}

Custom Exception filter

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Filters;

namespace SwaggerDemoAPI.Filters

{

public class CustomExceptionFilter : IExceptionFilter

{

public void OnException(ExceptionContext context)

{

string logPath = Path.Combine(Directory.GetCurrentDirectory(), "Logs");

if (!Directory.Exists(logPath))

Directory.CreateDirectory(logPath);

string logFile = Path.Combine(logPath, "error.log");

File.AppendAllText(logFile, $"[{DateTime.Now}] {context.Exception.Message}\n");

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

{

StatusCode = 500

};

}

}

}

Program.cs

using Microsoft.OpenApi.Models;

using SwaggerDemoAPI.Filters;

var builder = WebApplication.CreateBuilder(args);

// Register controllers and add global exception filter

builder.Services.AddControllers(options =>

{

options.Filters.Add<CustomExceptionFilter>();

});

// Register Swagger configuration

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo

{

Title = "Swagger Demo",

Version = "v1",

Description = "Employee API for learning Web API with Swagger UI.",

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

Contact = new OpenApiContact

{

Name = "Dipanjan Saha",

Email = "dipanjan1604@gmail.com",

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

},

License = new OpenApiLicense

{

Name = "License Terms",

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

}

});

// Optional: Add support for Authorization header in Swagger if you later use [Authorize]

c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

Name = "Authorization",

Type = SecuritySchemeType.ApiKey,

Scheme = "Bearer",

BearerFormat = "JWT",

In = ParameterLocation.Header,

Description = "Enter 'Bearer' followed by your token in the input box below."

});

c.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference { Type = ReferenceType.SecurityScheme, Id = "Bearer" }

},

new string[] {}

}

});

});

var app = builder.Build();

// Enable Swagger UI in Development

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

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

c.RoutePrefix = string.Empty; // optional: opens Swagger UI at root

});

}

app.UseHttpsRedirection();

// Enable authorization middleware (not authentication in this sample)

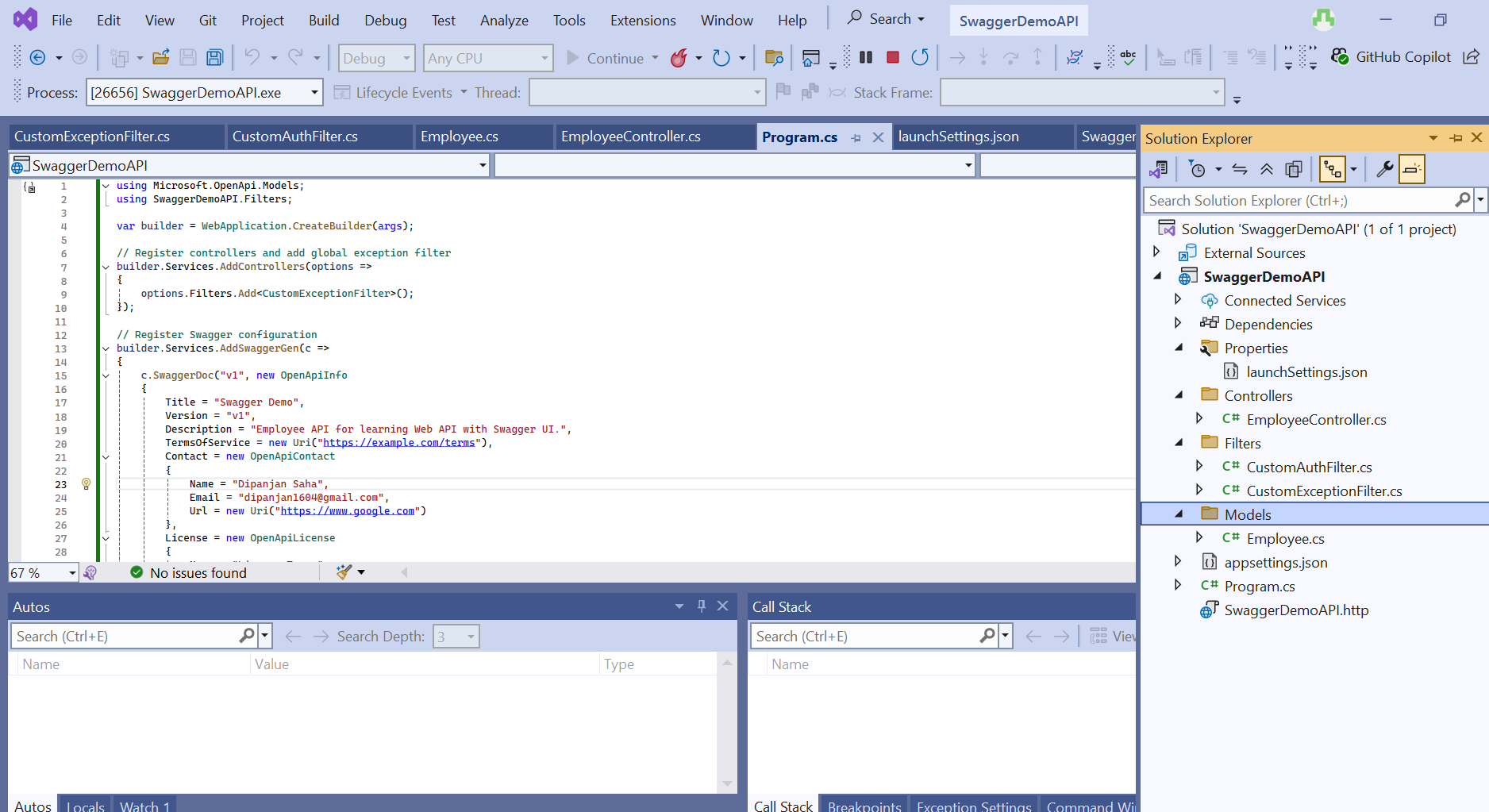
app.UseAuthorization();

// Map controller endpoints

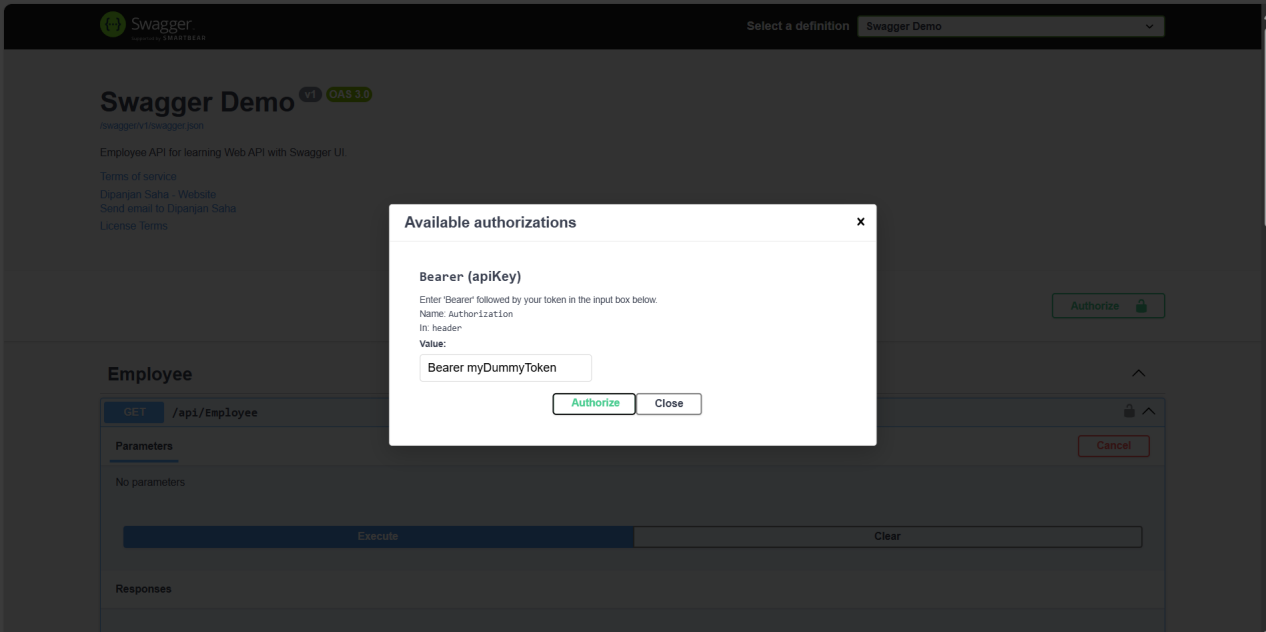
app.MapControllers();

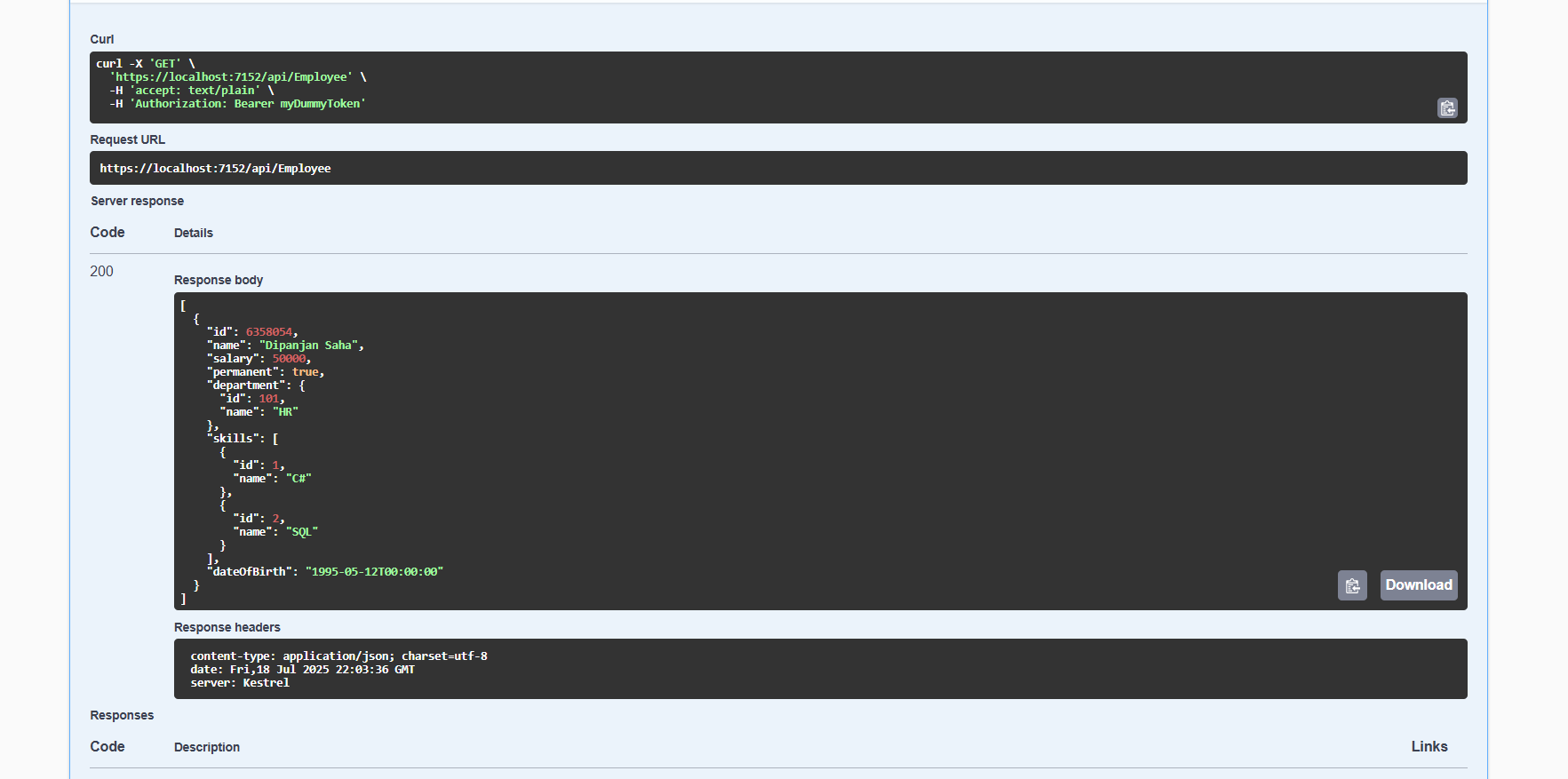
// Run the app

app.Run();

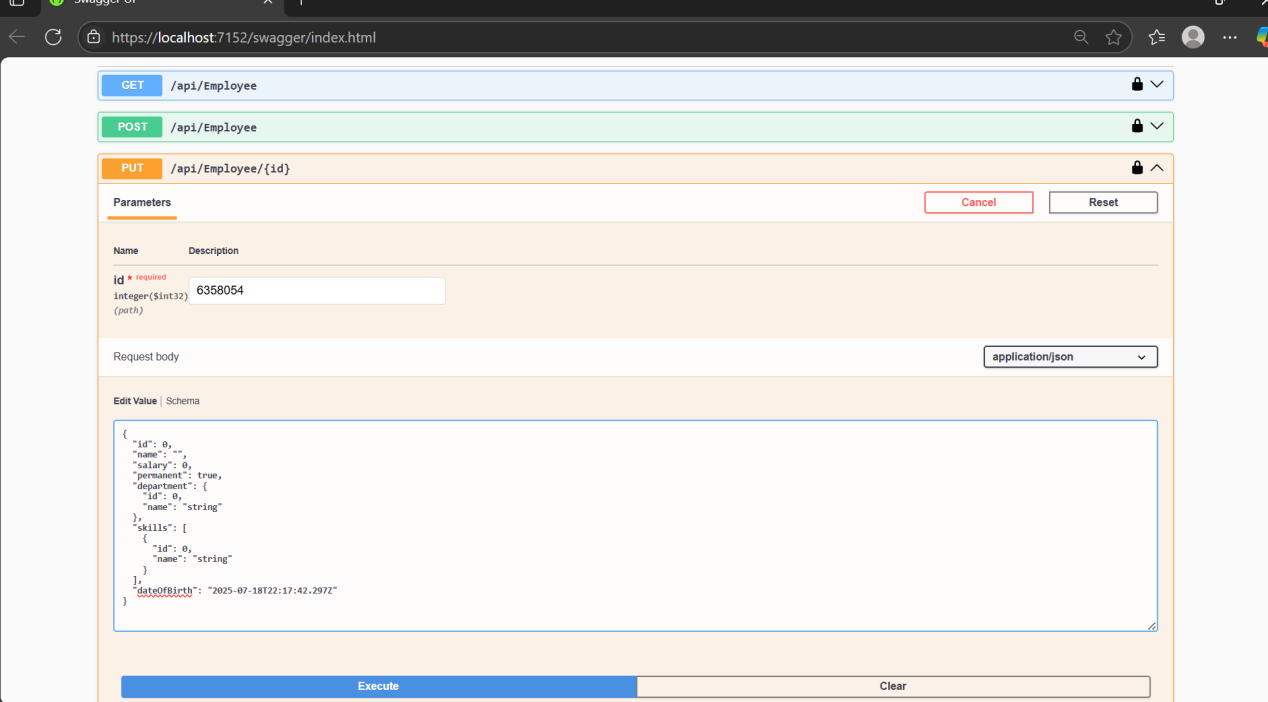
execution:

Swagger UI output USING authorization token

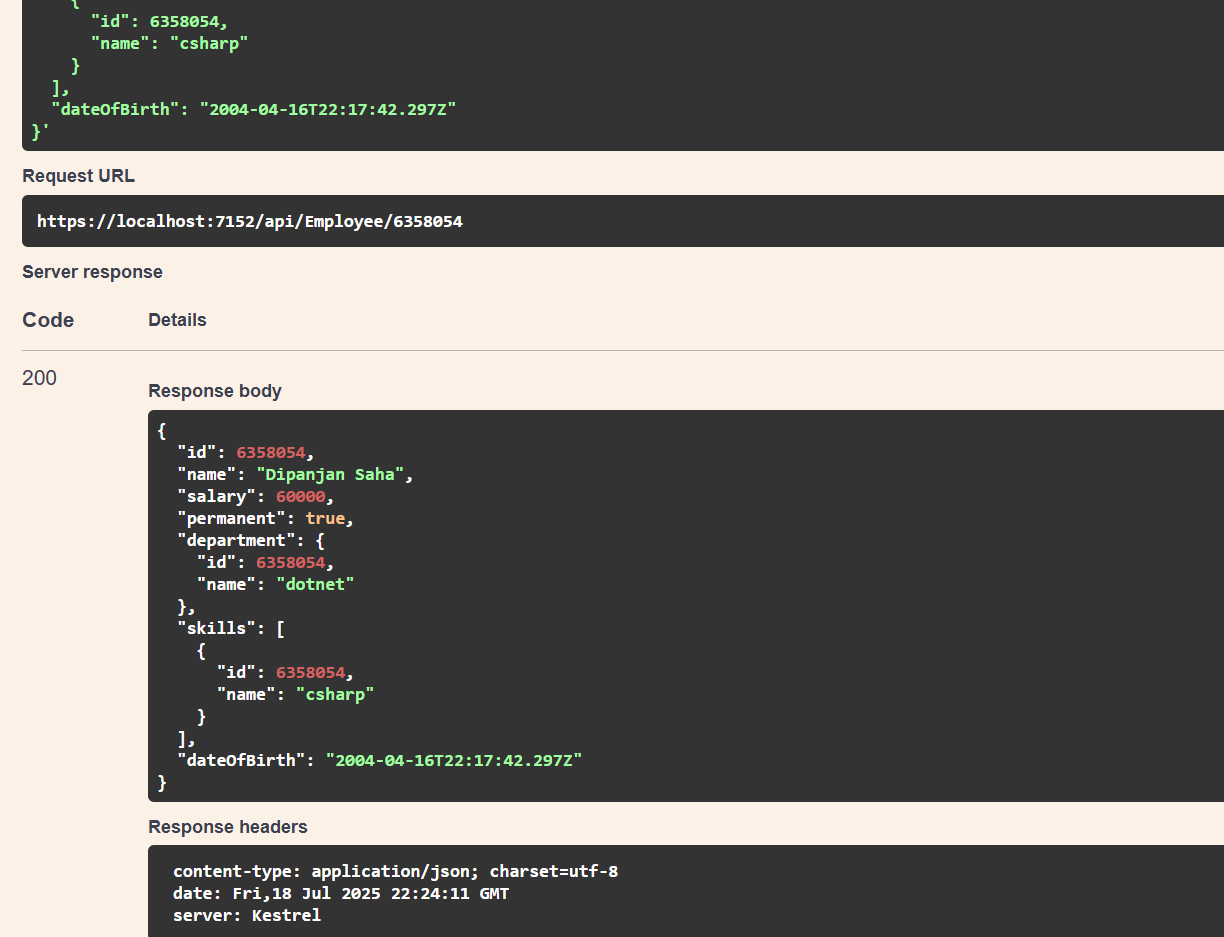




Week 4 crud operations



After editing the json updating the data:



Lab 5:

JsonWebToken

Program. Cs

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

// 1. Configure JWT Authentication

var securityKey = "mysuperdupersecretkeythatmeetsthelength"; // Same as the one used in token generator

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

builder.Services.AddAuthentication(options =>

{

options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = "mySystem",

ValidAudience = "myUsers",

IssuerSigningKey = symmetricSecurityKey

};

});

builder.Services.AddControllers();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI(); // <-- This generates the Swagger UI

}

// 2. Add Authentication Middleware

app.UseAuthentication(); // MUST be before UseAuthorization

app.UseAuthorization();

app.MapControllers();

app.Run();

Employee Controller with AUTHORIZATION

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using SwaggerDemoAPI.Filters;

using SwaggerDemoAPI.Models;

namespace SwaggerDemoAPI.Controllers

{

[ApiController]

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

[Authorize] // JWT token authentication

public class EmployeeController : ControllerBase

{

private static List<Employee> employees = new List<Employee>();

public EmployeeController()

{

if (employees.Count == 0)

employees = GetStandardEmployeeList();

}

[HttpGet]

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

[ProducesResponseType(500)]

public ActionResult<List<Employee>> GetStandard()

{

// Simulate an error to test exception filter

// throw new Exception("Test exception");

return Ok(employees);

}

[HttpPost]

public IActionResult AddEmployee([FromBody] Employee emp)

{

employees.Add(emp);

return Ok(emp);

}

[HttpPut("{id}")]

[ProducesResponseType(StatusCodes.Status200OK)]

[ProducesResponseType(StatusCodes.Status400BadRequest)]

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

{

if (id <= 0)

{

return BadRequest("Invalid employee id");

}

var employeeList = GetStandardEmployeeList();

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

if (employee == null)

{

return BadRequest("Invalid employee id");

}

// Update the employee fields

employee.Name = updatedEmployee.Name;

employee.Salary = updatedEmployee.Salary;

employee.Permanent = updatedEmployee.Permanent;

employee.Department = updatedEmployee.Department;

employee.Skills = updatedEmployee.Skills;

employee.DateOfBirth = updatedEmployee.DateOfBirth;

return Ok(employee); // Return updated employee

}

private List<Employee> GetStandardEmployeeList()

{

return new List<Employee>

{

new Employee

{

Id = 6358054,

Name = "Dipanjan Saha",

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(1995, 5, 12)

}

};

}

}

}

AuthController :

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 SwaggerDemoAPI.Controllers

{

[ApiController]

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

[AllowAnonymous] // Allows token generation without auth

public class AuthController : ControllerBase

{

[HttpGet("token")]

public IActionResult GetToken()

{

// Simulate user id and role

int userId = 1;

string userRole = "Admin";

var token = GenerateJSONWebToken(userId, userRole);

return Ok(new { Token = token });

}

private string GenerateJSONWebToken(int userId, string userRole)

{

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

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);

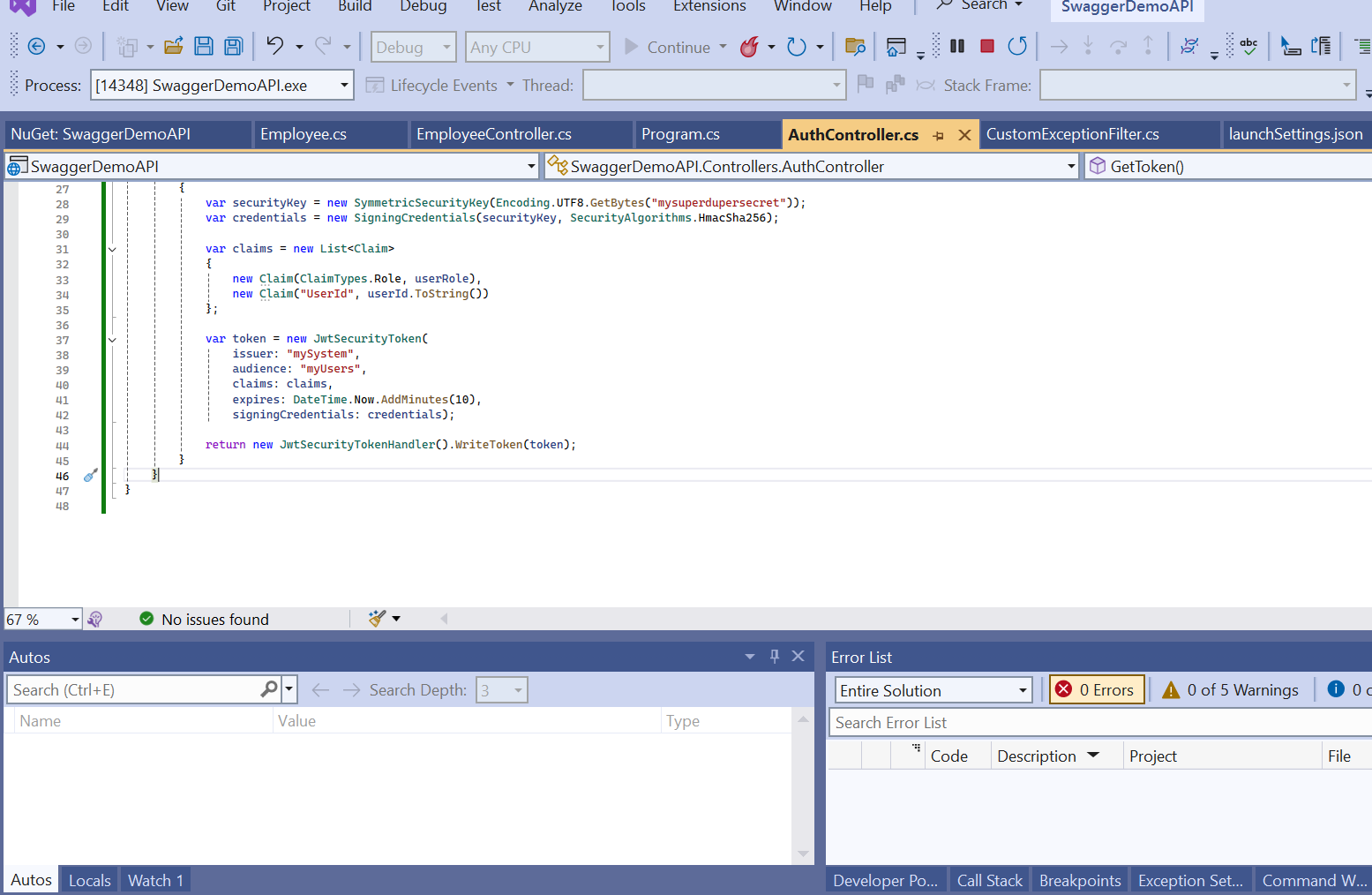
return new JwtSecurityTokenHandler().WriteToken(token);

}

}

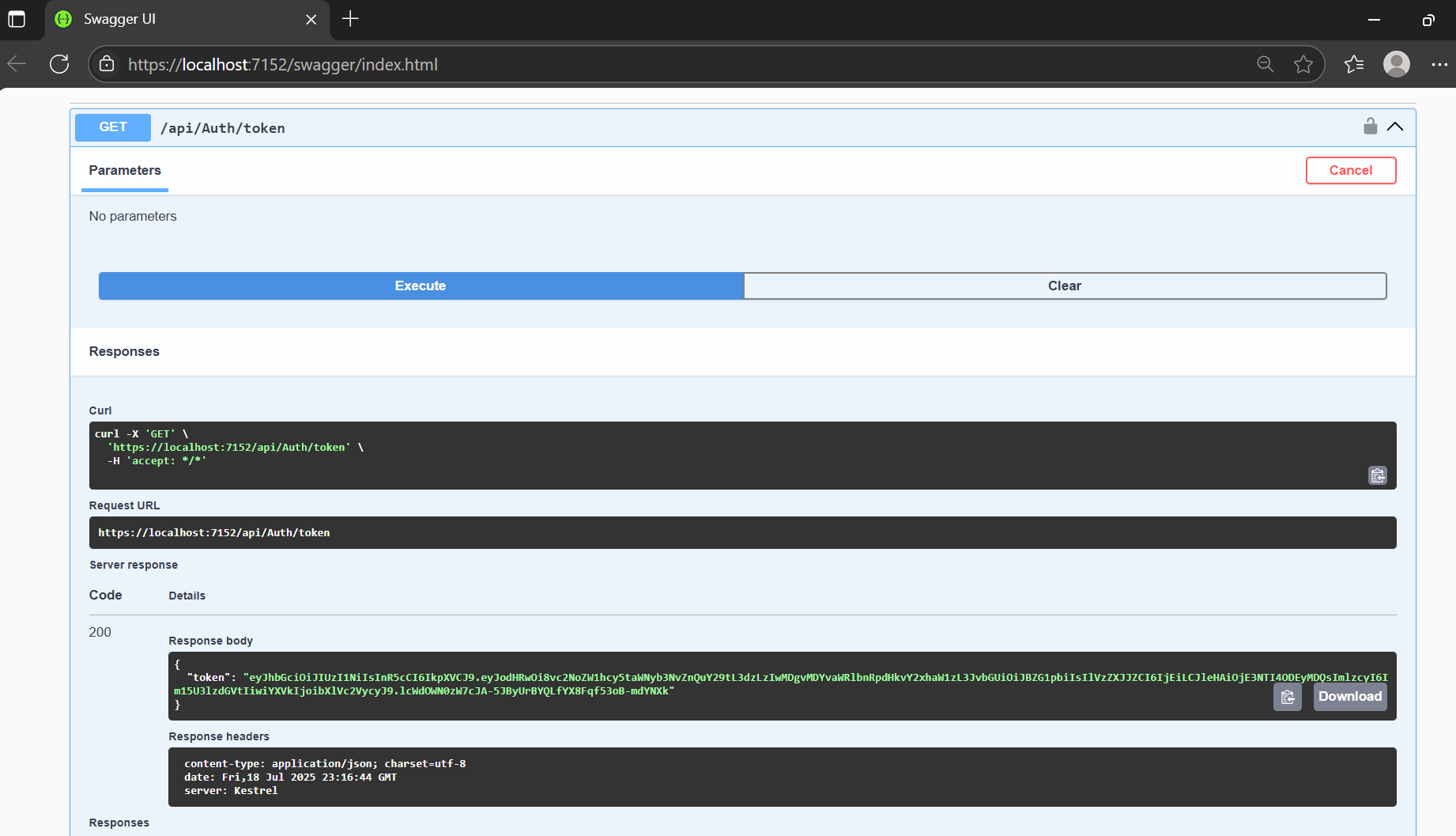
}

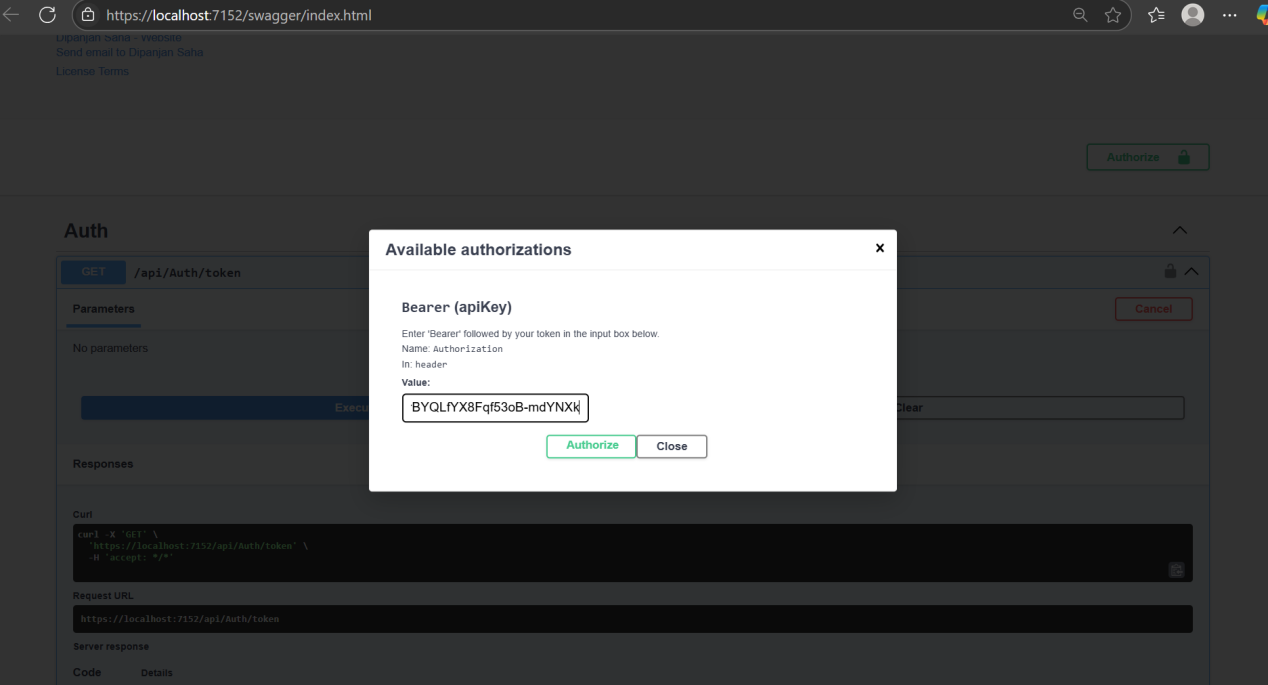
Code Execution



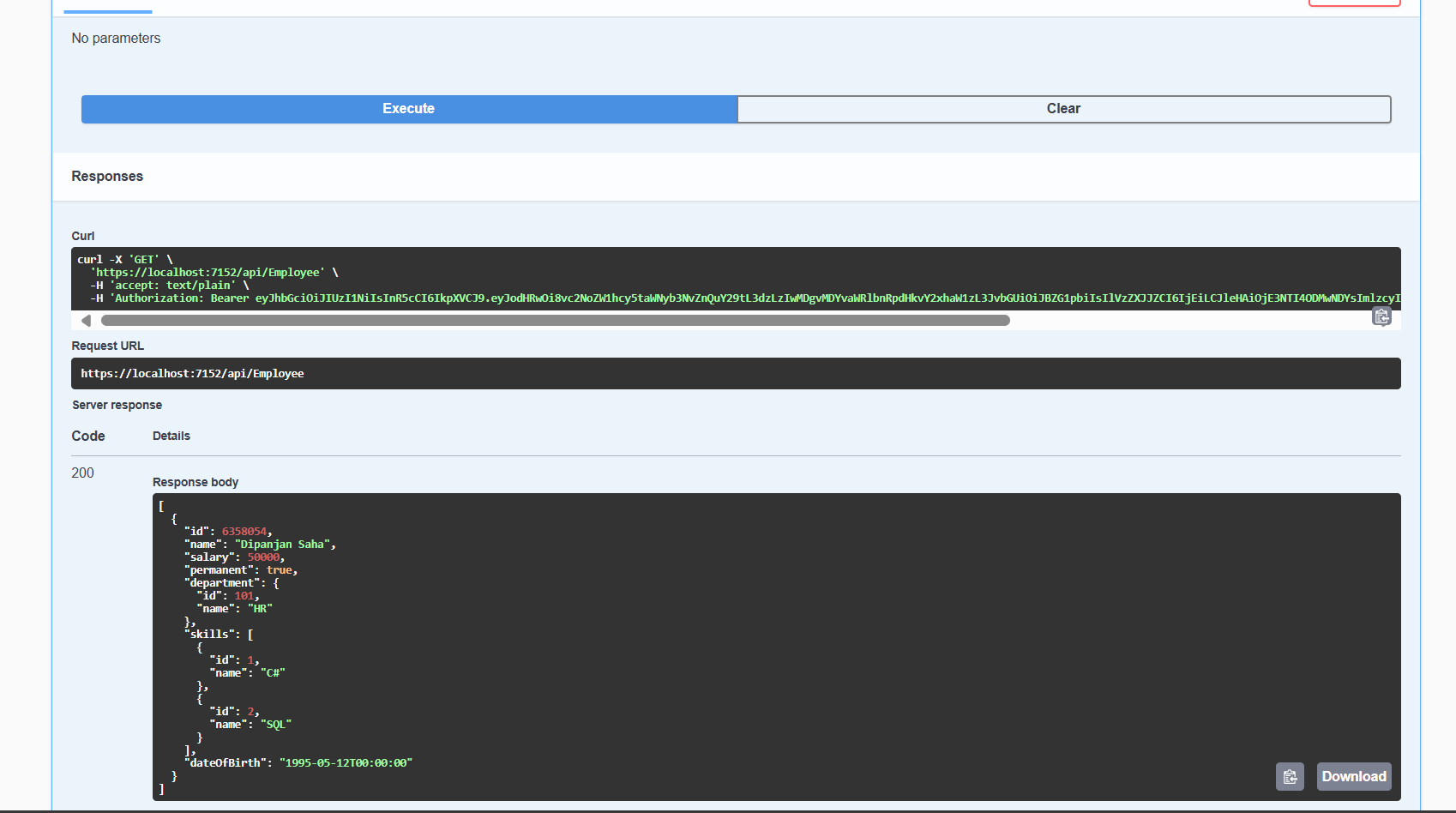
Swagger UI

Get th auth token



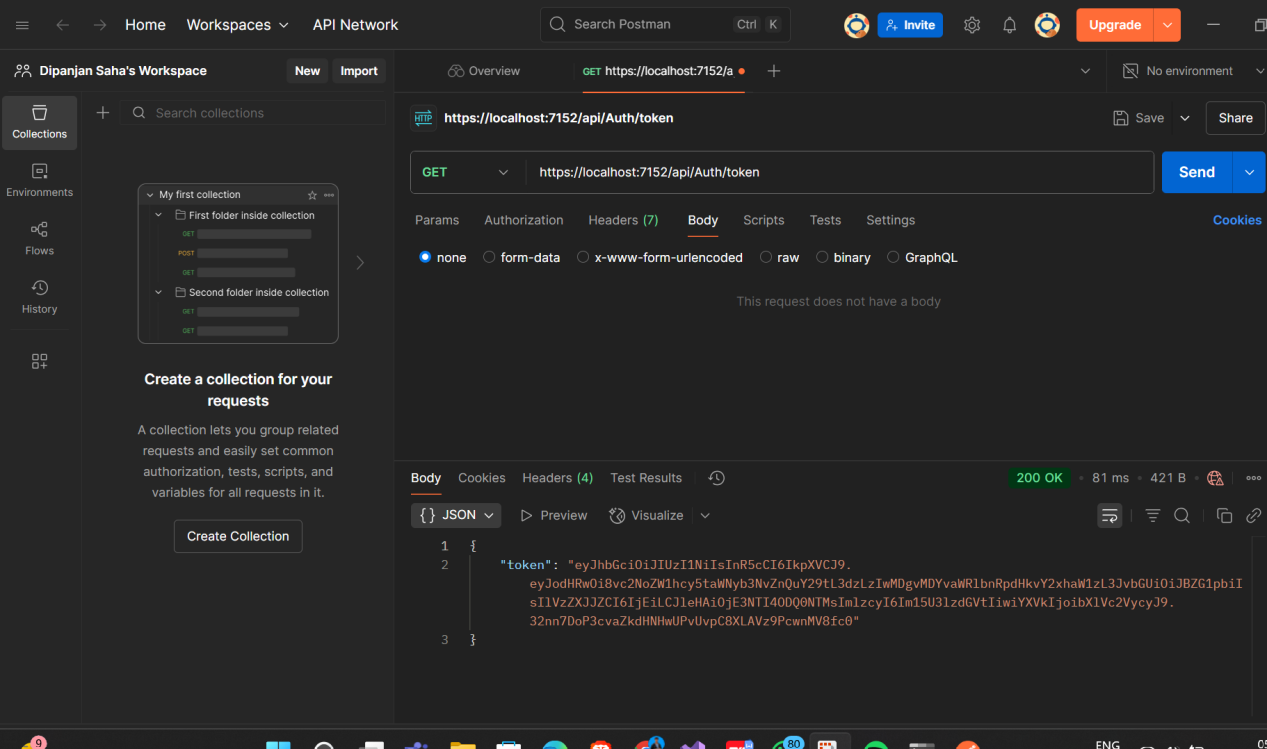


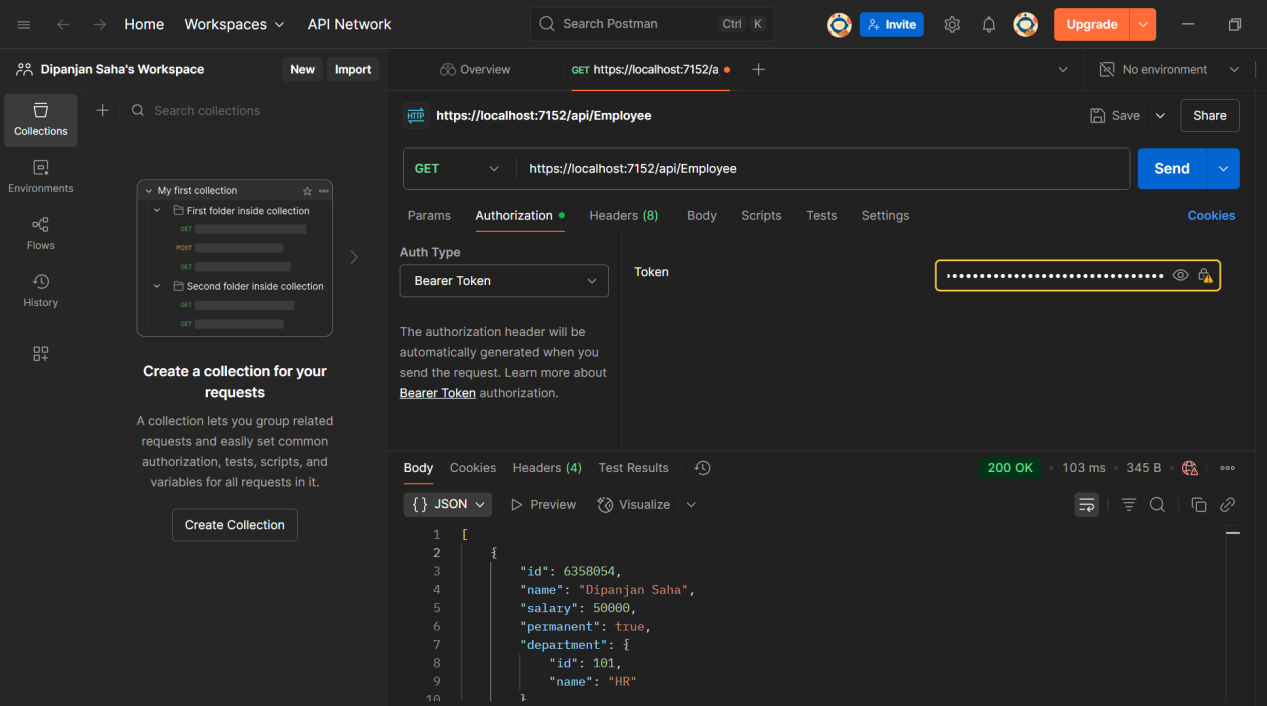
GET fetch :



Using postman

<https://localhost:7152/api/Auth/token>





Changed the Expiration time to 2 minutes

var token = new JwtSecurityToken(

issuer: "mySystem",

audience: "myUsers",

claims: claims,

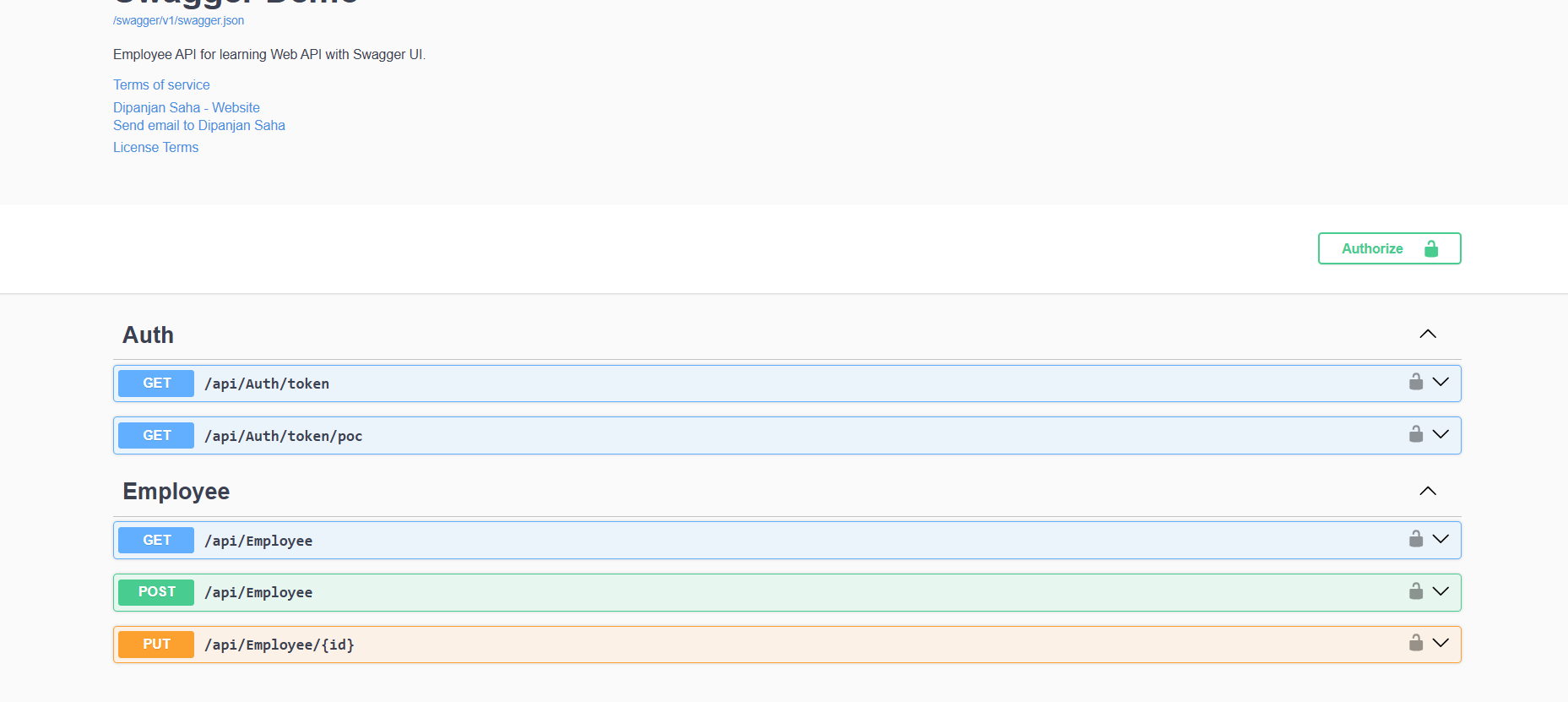
expires: DateTime.Now.AddMinutes(2),

signingCredentials: credentials);

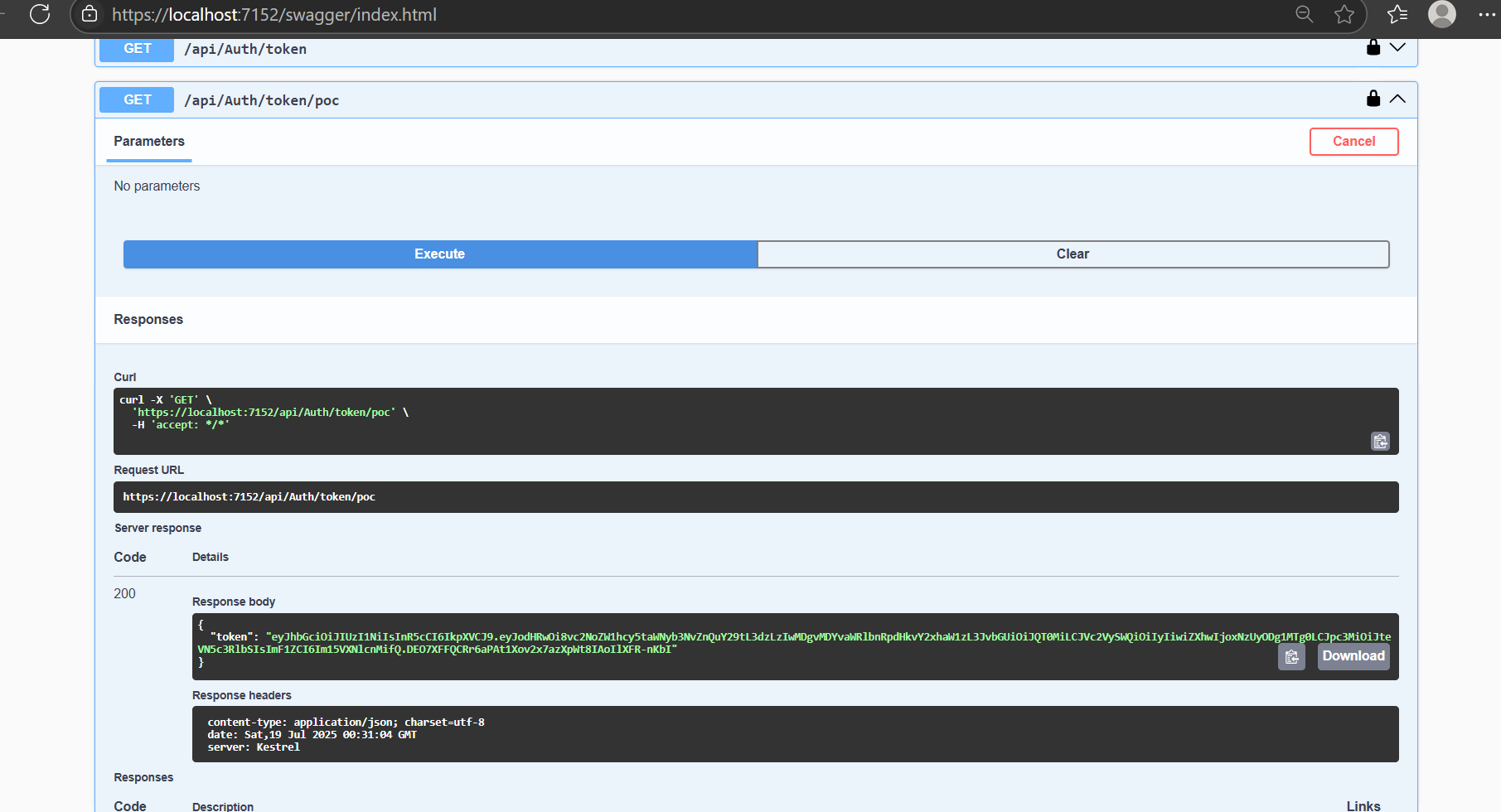
return new JwtSecurityTokenHandler().WriteToken(token);

Different Role Authorization

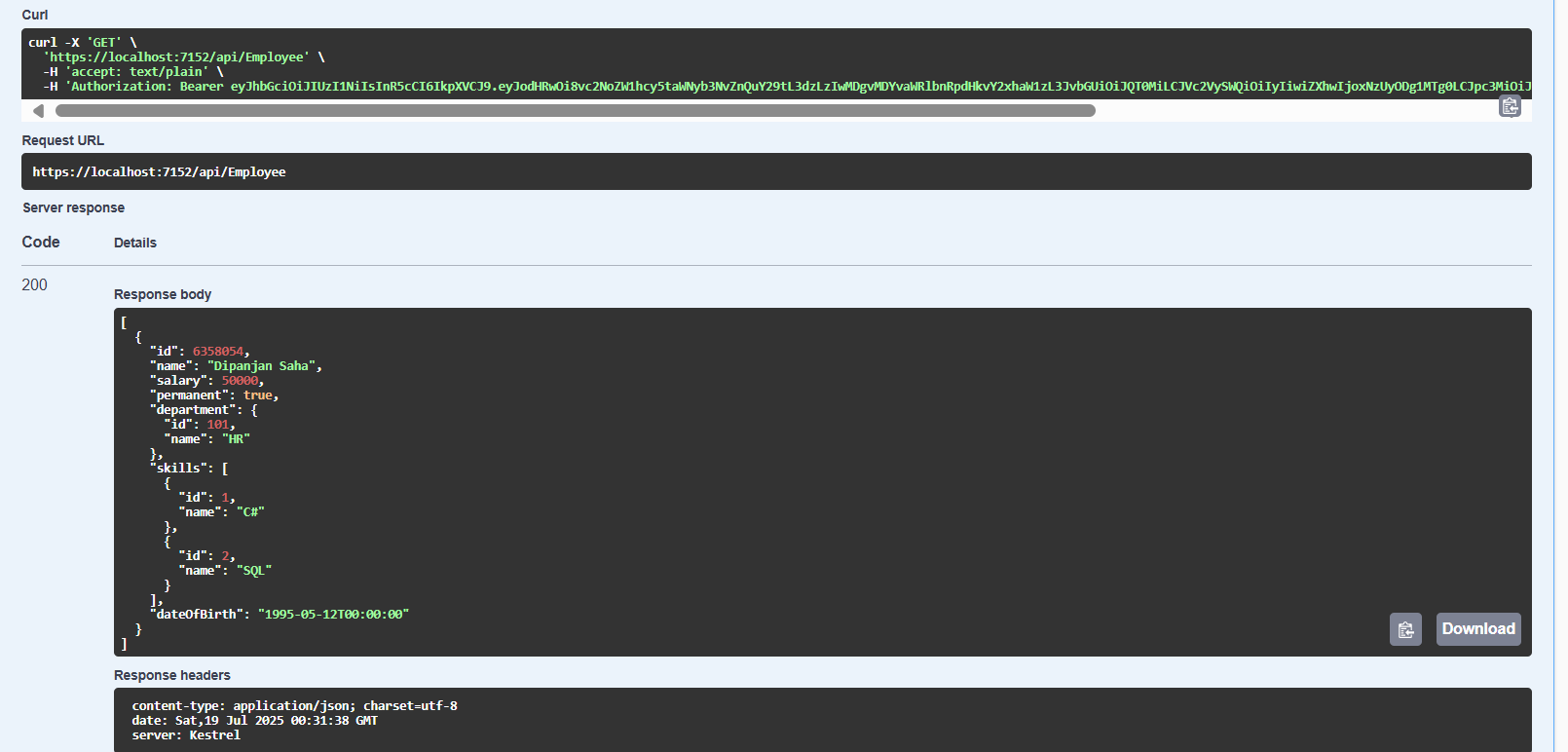
[Authorize(Roles = "POC,Admin")]



Token generation for bothe POC and admin role

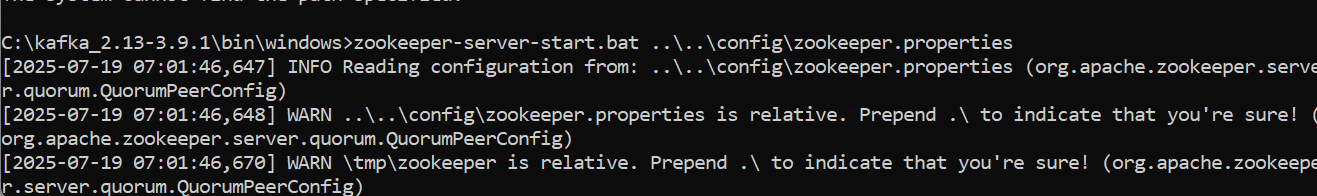


Able to fetch data from get call

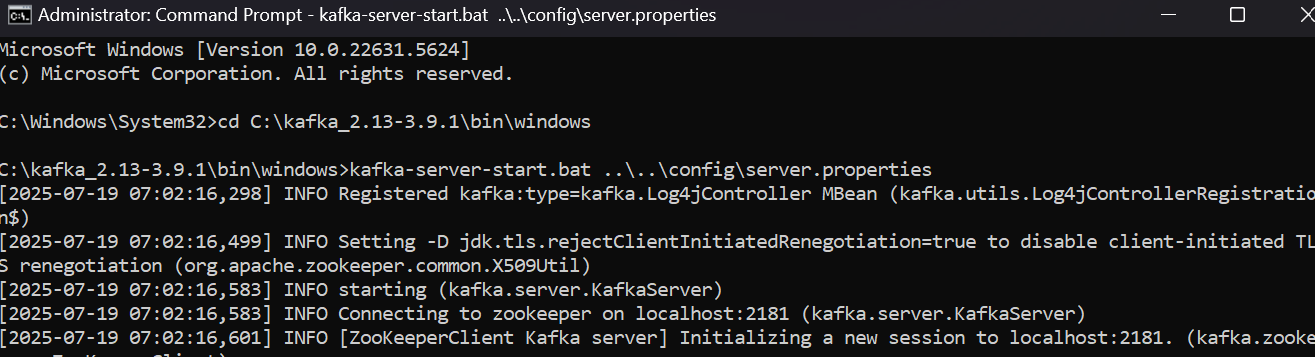


Lab 6:

Kafka: zookeeper



Kafka server start



Chat sender side

using System;

using Confluent.Kafka;

class Program

{

static async Task Main()

{

var config = new ProducerConfig

{

BootstrapServers = "localhost:9092"

};

using var producer = new ProducerBuilder<Null, string>(config).Build();

Console.WriteLine("=== Kafka Chat Producer ===");

Console.WriteLine("Type your message and press Enter. Type 'exit' to quit.");

while (true)

{

Console.Write("You: ");

var input = Console.ReadLine();

if (input.ToLower() == "exit")

break;

await producer.ProduceAsync("chat-topic", new Message<Null, string> { Value = input });

}

}

}

Chat reciever

class Program

{

static void Main()

{

var config = new ConsumerConfig

{

GroupId = "chat-group",

BootstrapServers = "localhost:9092",

AutoOffsetReset = AutoOffsetReset.Earliest

};

using var consumer = new ConsumerBuilder<Ignore, string>(config).Build();

consumer.Subscribe("chat-topic");

Console.WriteLine("=== Kafka Chat Consumer ===");

Console.WriteLine("Waiting for messages...");

while (true)

{

var cr = consumer.Consume();

Console.WriteLine($"\nFriend: {cr.Message.Value}");

}

}

}

