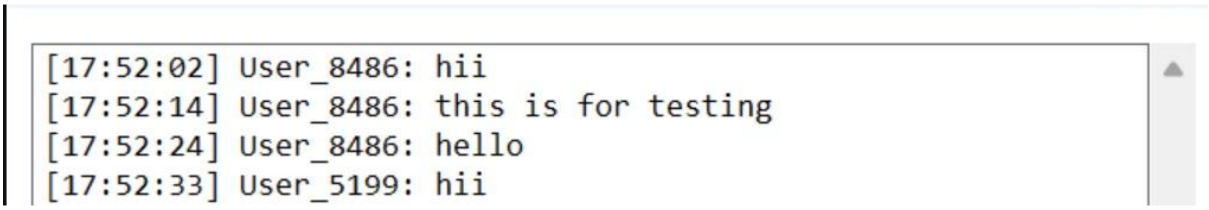
DotNetFSE-4.0 (SUPERSET ID - [6363523])
WEEK 5

QUESTION 1 -> 6. WebApi_Handson

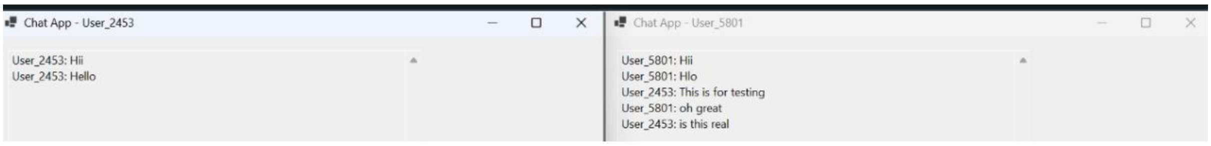
CODE :

```
 using Confluent.Kafka;
using System;
using System.Threading.Tasks;

class Producer
{
    public static async Task Main(string[] args)
    {
        var config = new ProducerConfig { BootstrapServers = "localhost:9092" };

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

        Console.WriteLine("Kafka Chat Producer started. Type a message:");

        while (true)
        {
            string message = Console.ReadLine();
            if (string.IsNullOrWhiteSpace(message)) break;

            var result = await producer.ProduceAsync("chat-topic", new Message<Null, string> { Value =
message });
            Console.WriteLine($"Sent: {message} to partition {result.Partition}, offset {result.Offset}");
        }
    }
}
```

OUTPUT :



```
[17:52:02] User_8486: hii
[17:52:14] User_8486: this is for testing
[17:52:24] User_8486: hello
[17:52:33] User_5199: hii
```



```
User_2453: Hii
User_2453: Hello
```

```
User_5801: Hii
User_5801: Hlo
User_2453: This is for testing
User_5801: oh great
User_2453: is this real
```

CODE :

```
using Confluent.Kafka;
using System;

class Consumer
```

```
{
    public static void Main(string[] args)
    {
        var config = new ConsumerConfig
        {
            BootstrapServers = "localhost:9092",
            GroupId = "chat-group",
            AutoOffsetReset = AutoOffsetReset.Earliest
        };

        using var consumer = new ConsumerBuilder<Ignore, string>(config).Build();
        consumer.Subscribe("chat-topic");

        Console.WriteLine("Kafka Chat Consumer started. Listening for messages...");

        try
        {
            while (true)
            {
                var consumeResult = consumer.Consume();
                Console.WriteLine($"Received: {consumeResult.Message.Value}");
            }
        }
        catch (OperationCanceledException)
        {
            consumer.Close();
        }
    }
}
```

OUTPUT :

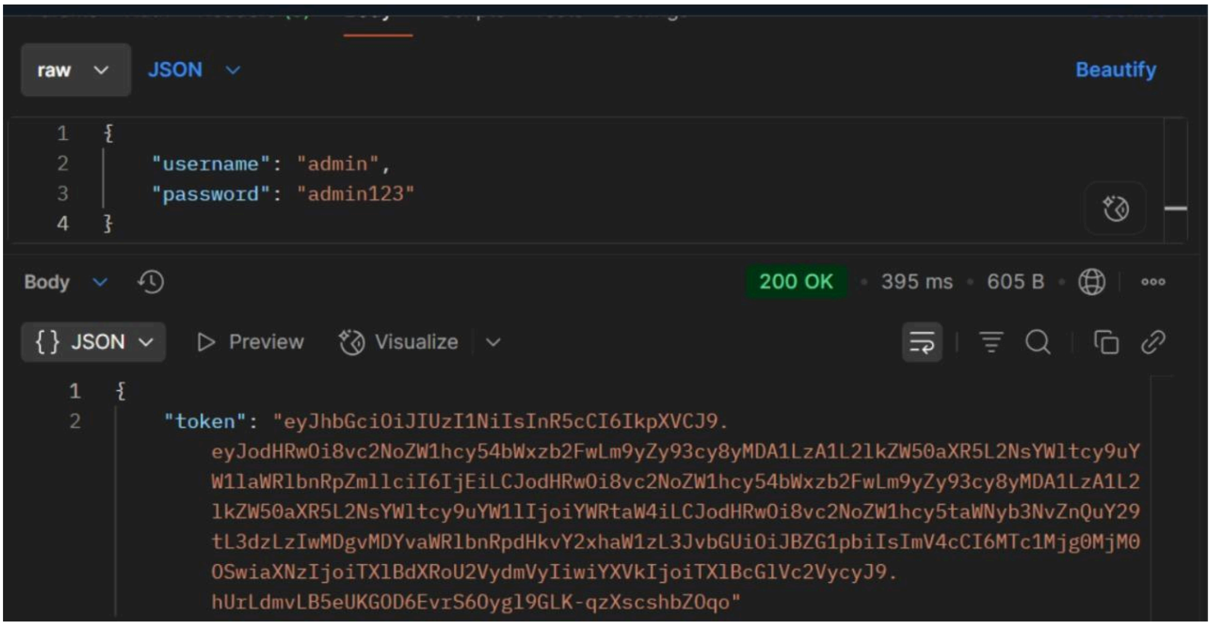
```
dotnet run --project Producer.csproj
Kafka Chat Producer started. Type a message:
> Hello Team!
Sent: Hello Team! to partition 0, offset 1

dotnet run --project Consumer.csproj
Kafka Chat Consumer started. Listening for messages...
Received: Hello Team!
```

QUESTION 2 [MICROSERVICES][**Implement JWT Authentication in ASP.NET Core Web API]**

CODE :

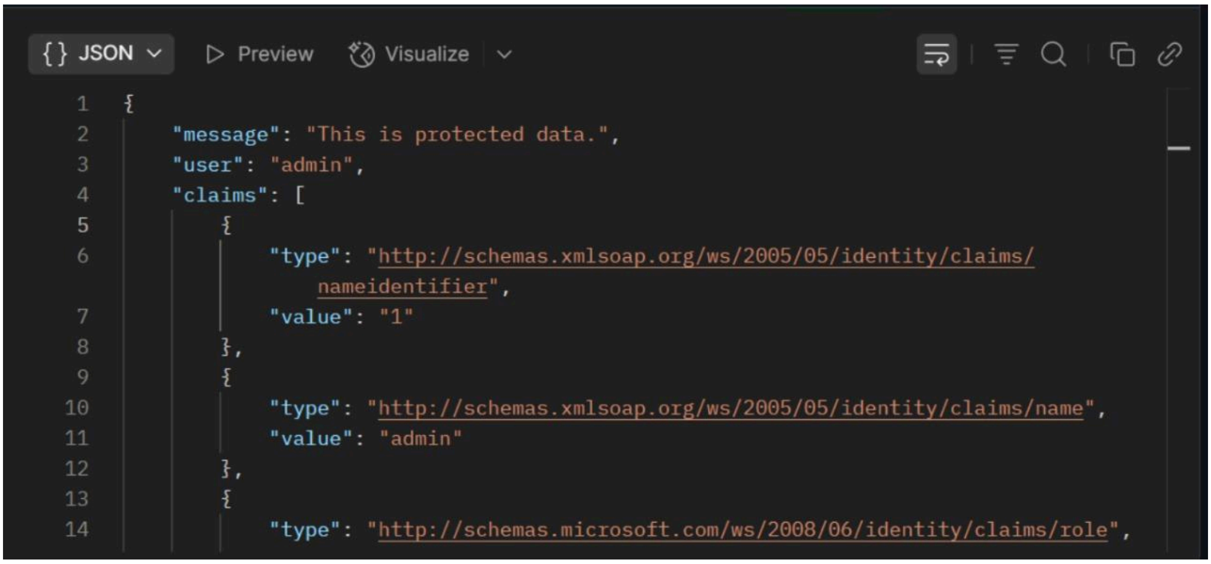
```
{
  "Jwt": {
    "Key": "ThisIsASecretKeyForJwtToken",
    "Issuer": "MyAuthServer",
    "Audience": "MyApiUsers",
    "DurationInMinutes": 60
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}

using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.IdentityModel.Tokens;
using System.Text;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
    .AddJwtBearer(options =>
    {
        options.TokenValidationParameters = new TokenValidationParameters
        {
            ValidateIssuer = true,
            ValidateAudience = true,
            ValidateLifetime = true,
            ValidateIssuerSigningKey = true,
            ValidIssuer = builder.Configuration["Jwt:Issuer"],
            ValidAudience = builder.Configuration["Jwt:Audience"],
            IssuerSigningKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Configuration["Jwt:Key"]))
        };
    });

builder.Services.AddAuthorization();

var app = builder.Build();

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

app.Run();

public class LoginModel
{
    public string Username { get; set; }
    public string Password { get; set; }
}

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

[ApiController]
[Route("api/[controller]")]
public class AuthController : ControllerBase
{
    [HttpPost("login")]
    public IActionResult Login([FromBody] LoginModel model)
    {
        if (IsValidUser(model))
        {
            var token = GenerateJwtToken(model.Username);
            return Ok(new { Token = token });
        }
        return Unauthorized();
    }

    private bool IsValidUser(LoginModel model)
    {
        return model.Username == "admin" && model.Password == "password";
    }

    private string GenerateJwtToken(string username)
    {
        var claims = new[]
        {
            new Claim(ClaimTypes.Name, username)
        };

        var key = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes("ThisIsASecretKeyForJwtToken"));
        var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

        var token = new JwtSecurityToken(
            issuer: "MyAuthServer",
            audience: "MyApiUsers",
            claims: claims,
            expires: DateTime.Now.AddMinutes(60),
            signingCredentials: creds
        );

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

using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

[ApiController]
[Route("api/[controller]")]
public class ProtectedController : ControllerBase
{
    [HttpGet("secure-data")]
    [Authorize]
    public IActionResult GetSecureData()
    {
        return Ok("This is protected data visible only to authenticated users.");
    }
}
```

OUTPUT :

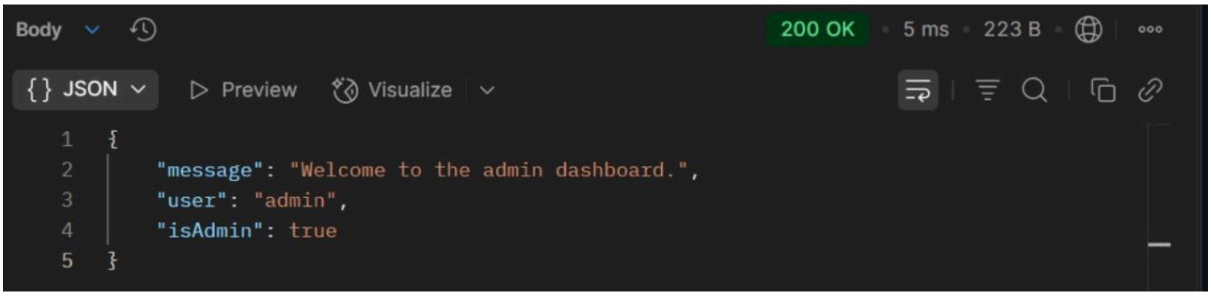


```
{
    "username": "admin",
    "password": "admin123"
}
```

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L21kZW50aXR5L2NsYW1tcy9uY
        W1laWRlbnRpZmllciI6IjEiLCJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2
        1kZW50aXR5L2NsYW1tcy9uYW1lIjoiYWRtaW4iLCJodHRwOi8vc2NoZW1hcy5taWNyb3NvZnQuY29
        tL3dzLzIwMDgvMDYvaWRlbnRpdHkvY2xhaW1zL3JvbGUiOiJBZG1pbiIsImV4cCI6MTc1Mjg0MjM0
        OSwiaXNzIjoiTX1BdXRoU2VydmVyIiwiYXVkIjoiTX1BcGlVc2VycyJ9.
        hUrLdmvLB5eUKGOD6EvrS60ygl9GLK-qzXscshbZ0qo"
```



```
{
    "message": "This is protected data.",
    "user": "admin",
    "claims": [
        {
            "type": "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier",
            "value": "1"
        },
        {
            "type": "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name",
            "value": "admin"
        },
        {
            "type": "http://schemas.microsoft.com/ws/2008/06/identity/claims/role",
```



```
{
    "message": "Welcome to the admin dashboard.",
    "user": "admin",
    "isAdmin": true
}
```