**Week 5 Hands On**

**Kafka Integration with C#**

1. Create a Chat Application which uses Kafka as a streaming platform and consume the chat messages in the command prompt.

**Publisher:**

using Confluent.Kafka;

var config = new ProducerConfig { BootstrapServers = "localhost:9092" };

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

*while* (true)

{

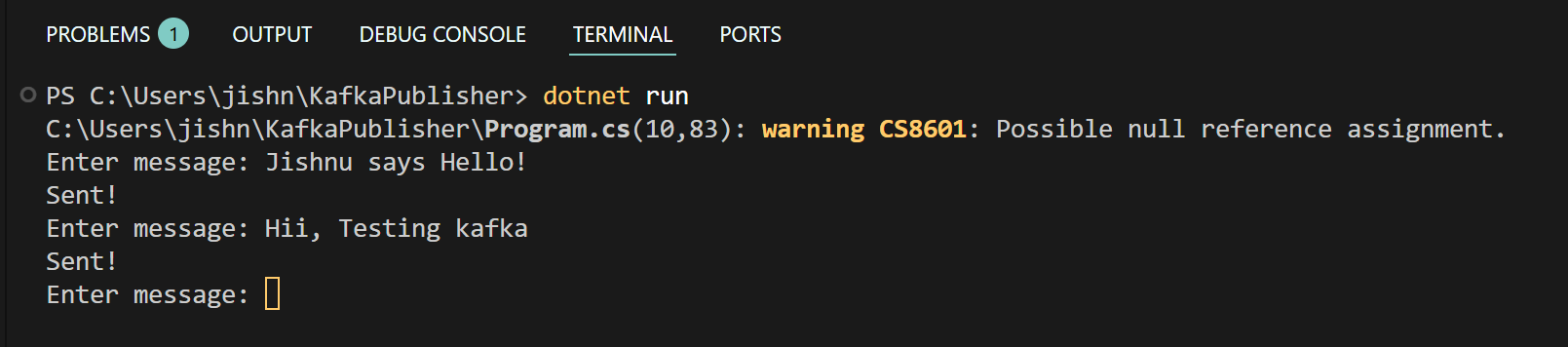
    Console.Write("Enter message: ");

    var msg = Console.ReadLine();

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

    Console.WriteLine("Sent!");

}



**Consumer:**

using Confluent.Kafka;

var config = new ConsumerConfig

{

    GroupId = "chat-consumers",

    BootstrapServers = "localhost:9092",

    AutoOffsetReset = AutoOffsetReset.Earliest

};

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

consumer.Subscribe("chat-topic");

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

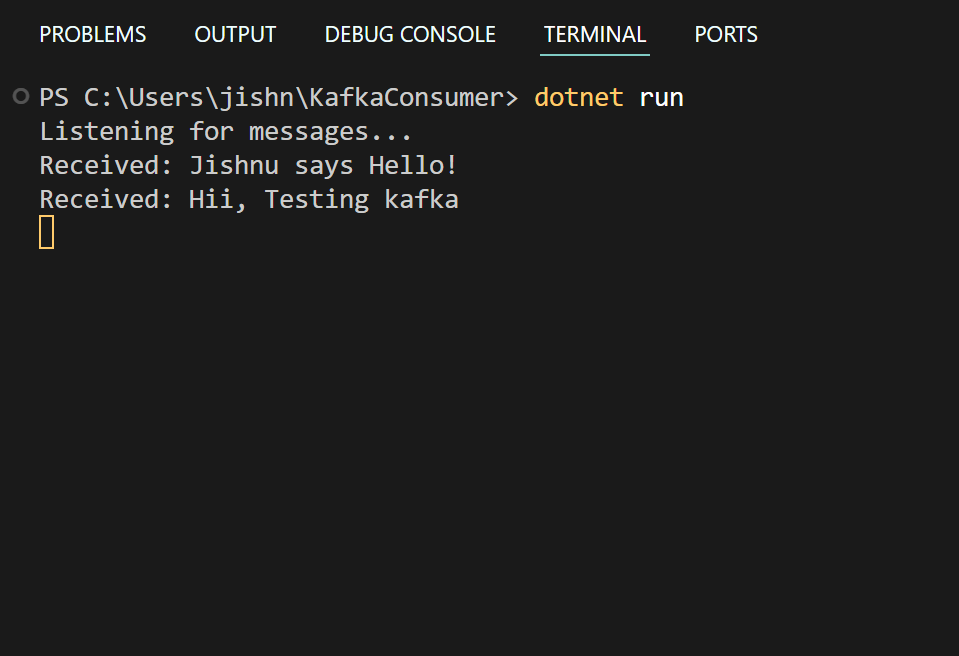
*while* (true)

{

    var result = consumer.Consume();

    Console.WriteLine($"Received: {result.Message.Value}");

}



1. Create a Chat Application using C# Windows Application using Kafka and consume the message in different client applications.

**Form1.Designer.cs:**

namespace KafkaChatClient;

partial class Form1

{

private System.ComponentModel.IContainer components = null;

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

private void InitializeComponent()

{

txtMessage = new TextBox();

btnSend = new Button();

lstMessages = new ListBox();

SuspendLayout();

//

// txtMessage

//

txtMessage.Location = new Point(90, 328);

txtMessage.Name = "txtMessage";

txtMessage.Size = new Size(449, 39);

txtMessage.TabIndex = 0;

//

// btnSend

//

btnSend.Location = new Point(587, 324);

btnSend.Name = "btnSend";

btnSend.Size = new Size(150, 46);

btnSend.TabIndex = 1;

btnSend.Text = "Send";

btnSend.UseVisualStyleBackColor = true;

btnSend.Click += btnSend\_Click; // ✅ corrected

//

// lstMessages

//

lstMessages.FormattingEnabled = true;

lstMessages.Location = new Point(90, 103);

lstMessages.Name = "lstMessages";

lstMessages.Size = new Size(647, 164);

lstMessages.TabIndex = 2;

AutoScaleDimensions = new SizeF(13F, 32F);

AutoScaleMode = AutoScaleMode.Font;

ClientSize = new Size(949, 545);

Controls.Add(lstMessages);

Controls.Add(btnSend);

Controls.Add(txtMessage);

Name = "Form1";

Text = "Kafka Chat Client";

ResumeLayout(false);

PerformLayout();

}

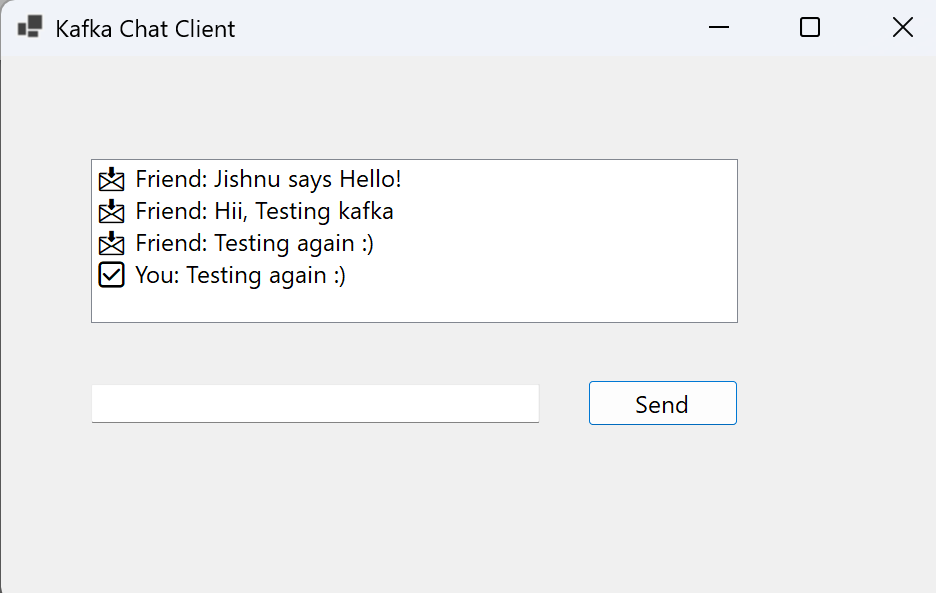
#endregion

private TextBox txtMessage;

private Button btnSend;

private ListBox lstMessages;

}



**MicroServices**

**Implementing JWT Authentication in Web API**

**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();

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.UseSwagger();

app.UseSwaggerUI();

app.UseHttpsRedirection();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

**LoginModel.cs**

namespace JwtAuthService.Models;

public class LoginModel

{

    public string Username { get; set; }

    public string Password { get; set; }

}

**AuthController.cs**

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

using JwtAuthService.Models;

[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);

    }

}

