WebAPI Hands-On Assignment Report

Name: Arya Vats
SupersetID: 6358118
Date:13-07-2025

Table of Contents

Lab 1: REST & WebAPI Basics

- Implemented ValuesController with GET, POST, PUT, DELETE
- Used `ControllerBase`, `[Route]`, and `[Http...]` attributes
- Configured routing and middleware in `Program.cs`

Code Snippet:

csharp
CopyEdit

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

```
public class ValuesController : ControllerBase
{
    [HttpGet]
    public IActionResult Get() => Ok(new[] { "value1", "value2" });
    [HttpPost]
    public IActionResult Post([FromBody] string value) =>
Created("", value);
    [HttpPut("{id}")]
    public IActionResult Put(int id, [FromBody] string val) =>
Ok(new { id, val });
    [HttpDelete("{id}")]
    public IActionResult Delete(int id) => NoContent();
}
```

Lab 2: Swagger & Postman

- Enabled Swagger using `AddSwaggerGen`, `UseSwaggerUI`
- Used Swagger to try out endpoints
- Postman used to test GET/POST calls and check response headers/status codes

Code Snippet:

csharp
CopyEdit

```
builder.Services.AddSwaggerGen();
app.UseSwagger();
app.UseSwaggerUI(c =>
{
    c.SwaggerEndpoint("/swagger/v1/swagger.json", "MyApi v1");
});
```

Lab 3: Models & Filters

- Created custom model `Employee` with nested objects

- Implemented `CustomAuthFilter` and `CustomExceptionFilter`
- Used `[FromBody]` to receive JSON input

Code Snippet:

csharp
CopyEdit

```
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; }
}
```

Lab 4: CRUD Operations

- Used PUT and DELETE to update and remove employees
- Validated ID and returned proper HTTP status codes (`BadRequest`, `Ok`, `NoContent`)

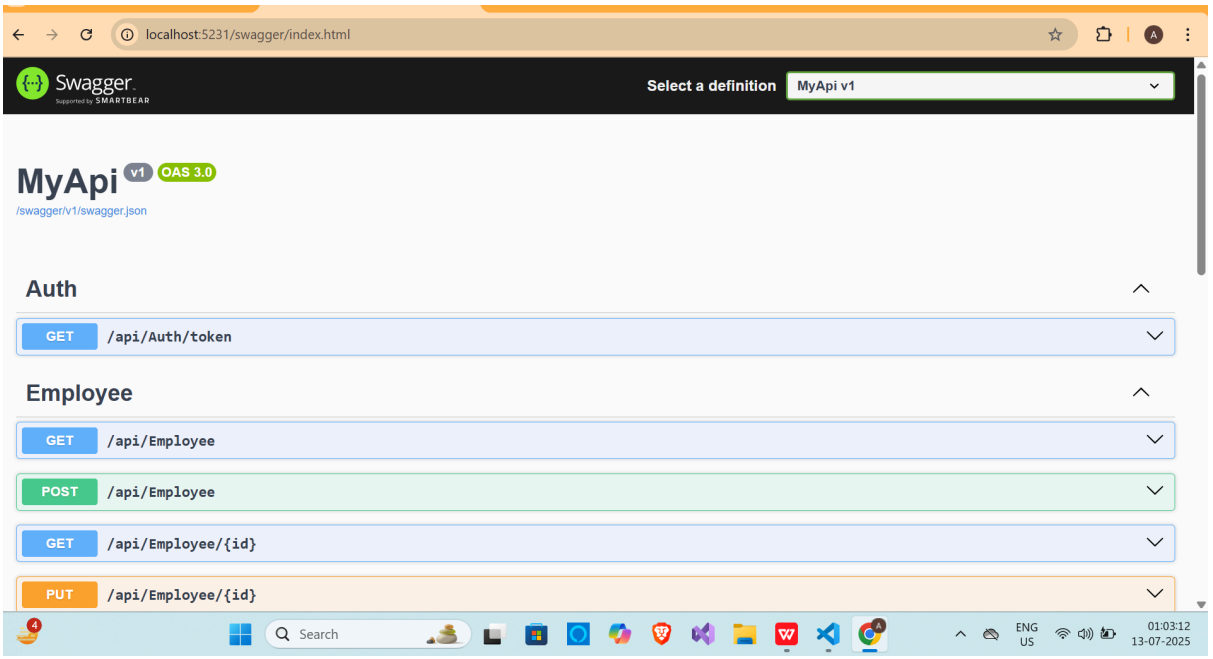
Lab 5: CORS & JWT Authentication

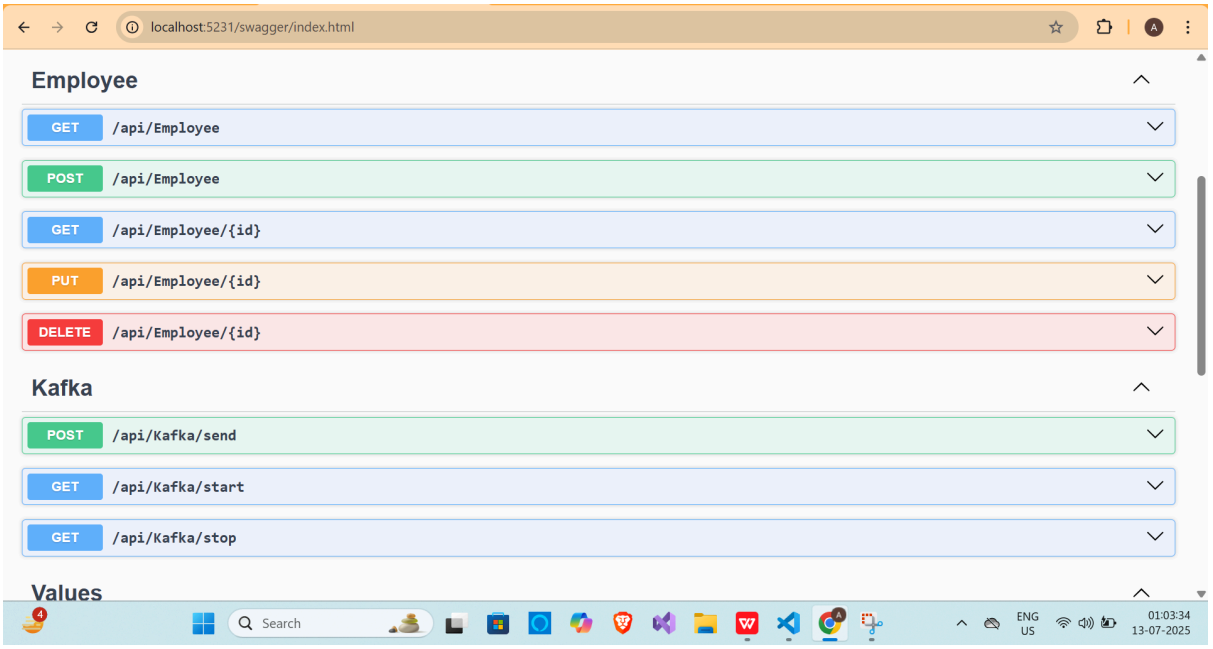
- Generated token via `AuthController`
- Configured `JwtBearer` and added `[Authorize]` to protect Employee API
- Enabled CORS globally with `UseCors()`

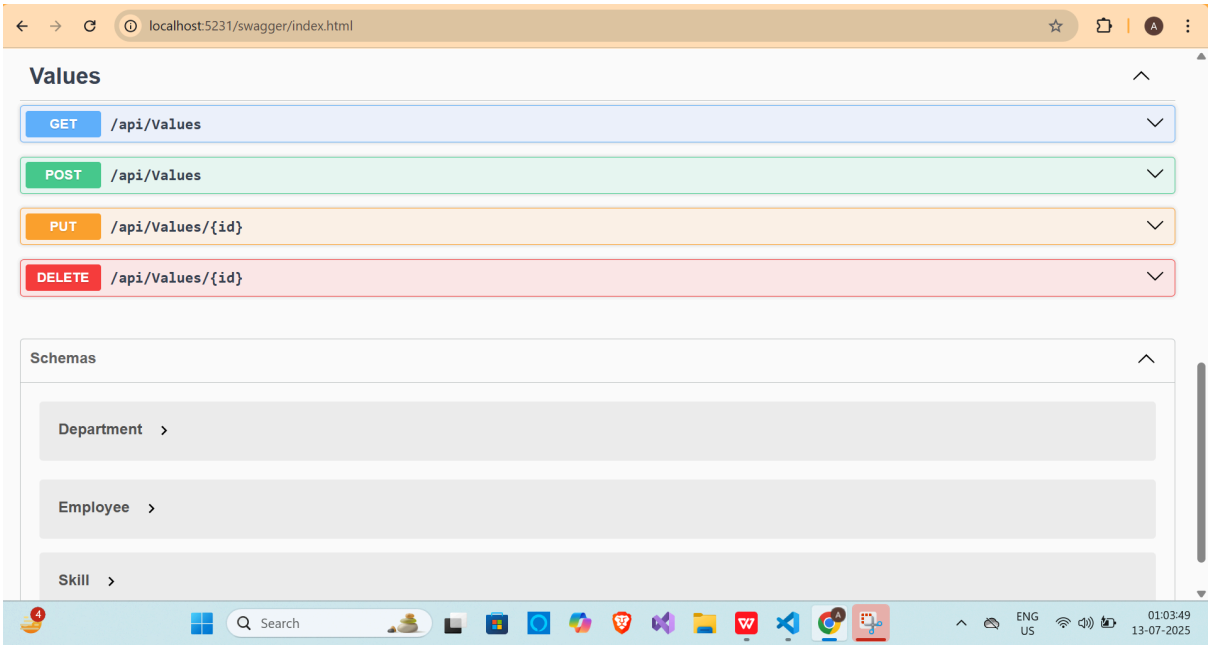
Lab 6: Kafka Integration

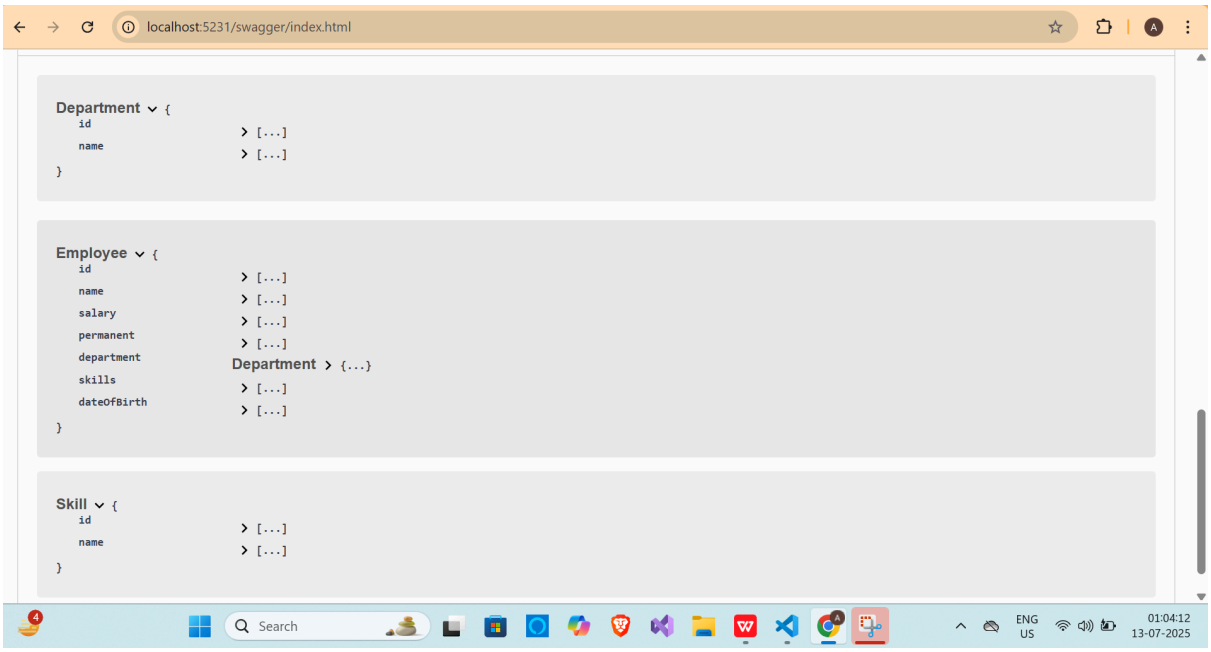
- Created Kafka producer and consumer
- Used `/api/Kafka/send` to produce, `/api/Kafka/start` to consume messages
- Displayed messages in terminal

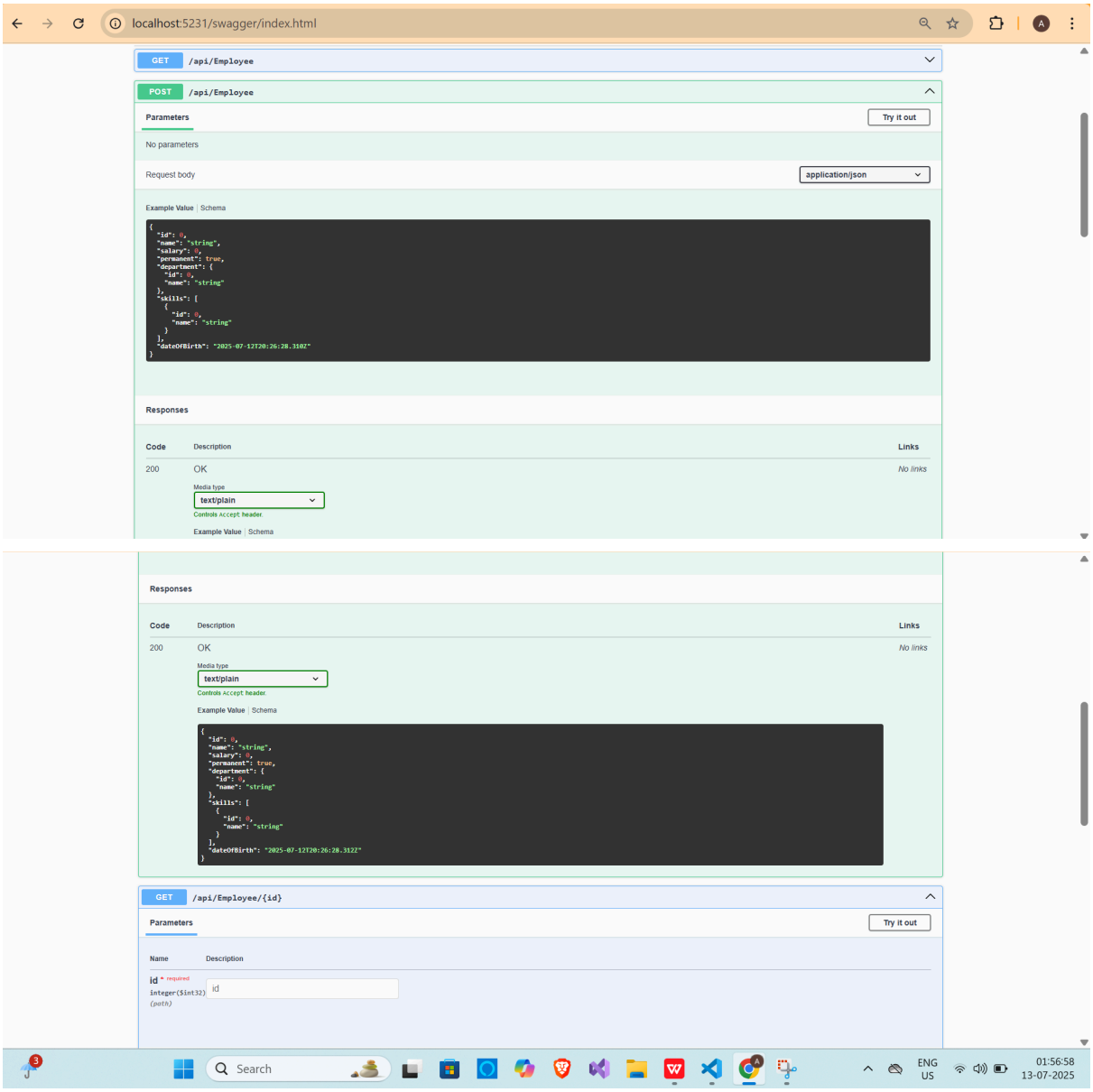
Screenshots & Output Evidence






localhost:5231/swagger/index.html
Values
GET /api/Values
POST /api/Values
PUT /api/Values/{id}
DELETE /api/Values/{id}
Schemas
Department
Employee
Skill
Search
ENG
US
01:03:49
13-07-2025

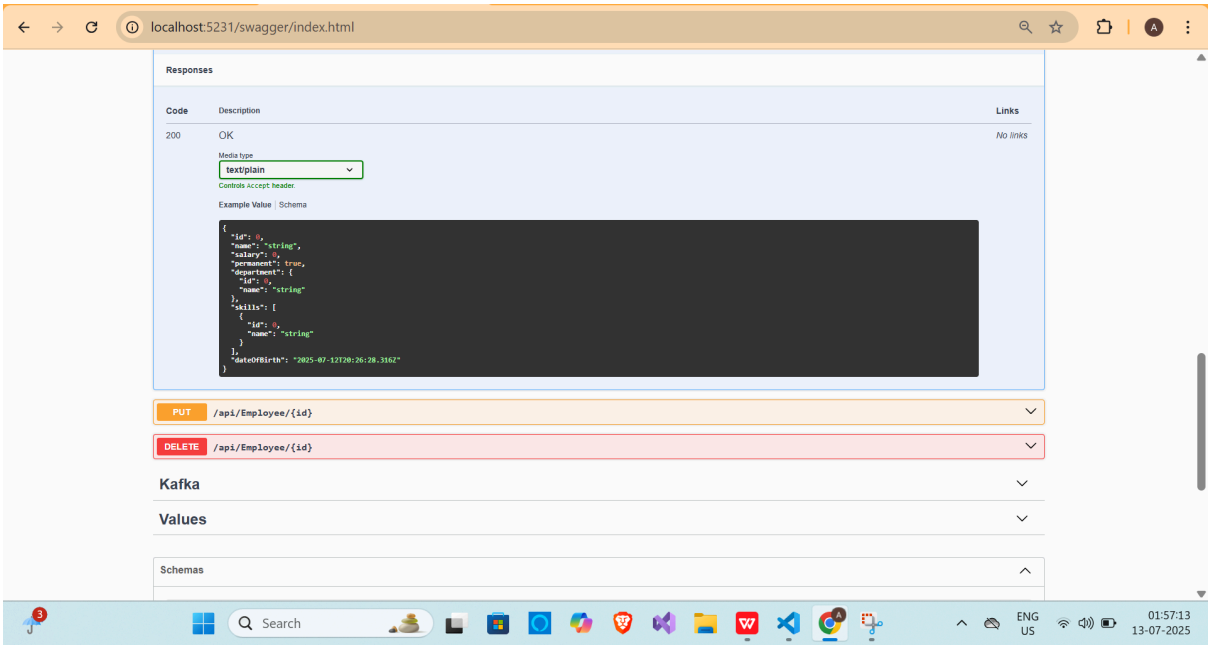

localhost:5231/swagger/index.html
Department { id [...] name [...] }
Employee { id [...] name [...] salary [...] permanent [...] department Department {...} skills [...] dateOfBirth [...] }
Skill { id [...] name [...] }
Search
ENG
US
01:04:12
13-07-2025

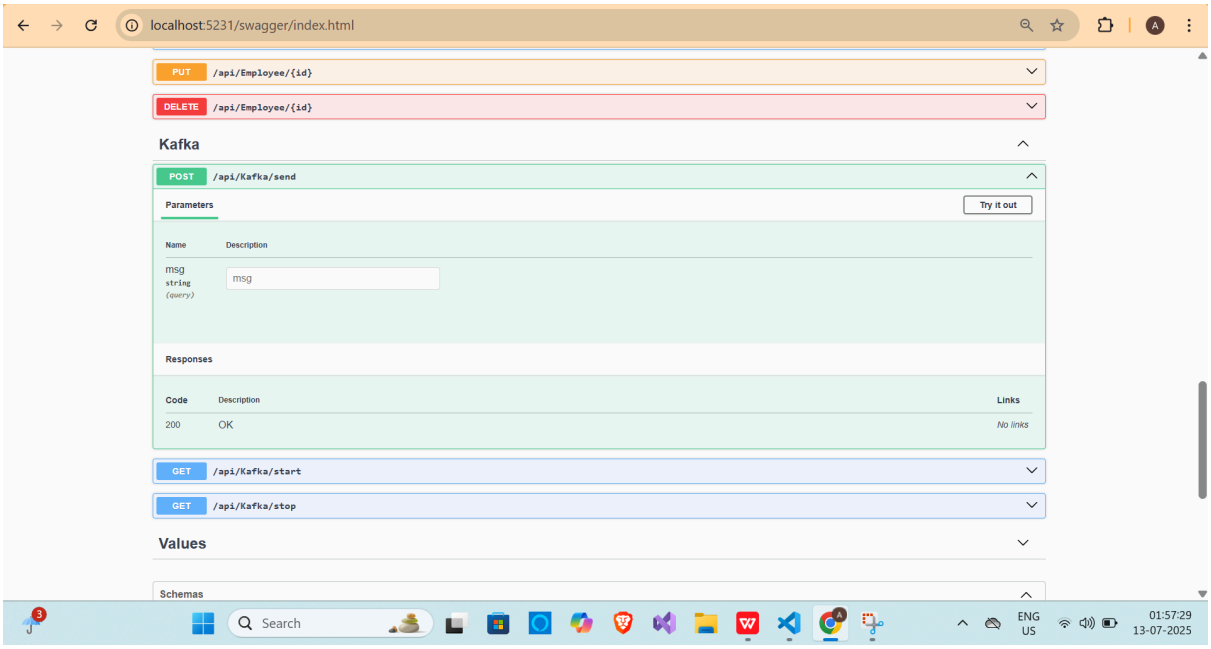

localhost:5231/swagger/index.html
GET /api/Employee
POST /api/Employee
Parameters
Try it out
No parameters
Request body
application/json
Example Value | Schema
{
"id": 0,
"name": "string",
"salary": 0,
"permanent": true,
"department": {
"id": 0,
"name": "string"
},
"skills": [
{
"id": 0,
"name": "string"
}
],
"dateOfBirth": "2025-07-12T20:26:28.3102"
}
Responses
Code
Description
Links
200
OK
No links
Media type
text/plain
Controls Accept header.
Example Value | Schema

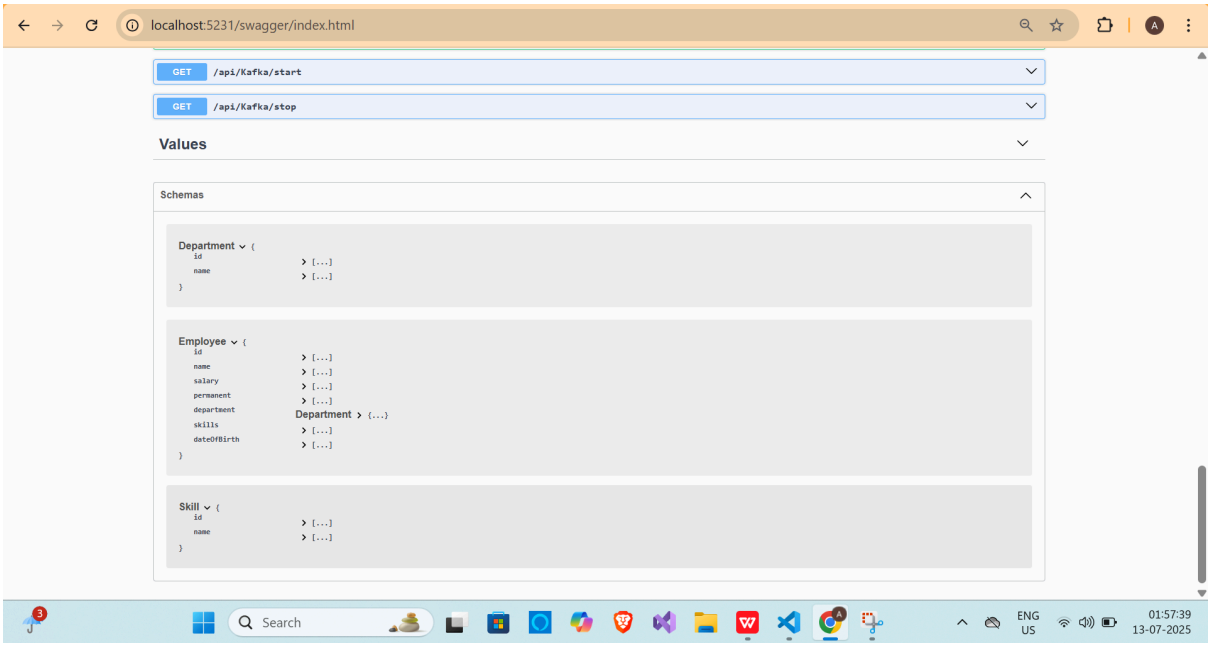
Responses
Code
Description
Links
200
OK
No links
Media type
text/plain
Controls Accept header.
Example Value | Schema
{
"id": 0,
"name": "string",
"salary": 0,
"permanent": true,
"department": {
"id": 0,
"name": "string"
},
"skills": [
{
"id": 0,
"name": "string"
}
],
"dateOfBirth": "2025-07-12T20:26:28.3122"
}

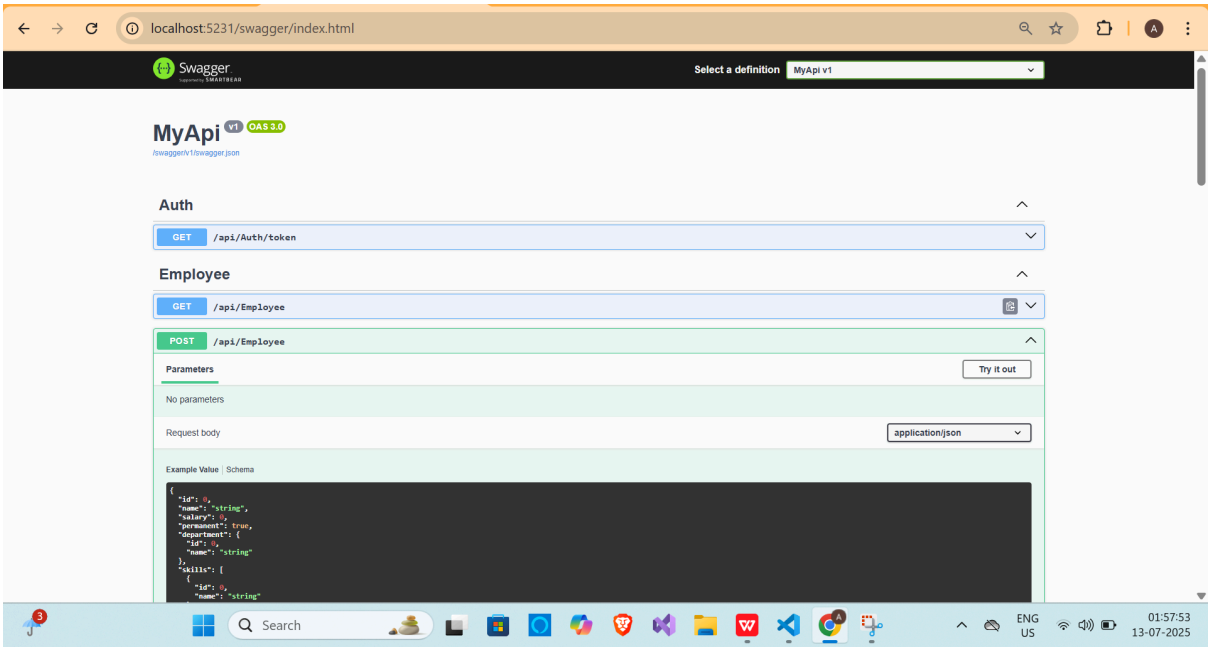
GET /api/Employee/{id}
Parameters
Try it out
Name
Description
id * required
integer($int32)
(path)
id

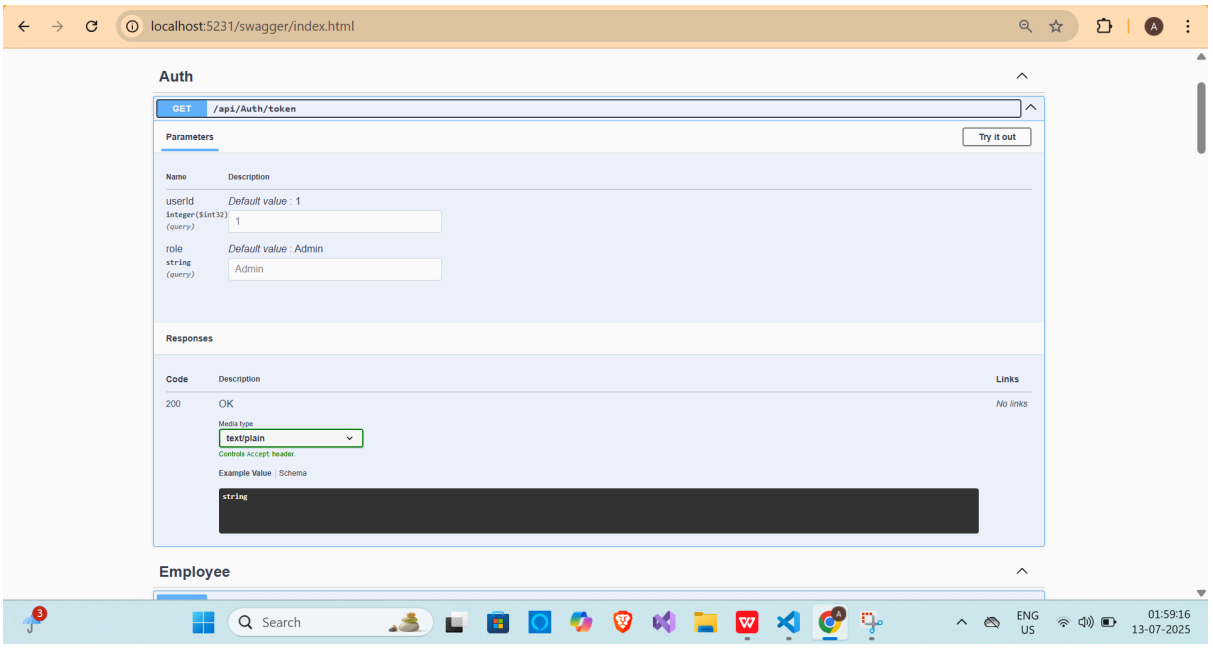
Search
ENG
US
01:56:58
13-07-2025


localhost:5231/swagger/index.html
Responses
Code
Description
Links
200
OK
No links
Media type
text/plain
Controls Accept header.
Example Value | Schema
{
"id": 0,
"name": "string",
"salary": 0,
"permanent": true,
"department": {
"id": 0,
"name": "string"
},
"skills": [
{
"id": 0,
"name": "string"
}
],
"dateOfBirth": "2025-07-12T20:26:28.316Z"
}
PUT /api/Employee/{id}
DELETE /api/Employee/{id}
Kafka
Values
Schemas
Search
ENG
US
01:57:13
13-07-2025


localhost:5231/swagger/index.html
PUT /api/Employee/{id}
DELETE /api/Employee/{id}
Kafka
POST /api/Kafka/send
Parameters
Try it out
Name
Description
msg
string
(query)
msg
Responses
Code
Description
Links
200
OK
No links
GET /api/Kafka/start
GET /api/Kafka/stop
Values
Schemas
Search
ENG
US
01:57:29
13-07-2025


localhost:5231/swagger/index.html
GET /api/Kafka/start
GET /api/Kafka/stop
Values
Schemas
Department { id name } [...] [...]
Employee { id name salary permanent department skills dateOfBirth } [...] [...] [...] [...] Department {...} [...] [...]
Skill { id name } [...] [...]
Search
ENG US
01:57:39 13-07-2025





Full Code Listing:

Program.cs

csharp
CopyEdit

```
// ---------- Program.cs ----------
using Confluent.Kafka;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.IdentityModel.Tokens;
using MyApi.Filters;
using MyApi.KafkaClient;
```

```
using System.Text;

var builder = WebApplication.CreateBuilder(args);
var config = builder.Configuration;

var jwt = config.GetSection("JwtSettings");
var key = Encoding.UTF8.GetBytes(jwt["Key"]);

builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen(c =>
{
    c.SwaggerDoc("v1", new() { Title = "MyApi", Version = "v1" });
});

builder.Services.AddCors(o => o.AddDefaultPolicy(p =>
p.AllowAnyOrigin().AllowAnyHeader().AllowAnyMethod()));

builder.Services.AddAuthentication(o =>
{
    o.DefaultAuthenticateScheme =
JwtBearerDefaults.AuthenticationScheme;
    o.DefaultChallengeScheme    =
JwtBearerDefaults.AuthenticationScheme;
})
.AddJwtBearer(o =>
{
    o.TokenValidationParameters = new TokenValidationParameters
    {
        ValidateIssuer           = true,
        ValidateAudience         = true,
        ValidateLifetime         = true,
        ValidateIssuerSigningKey = true,
        ValidIssuer              = jwt["Issuer"],
        ValidAudience            = jwt["Audience"],
        IssuerSigningKey         = new SymmetricSecurityKey(key)
    };
});

builder.Services.AddSingleton<KafkaProducer>();
builder.Services.AddSingleton<KafkaConsumer>();
```

```
builder.Services.AddMvc(opts =>
opts.Filters.Add<CustomExceptionFilter>());

var app = builder.Build();

app.UseCors();
app.UseSwagger();
app.UseSwaggerUI(c => c.SwaggerEndpoint("/swagger/v1/swagger.json",
"MyApi v1"));

app.UseAuthentication();
app.UseAuthorization();
app.MapControllers();
app.Run();
```

---

appsettings.json

json
CopyEdit

```
{
  "JwtSettings": {
    "Key": "mysuperdupersecret",
    "Issuer": "mySystem",
    "Audience": "myUsers",
    "ExpireMinutes": "10"
  },
  "Kafka": {
    "BootstrapServers": "localhost:9092",
    "Topic": "chat-topic"
  },
  "AllowedHosts": "*"
}
```

Models

Employee.cs

csharp
CopyEdit

```
namespace MyApi.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; }
}
```

Department.cs

csharp
CopyEdit

```
namespace MyApi.Models;
public class Department
{
    public int Id { get; set; }
    public string Name { get; set; }
}
```

Skill.cs

csharp
CopyEdit

```
namespace MyApi.Models;
public class Skill
{
    public int Id { get; set; }
    public string Name { get; set; }
}
```

Filters

CustomAuthFilter.cs

csharp
CopyEdit

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Filters;

namespace MyApi.Filters;
public class CustomAuthFilter : ActionFilterAttribute
```

```
{
    public override void OnActionExecuting(ActionExecutingContext
context)
    {
        if
(!context.HttpContext.Request.Headers.TryGetValue("Authorization",
out var token))
            context.Result = new BadRequestObjectResult("Invalid
request - No Auth token");
        else if (!token.ToString().StartsWith("Bearer "))
            context.Result = new BadRequestObjectResult("Invalid
request - Token present but Bearer unavailable");
    }
}
```

CustomExceptionFilter.cs

csharp
CopyEdit

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Filters;

namespace MyApi.Filters;
public class CustomExceptionFilter : IExceptionFilter
{
    public void OnException(ExceptionContext context)
    {
        context.Result = new ObjectResult(new { error =
context.Exception.Message }) { StatusCode = 500 };
    }
}
```

Controllers

ValuesController.cs

csharp
CopyEdit

```
using Microsoft.AspNetCore.Mvc;

namespace MyApi.Controllers;
[ApiController]
```

```
[Route("api/[controller]")]
public class ValuesController : ControllerBase
{
    [HttpGet]
    public IActionResult Get() => Ok(new[] { "value1", "value2" });

    [HttpPost]
    public IActionResult Post([FromBody] string value) =>
Created("", value);

    [HttpPut("{id}")]
    public IActionResult Put(int id, [FromBody] string val) =>
Ok(new { id, val });

    [HttpDelete("{id}")]
    public IActionResult Delete(int id) => NoContent();
}
```

EmployeeController.cs

csharp

CopyEdit

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using MyApi.Models;

namespace MyApi.Controllers;
[ApiController]
[Route("api/[controller]")]
[Authorize(Roles = "Admin,POC")]
public class EmployeeController : ControllerBase
{
    private static List<Employee> _list = new()
    {
        new()
        {
            Id = 1,
            Name = "Alice",
            Salary = 50000,
            Permanent = true,
            Department = new() { Id = 10, Name = "HR" },
            Skills = new() { new Skill { Id = 1, Name = "C#" } },
```

```
            DateOfBirth = DateTime.Parse("1990-01-01")
        }
    };

    [HttpGet]
    public ActionResult<List<Employee>> Get() => Ok(_list);

    [HttpGet("{id}")]
    public ActionResult<Employee> Get(int id)
    {
        var emp = _list.FirstOrDefault(x => x.Id == id);
        return emp == null ? BadRequest("Invalid ID") : Ok(emp);
    }

    [HttpPost]
    public ActionResult<Employee> Post([FromBody] Employee emp)
    {
        emp.Id = _list.Max(x => x.Id) + 1;
        _list.Add(emp);
        return CreatedAtAction(nameof(Get), new { id = emp.Id },
emp);
    }

    [HttpPut("{id}")]
    public ActionResult<Employee> Put(int id, [FromBody] Employee
emp)
    {
        var index = _list.FindIndex(x => x.Id == id);
        if (index < 0) return BadRequest("Invalid ID");
        emp.Id = id;
        _list[index] = emp;
        return Ok(emp);
    }

    [HttpDelete("{id}")]
    public IActionResult Delete(int id)
    {
        var emp = _list.FirstOrDefault(x => x.Id == id);
        if (emp == null) return BadRequest("Invalid ID");
        _list.Remove(emp);
        return NoContent();
```

```
    }
}
```

AuthController.cs

csharp
CopyEdit

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;

namespace MyApi.Controllers;
[ApiController]
[Route("api/[controller]")]
[AllowAnonymous]
public class AuthController : ControllerBase
{
    private readonly IConfiguration _config;
    public AuthController(IConfiguration cfg) => _config = cfg;

    [HttpGet("token")]
    public ActionResult<string> GetToken(int userId = 1, string role
= "Admin")
    {
        var jwt = _config.GetSection("JwtSettings");
        var key  = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwt["Key"]));
        var creds = new SigningCredentials(key,
SecurityAlgorithms.HmacSha256);
        var claims = new[]
        {
            new Claim(ClaimTypes.Role, role),
            new Claim("UserId", userId.ToString())
        };

        var token = new JwtSecurityToken(
            issuer: jwt["Issuer"],
            audience: jwt["Audience"],
            claims: claims,
```

```
            expires:
DateTime.Now.AddMinutes(double.Parse(jwt["ExpireMinutes"])),
            signingCredentials: creds
        );

        return new JwtSecurityTokenHandler().WriteToken(token);
    }
}
```

KafkaController.cs

csharp
CopyEdit

```
using Microsoft.AspNetCore.Mvc;
using MyApi.KafkaClient;

namespace MyApi.Controllers;
[ApiController]
[Route("api/[controller]")]
public class KafkaController : ControllerBase
{
    private readonly KafkaProducer _producer;
    private readonly KafkaConsumer _consumer;
    private static CancellationTokenSource _cts = new();

    public KafkaController(KafkaProducer prod, KafkaConsumer cons)
    {
        _producer = prod;
        _consumer = cons;
    }

    [HttpPost("send")]
    public async Task<IActionResult> Send(string msg)
    {
        await _producer.ProduceAsync(msg);
        return Ok();
    }

    [HttpGet("start")]
    public IActionResult Start()
    {
        _cts = new();
```

```
        Task.Run(() => _consumer.Consume(_cts.Token));
        return Ok("Kafka consumer started");
    }

    [HttpGet("stop")]
    public IActionResult Stop()
    {
        _cts.Cancel();
        return Ok("Kafka consumer stopped");
    }
}
```

KafkaClient

KafkaProducer.cs

csharp
CopyEdit

```
using Confluent.Kafka;
using Microsoft.Extensions.Configuration;

namespace MyApi.KafkaClient;
public class KafkaProducer
{
    private readonly IProducer<Null, string> _producer;
    private readonly string _topic;
    public KafkaProducer(IConfiguration cfg)
    {
        var p = new ProducerConfig { BootstrapServers =
cfg["Kafka:BootstrapServers"] };
        _producer = new ProducerBuilder<Null, string>(p).Build();
        _topic    = cfg["Kafka:Topic"];
    }
    public async Task ProduceAsync(string msg) =>
        await _producer.ProduceAsync(_topic, new Message<Null,
string> { Value = msg });
}
```

KafkaConsumer.cs

csharp
CopyEdit

```
using Confluent.Kafka;
using Microsoft.Extensions.Configuration;

namespace MyApi.KafkaClient;
public class KafkaConsumer
{
    private readonly IConfiguration _cfg;
    public KafkaConsumer(IConfiguration cfg) => _cfg = cfg;
    public void Consume(CancellationToken ct)
    {
        var c = new ConsumerConfig
        {
            BootstrapServers = _cfg["Kafka:BootstrapServers"],
            GroupId          = "chat-consumers",
            AutoOffsetReset  = AutoOffsetReset.Earliest
        };
        using var consumer = new ConsumerBuilder<Ignore,
string>(c).Build();
        consumer.Subscribe(_cfg["Kafka:Topic"]);
        while (!ct.IsCancellationRequested)
        {
            var res = consumer.Consume(ct);
            Console.WriteLine("Received: " + res.Message.Value);
        }
    }
}
```

Conclusion

This WebAPI assignment gave me hands-on experience with:

- RESTful API design
- Swagger documentation
- JWT-based authentication
- CORS policy handling
- Custom model and filter creation

- Kafka-based message streaming in .NET