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

**TASK 1 :-**

Objective

This task focuses on understanding the foundational concepts of RESTful Web Services, Web API, and microservices. It also introduces the HTTP communication model, configuration structure in .NET Core, and building a basic Web API.

1.1 What is a RESTful Web Service?

REST (Representational State Transfer) is an architectural style that defines a set of constraints to be used when creating web services. RESTful services use standard HTTP protocols for communication and treat resources as addressable entities identified by URIs.

Key Features of REST:

Stateless: Each request from the client contains all the information needed to process the request. The server does not maintain any client context.

Resource-Based: Resources are accessed via standard URIs (e.g., /api/products).

Standard HTTP Methods: HTTP verbs like GET, POST, PUT, and DELETE are used to perform operations on resources.

Uniform Interface: A standardized way to access and interact with resources.

Multiple Formats: Unlike older SOAP-based services, REST supports multiple data formats (JSON, XML, etc.), with JSON being the most widely used.

1.2 Web Service vs Web API vs Microservices

| **Feature** | **Web Service (SOAP)** | **Web API (REST)** | **Microservices** |
| --- | --- | --- | --- |
| Protocol | SOAP | HTTP | HTTP, AMQP, etc. |
| Format | XML | JSON/XML | JSON, Protocol Buffers, etc. |
| Platform | Platform-dependent | Platform-independent | Platform-independent |
| Deployment Unit | Single | Usually single | Independent, deployable services |
| Focus | Reusability | Lightweight interfaces | Domain-driven, loosely coupled services |

Microservices are a design pattern where applications are composed of small, independent services that communicate over HTTP or messaging queues. Each service represents a single business capability.

1.3 What is HttpRequest and HttpResponse?

HttpRequest: Represents the incoming HTTP request. It contains details such as URL, headers, body, and query parameters.

HttpResponse: Represents the response sent back to the client. It includes the status code, headers, and response body.

1.4 Common HTTP Verbs (Action Methods)

| **HTTP Verb** | **Usage** | **Attribute in Web API** |
| --- | --- | --- |
| GET | Retrieve data | [HttpGet] |
| POST | Create new data | [HttpPost] |
| PUT | Update data | [HttpPut] |
| DELETE | Remove data | [HttpDelete] |

These verbs correspond to the CRUD operations on data.

1.5 Common HTTP Status Codes

| **Code** | **Meaning** | **Usage in Web API** |
| --- | --- | --- |
| 200 | OK | Request successful |
| 400 | Bad Request | Invalid client input |
| 401 | Unauthorized | Auth token missing/invalid |
| 500 | Internal Server Error | Server-side issue |

**FirstWebApi.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.MapGet("/weatherforecast", () =>

{

    var summaries = new[]

    {

        "Freezing", "Bracing", "Chilly", "Cool", "Mild",

        "Warm", "Balmy", "Hot", "Sweltering", "Scorching"

    };

    var forecast = Enumerable.Range(1, 5).Select(index =>

        new WeatherForecast

        (

            DateOnly.FromDateTime(DateTime.Now.AddDays(index)),

            Random.Shared.Next(-20, 55),

            summaries[Random.Shared.Next(summaries.Length)]

        ))

        .ToArray();

    return forecast;

})

.WithName("GetWeatherForecast");

app.Run();

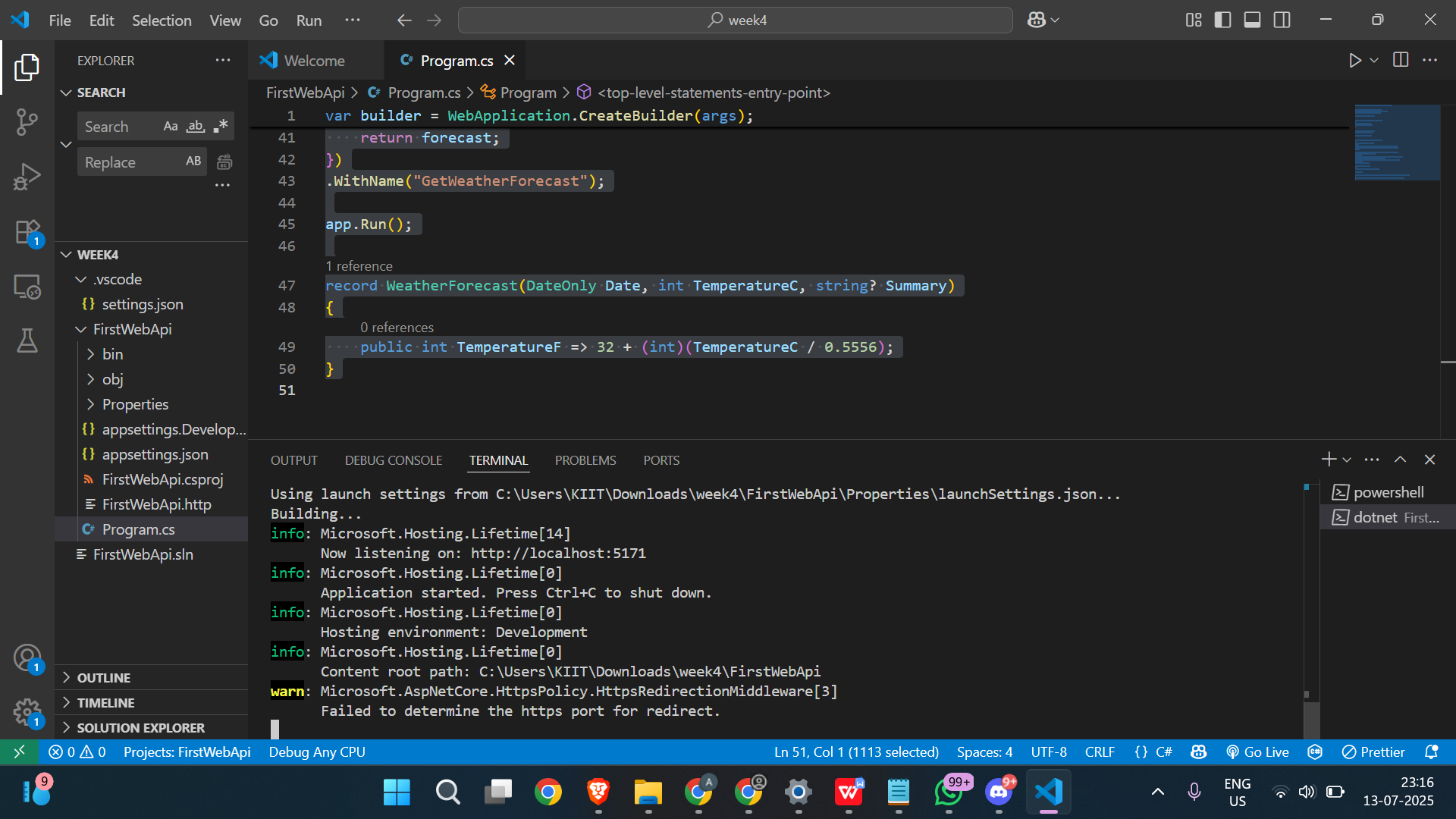
record WeatherForecast(DateOnly Date, int TemperatureC, string? Summary)

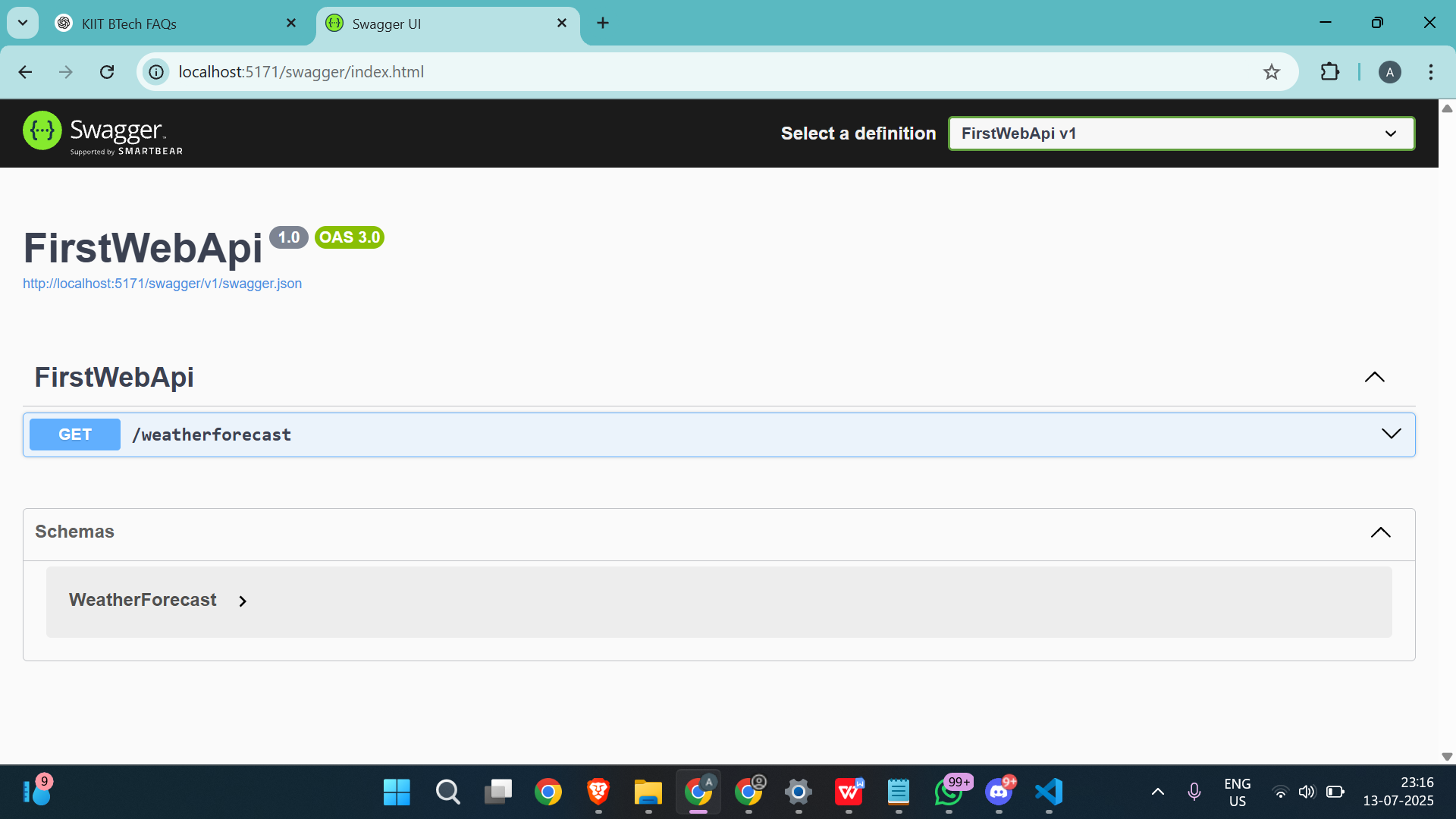
{

    public int TemperatureF => 32 + (int)(TemperatureC / 0.5556);

}

**SCREENSHOTS**(task 1):

-



**TASK 2 :-**

2.1 What is Swagger?

Swagger (also known as OpenAPI) is an open-source tool that allows automatic generation of interactive API documentation. It simplifies development by providing a UI where APIs can be tested directly in the browser.

Key Features:

Lists all available API endpoints.

Allows testing of endpoints via UI.

Displays HTTP response codes, parameter types, and response bodies.

**Program.cs**

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

    c.SwaggerDoc("v1", new Microsoft.OpenApi.Models.OpenApiInfo

    {

        Title = "Swagger Demo",

        Version = "v1",

        Description = "TBD",

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

        Contact = new Microsoft.OpenApi.Models.OpenApiContact

        {

            Name = "John Doe",

            Email = "john@xyzmail.com",

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

        },

        License = new Microsoft.OpenApi.Models.OpenApiLicense

        {

            Name = "License Terms",

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

        }

    });

});

var app = builder.Build();

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();

**EmployeeController.cs**

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

namespace FirstWebApi.Controllers

{

    [ApiController]

    [Route("emp")]

    public class EmployeeController : ControllerBase

    {

        private static readonly List<Employee> employees = new List<Employee>

        {

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

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

            new Employee { Id = 3, Name = "Charlie", Department = "Finance" }

        };

        [HttpGet]

        public IActionResult Get()

        {

            return Ok(employees);

        }

    }

    public class Employee

    {

        public int Id { get; set; }

        public string Name { get; set; }

        public string Department { get; set; }

    }

}

**Valusecontroller.cs**

using Microsoft.AspNetCore.Mvc;

namespace FirstWebApi.Controllers

{

    [ApiController]

    [Route("[controller]")]

    public class ValuesController : ControllerBase

    {

        [HttpGet]

        public IActionResult Get()

        {

            return Ok(new string[] { "value1", "value2" });

        }

        [HttpPost]

        public IActionResult Post([FromBody] string value)

        {

            return Ok($"Received: {value}");

        }

        [HttpPut("{id}")]

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

        {

            return Ok($"Updated ID {id} with {value}");

        }

        [HttpDelete("{id}")]

        public IActionResult Delete(int id)

        {

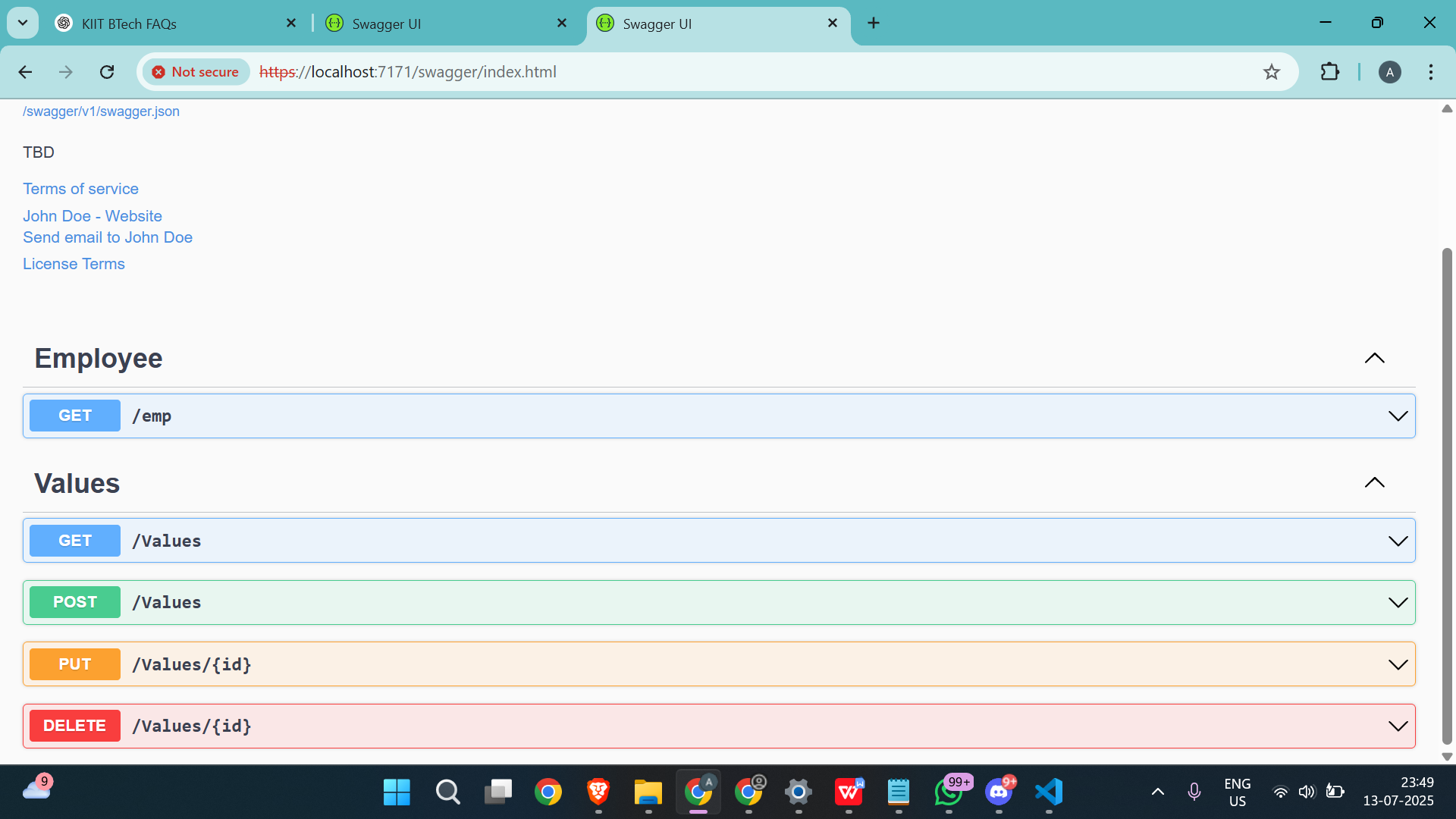
            return Ok($"Deleted ID {id}");

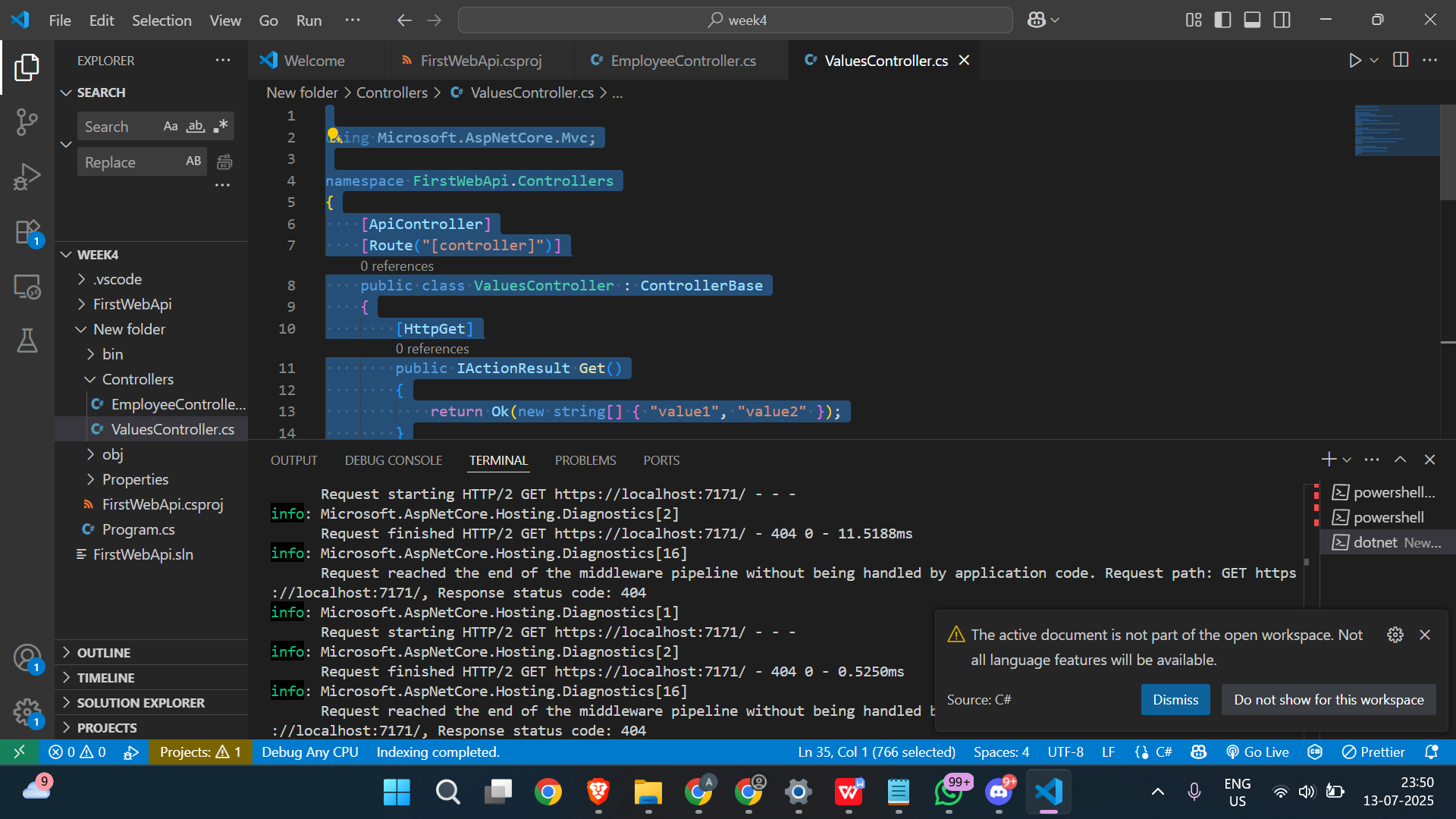
        }

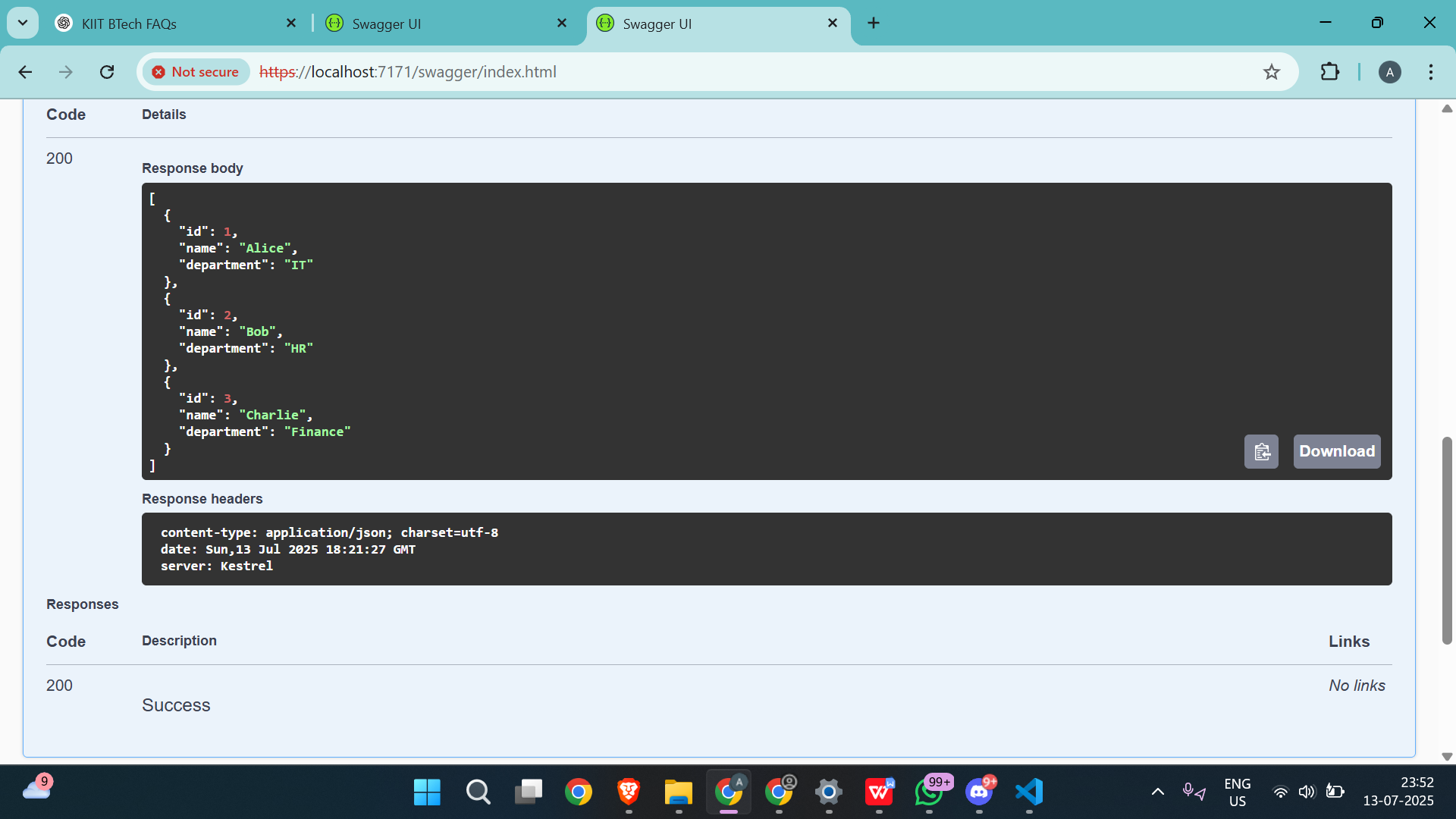
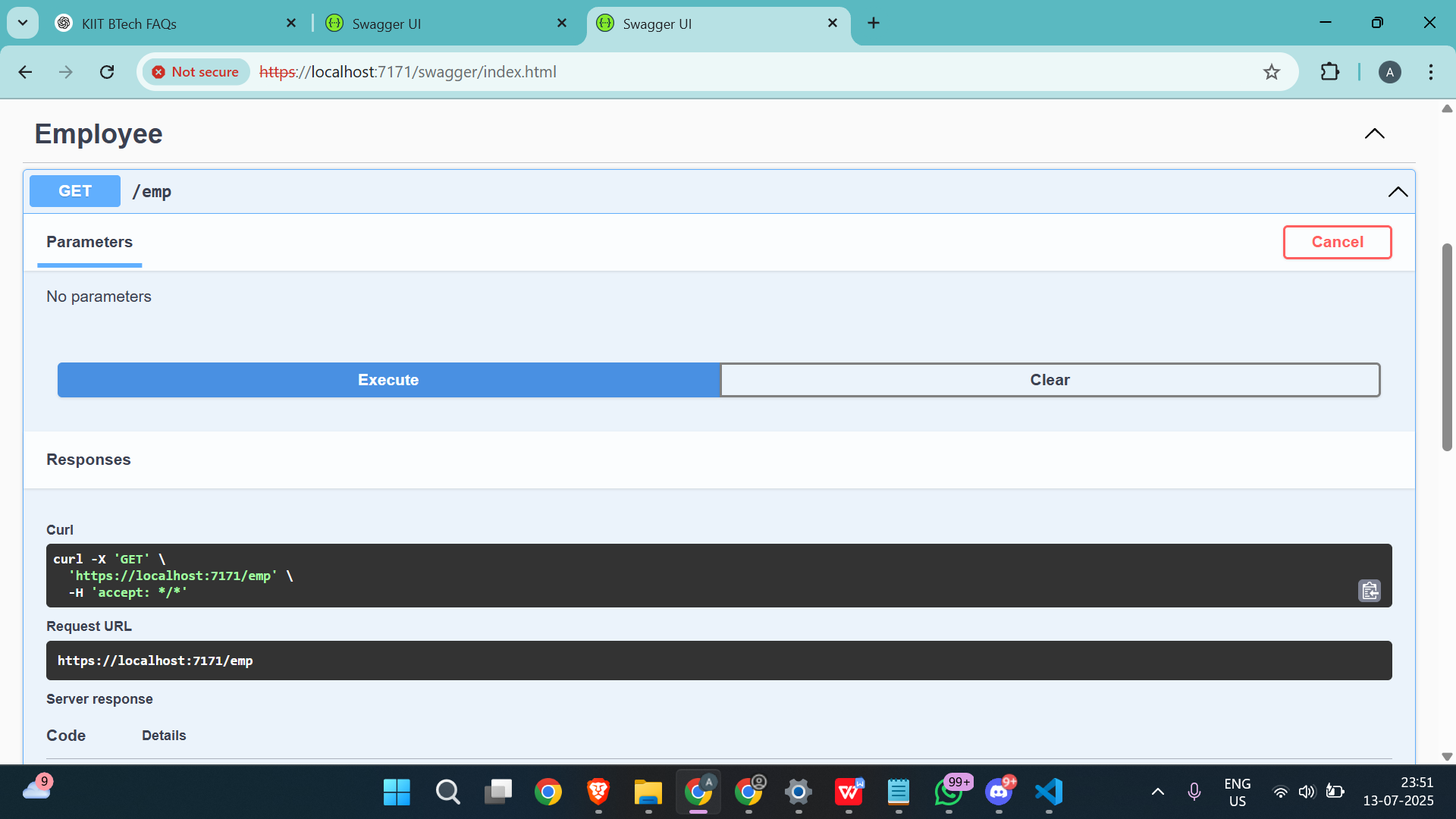
    }

}

**SCREENSHOTS**(task 2):-







**TASK 3 :-**

**Program.cs**

using EmployeeApi.Filters;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container

builder.Services.AddControllers(options =>

{

    options.Filters.Add<CustomExceptionFilter>();

});

// 🔹 Add Swagger

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// 🔹 Enable Swagger

if (app.Environment.IsDevelopment() || app.Environment.IsProduction())

{

    app.UseSwagger();

    app.UseSwaggerUI();

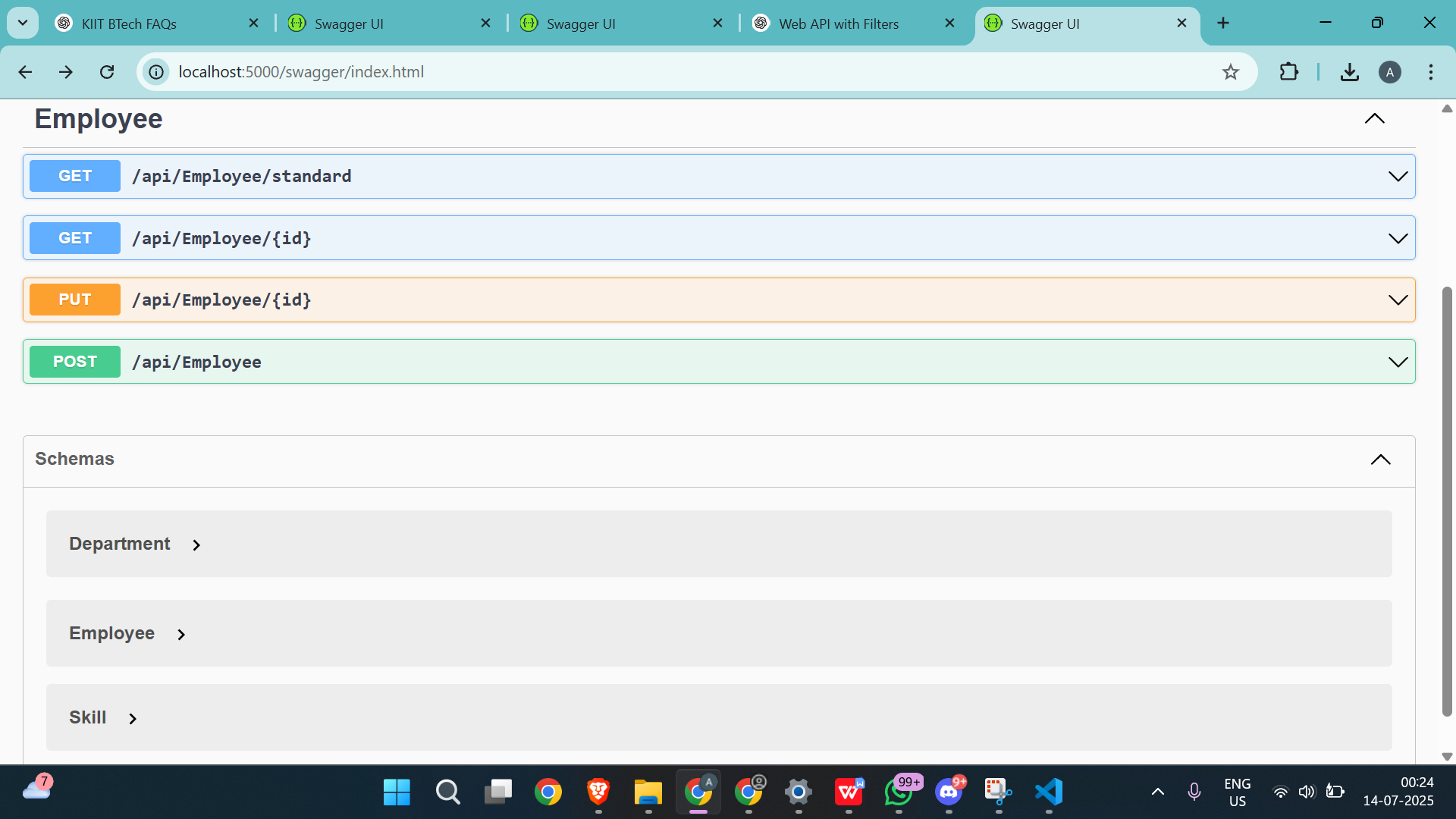
}

app.UseAuthorization();

app.MapControllers();

app.Run();

**SCREENSHOTS**(task 3):-



**TASK 4 :-**

**Program.cs**

using EmployeeApi.Filters;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container

builder.Services.AddControllers(options =>

{

    options.Filters.Add<CustomExceptionFilter>();

});

// 🔹 Add Swagger

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// 🔹 Enable Swagger

if (app.Environment.IsDevelopment() || app.Environment.IsProduction())

{

    app.UseSwagger();

    app.UseSwaggerUI();

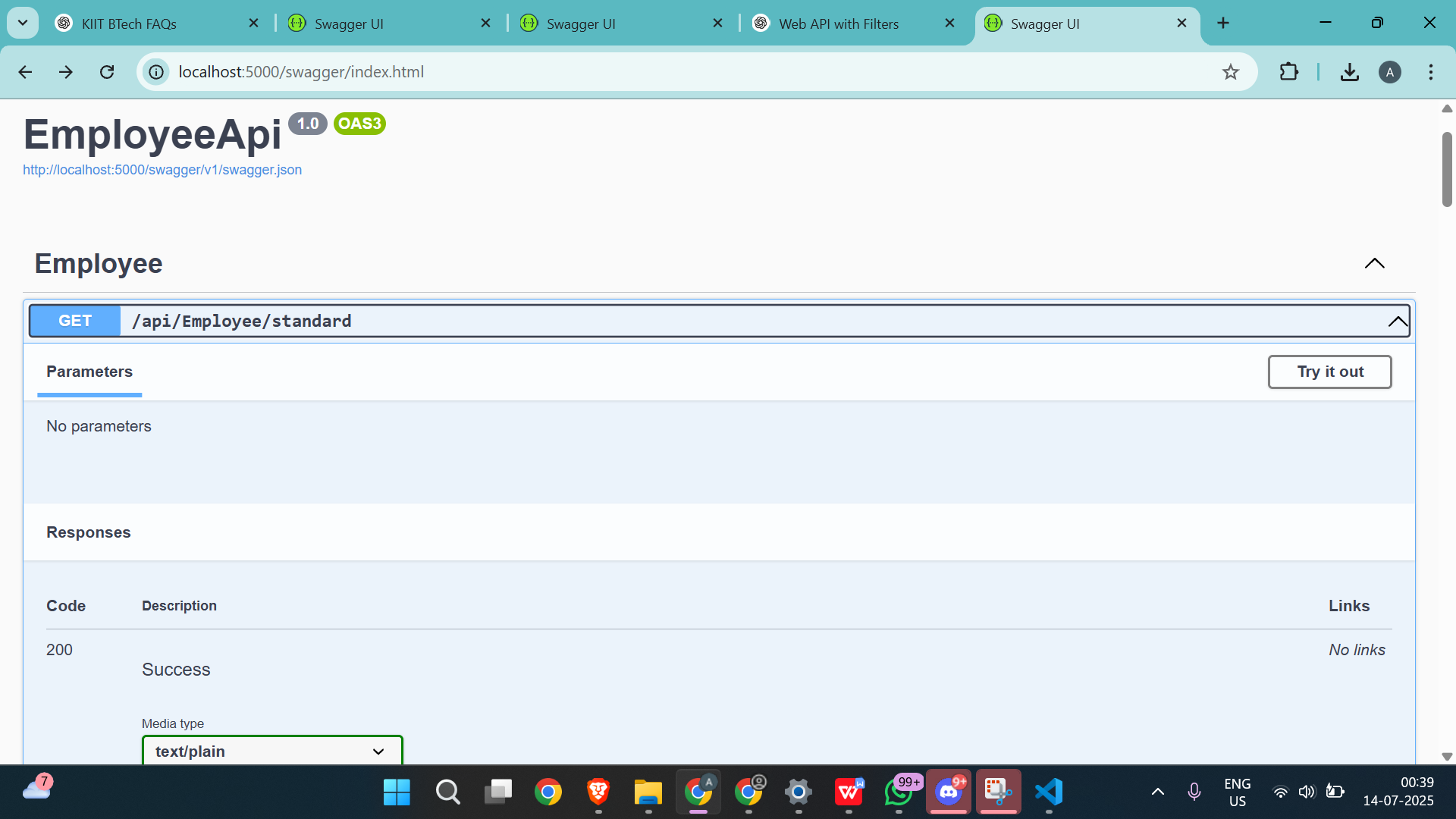
}

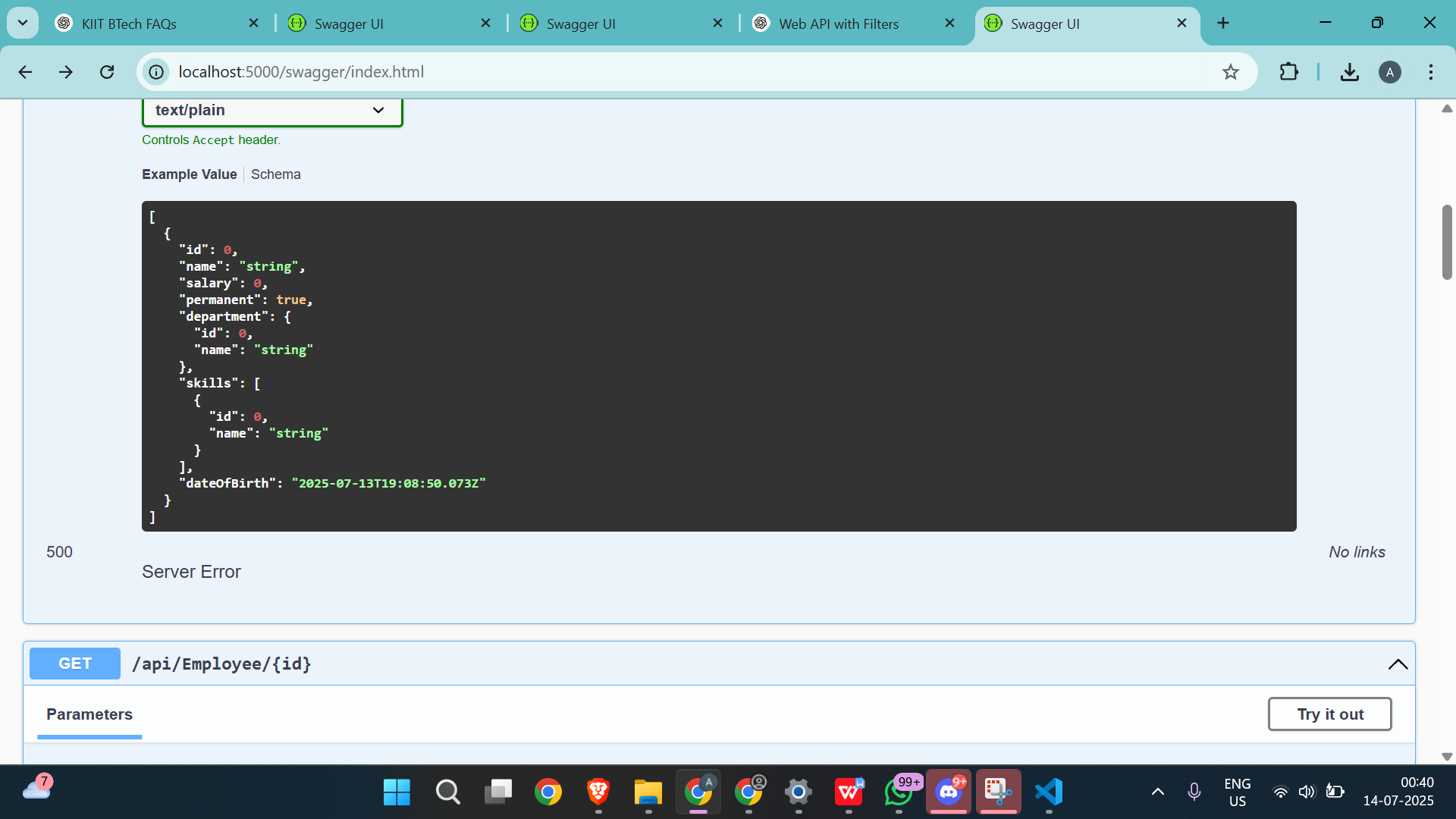
app.UseAuthorization();

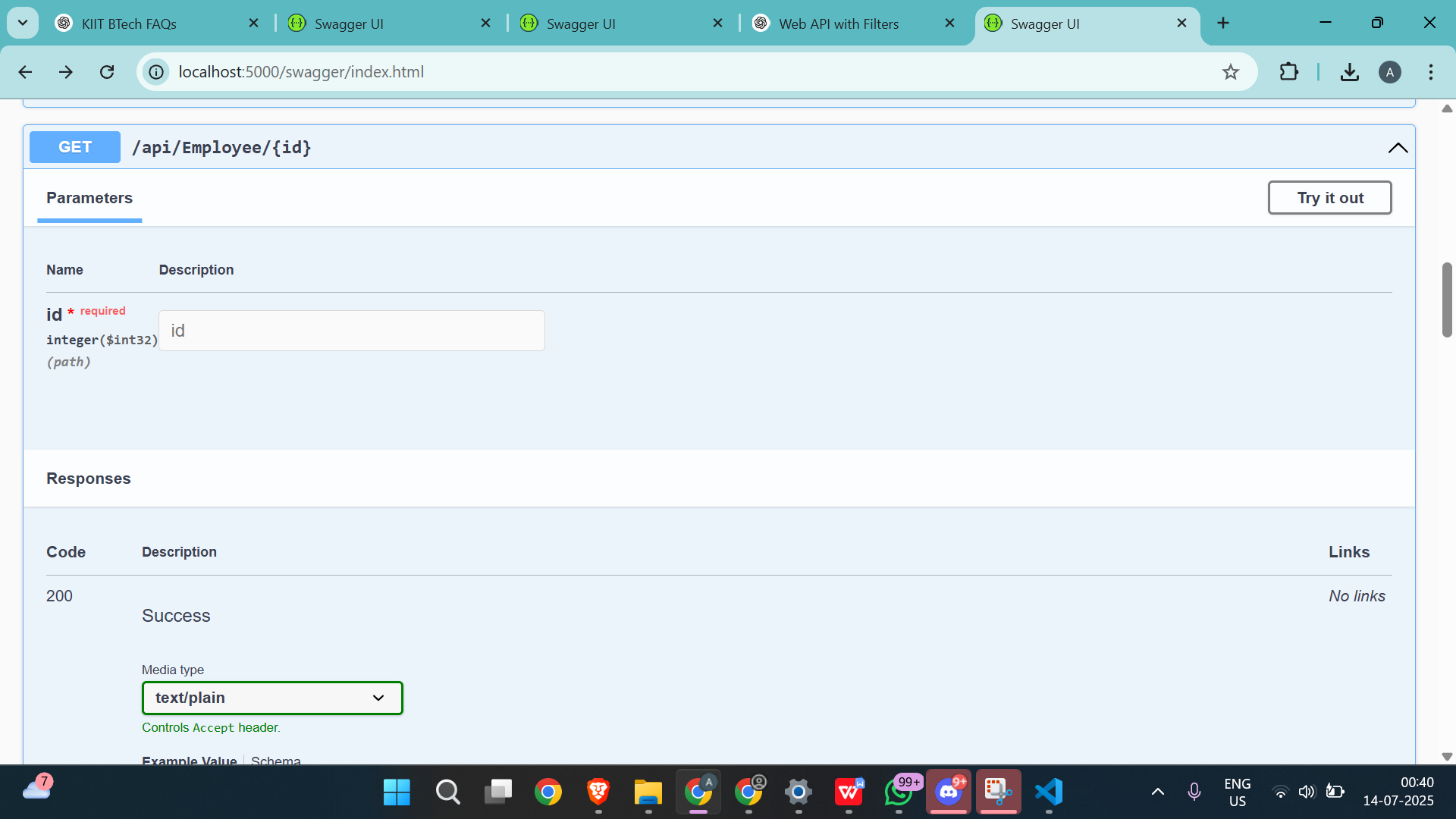
app.MapControllers();

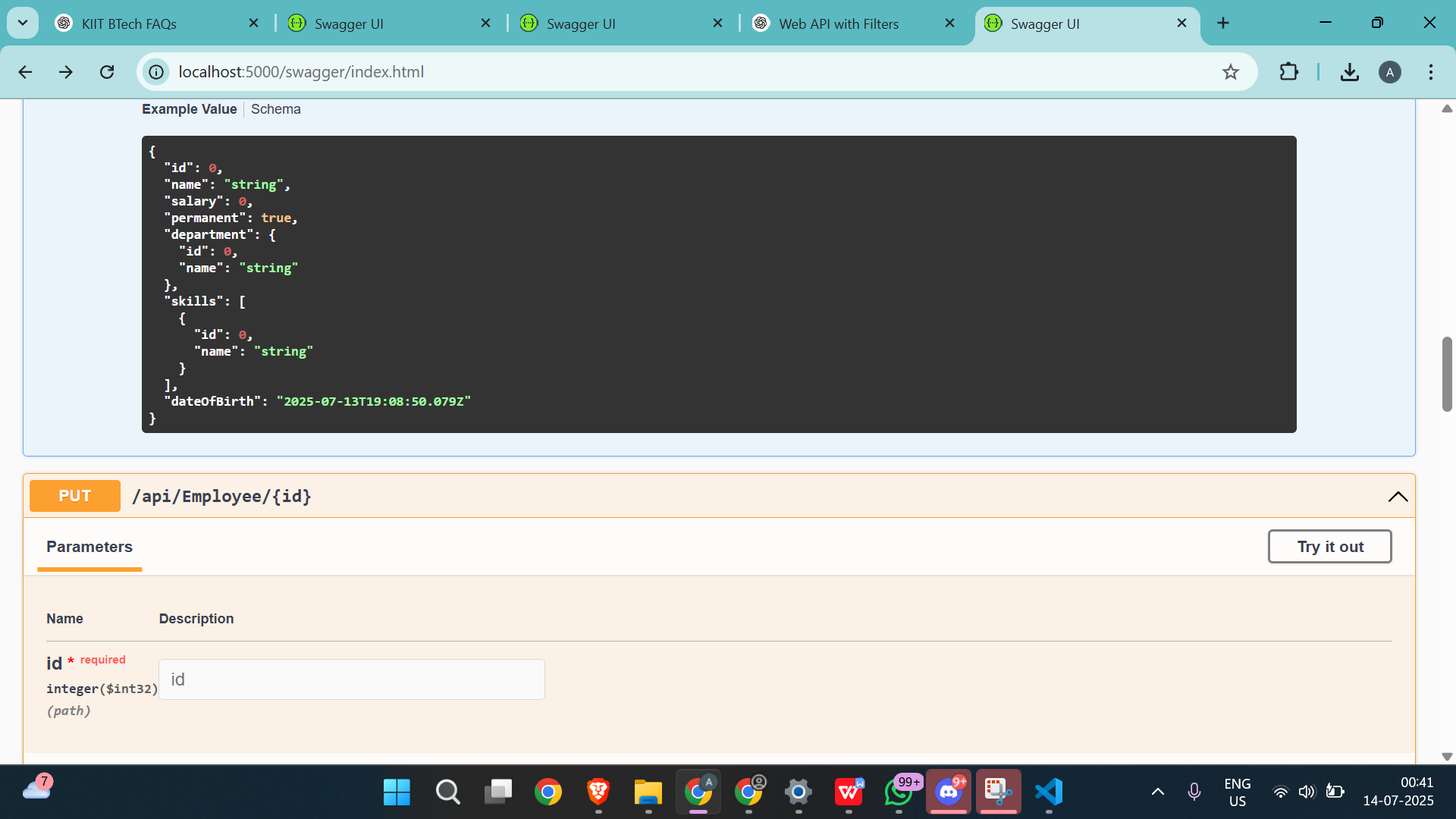
app.Run();

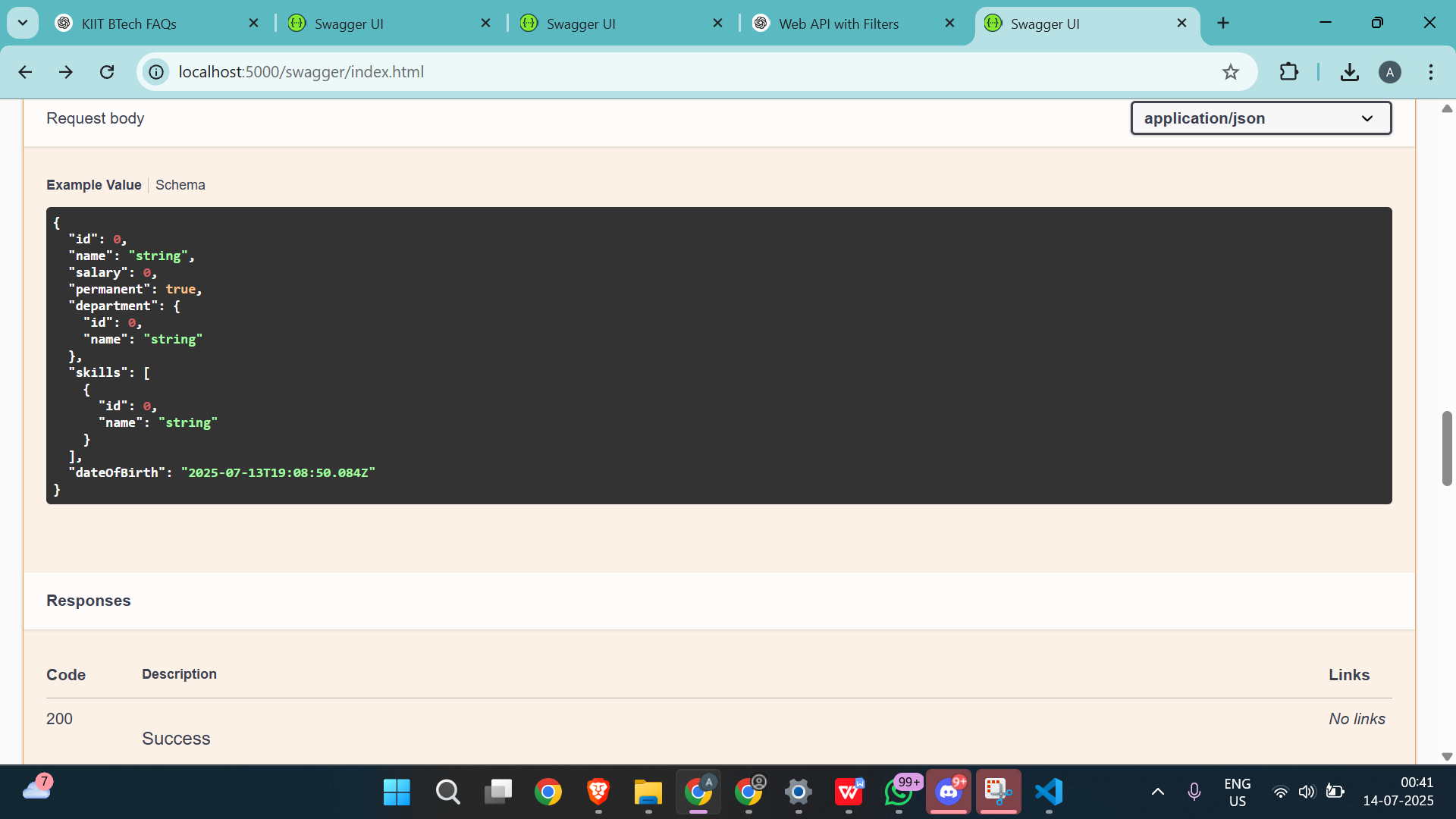
**SCREENSHOTS**(task 4):-

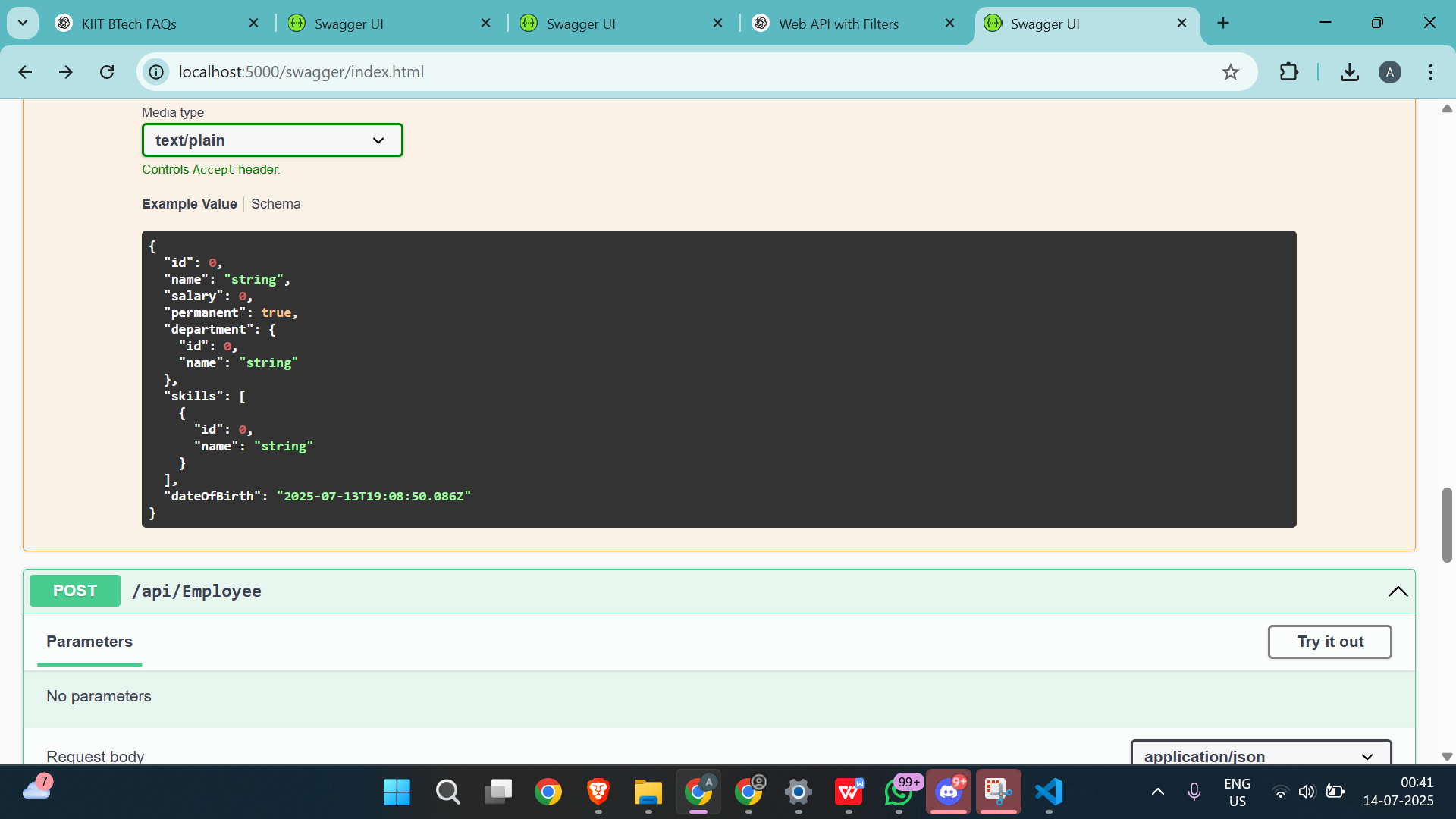


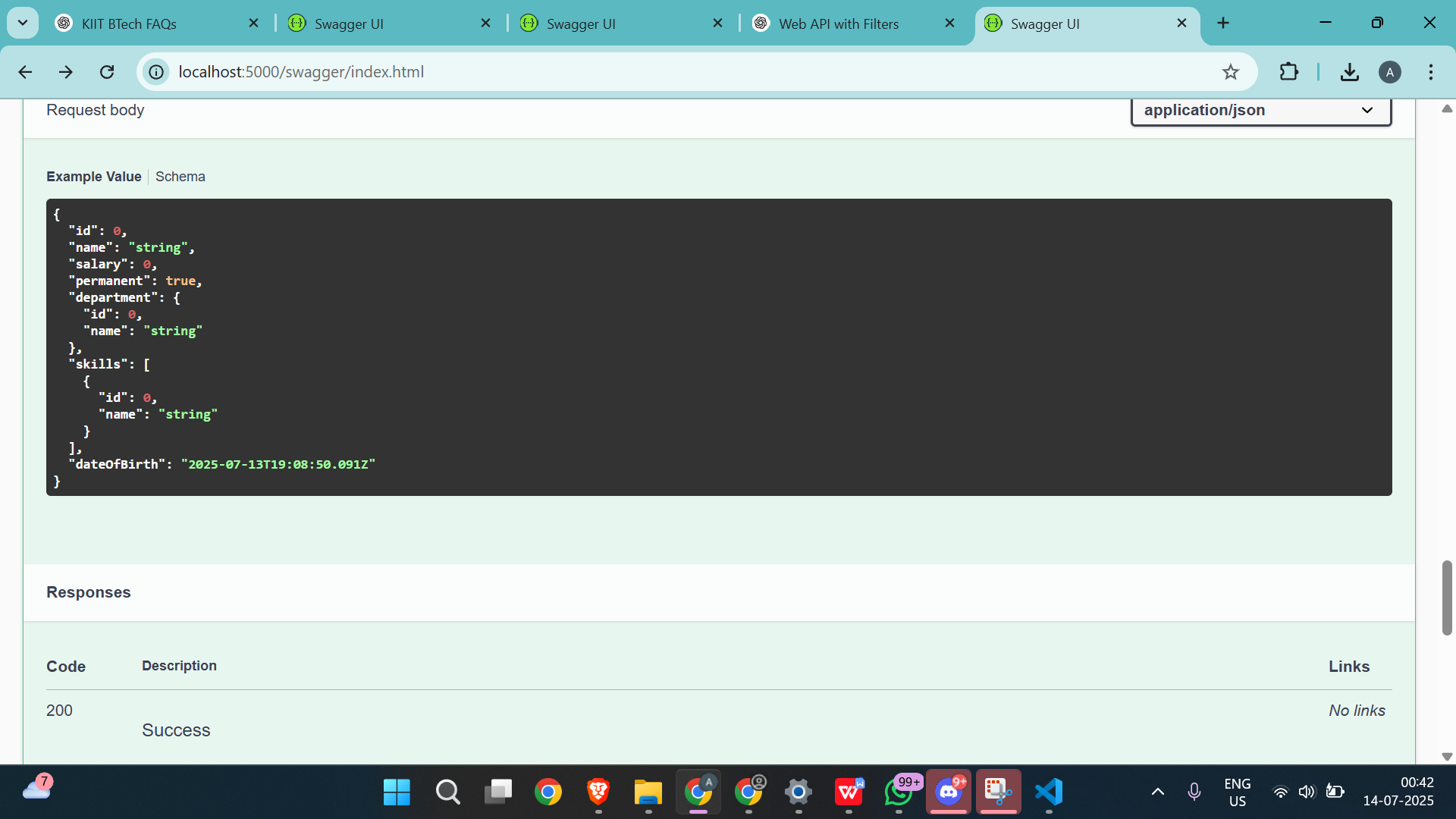


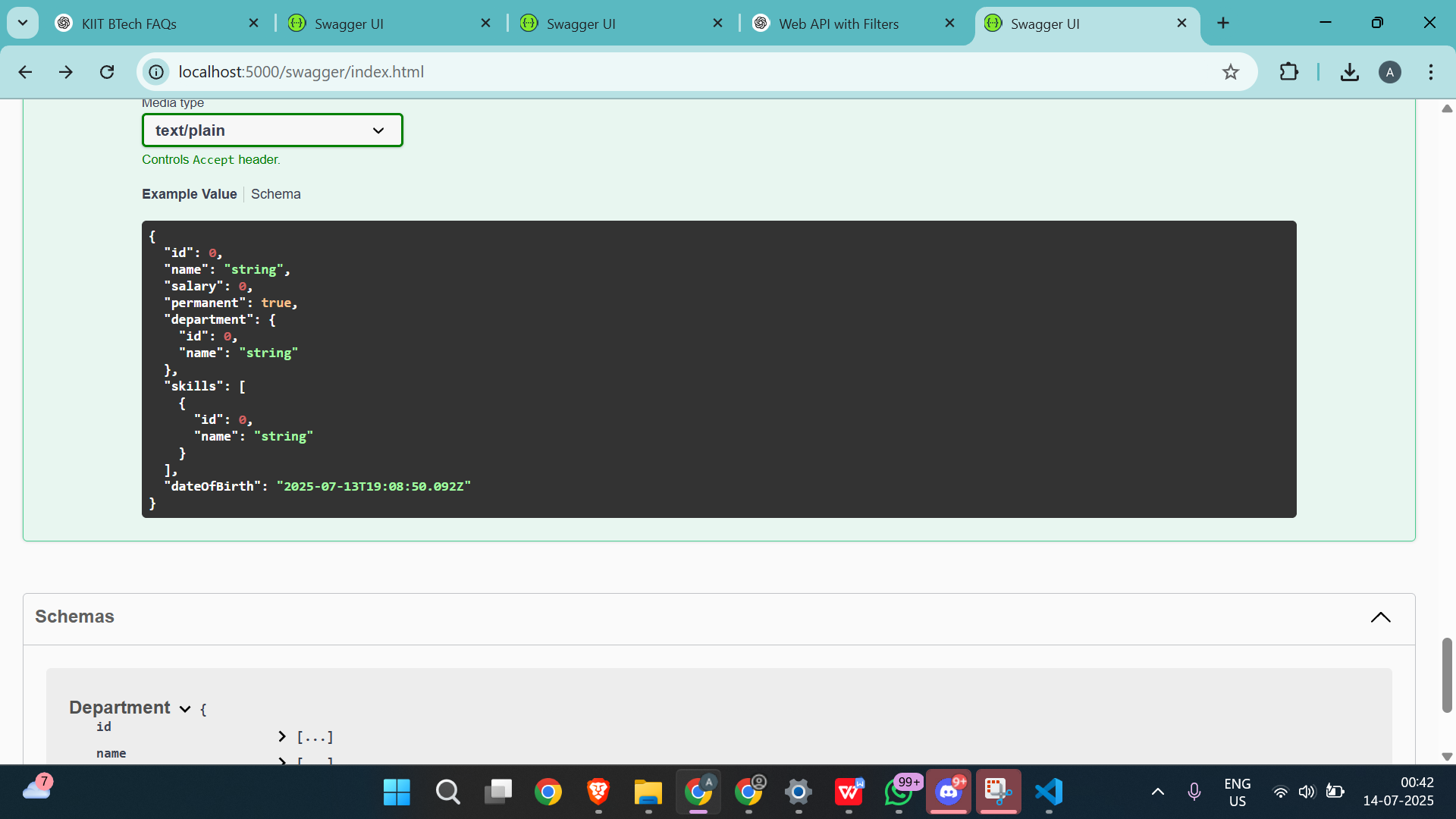












**TASK 5 :-**

**Program.cs**

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Hosting;

namespace JwtCorsApi

{

    public class Program

    {

        public static void Main(string[] args) =>

            CreateHostBuilder(args).Build().Run();

        public static IHostBuilder CreateHostBuilder(string[] args) =>

            Host.CreateDefaultBuilder(args)

                .ConfigureWebHostDefaults(webBuilder =>

                {

                    webBuilder.UseStartup<Startup>();

                });

    }

}

**Startup.cs**

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

namespace JwtCorsApi

{

    public class Startup

    {

        public Startup(IConfiguration configuration)

        {

            Configuration = configuration;

        }

        public IConfiguration Configuration { get; }

        public void ConfigureServices(IServiceCollection services)

        {

            services.AddControllers();

            string securityKey = "mysuperdupersecret";

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

            services.AddAuthentication(x =>

            {

                x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

                x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

            })

            .AddJwtBearer(x =>

            {

                x.TokenValidationParameters = new TokenValidationParameters

                {

                    ValidateIssuer = true,

                    ValidateAudience = true,

                    ValidateLifetime = true,

                    ValidateIssuerSigningKey = true,

                    ValidIssuer = "mySystem",

                    ValidAudience = "myUsers",

                    IssuerSigningKey = symmetricSecurityKey

                };

            });

            services.AddCors();

        }

        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

        {

            if (env.IsDevelopment())

            {

                app.UseDeveloperExceptionPage();

            }

            app.UseRouting();

            app.UseCors(builder =>

                builder.AllowAnyOrigin()

                       .AllowAnyMethod()

                       .AllowAnyHeader());

            app.UseAuthentication();

            app.UseAuthorization();

            app.UseEndpoints(endpoints =>

            {

                endpoints.MapControllers();

            });

        }

    }

}

**Authcontroller.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Authorization;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

using System.Collections.Generic;

using System;

namespace JwtCorsApi.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    [AllowAnonymous]

    public class AuthController : ControllerBase

    {

        [HttpGet("token")]

        public IActionResult GetToken()

        {

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

            return Ok(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);

        }

    }

}

**SCREENSHOTS**(task 5):-

