**6. 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.**

**ProducerApp.cs**

using Confluent.Kafka;

using System;

using System.Threading.Tasks;

public class ProducerApp

{

public static async Task RunProducer()

{

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

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

Console.WriteLine("Kafka Producer Started. Type messages below:");

while (true)

{

Console.Write("You: ");

var input = Console.ReadLine();

if (input == "exit") break;

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

}

}

}

**ConsumerApp.cs**

using Confluent.Kafka;

using System;

using System.Threading;

public class ConsumerApp

{

public static void RunConsumer()

{

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 Consumer Started. Listening for messages...");

while (true)

{

var consumeResult = consumer.Consume(CancellationToken.None);

Console.WriteLine($"Friend: {consumeResult.Message.Value}");

}

}

}

**Program.cs**

using System;

using System.Threading.Tasks;

class Program

{

static async Task Main(string[] args)

{

if (args.Length == 0)

{

Console.WriteLine("Usage: dotnet run [Producer|Consumer]");

return;

}

switch (args[0].ToLower())

{

case "producer":

await ProducerApp.RunProducer();

break;

case "consumer":

ConsumerApp.RunConsumer();

break;

default:

Console.WriteLine("Invalid argument. Use 'Producer' or 'Consumer'");

break;

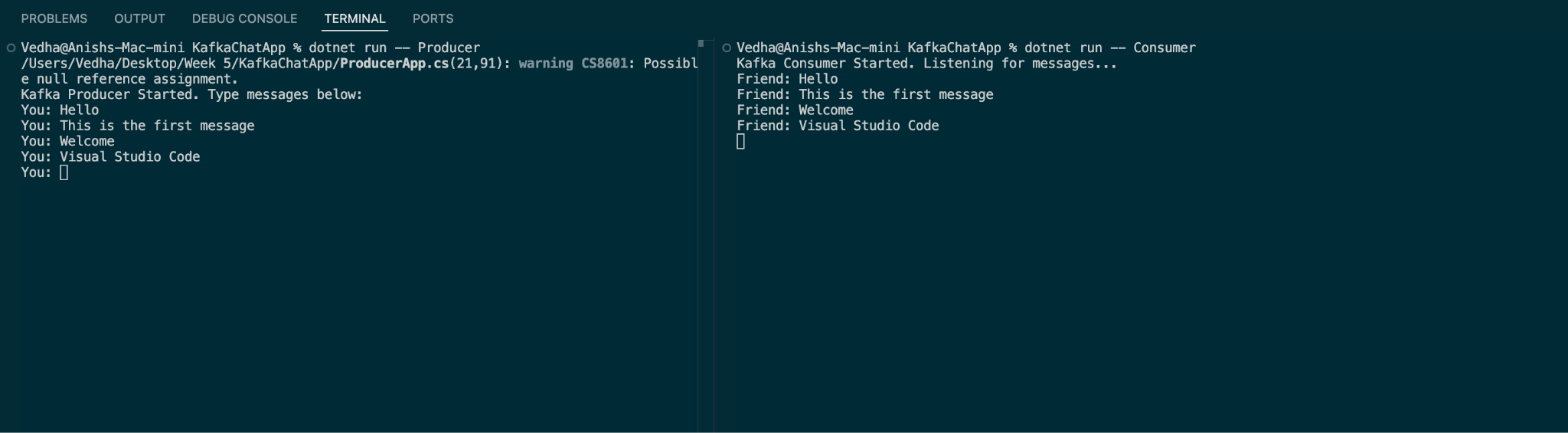
}

}

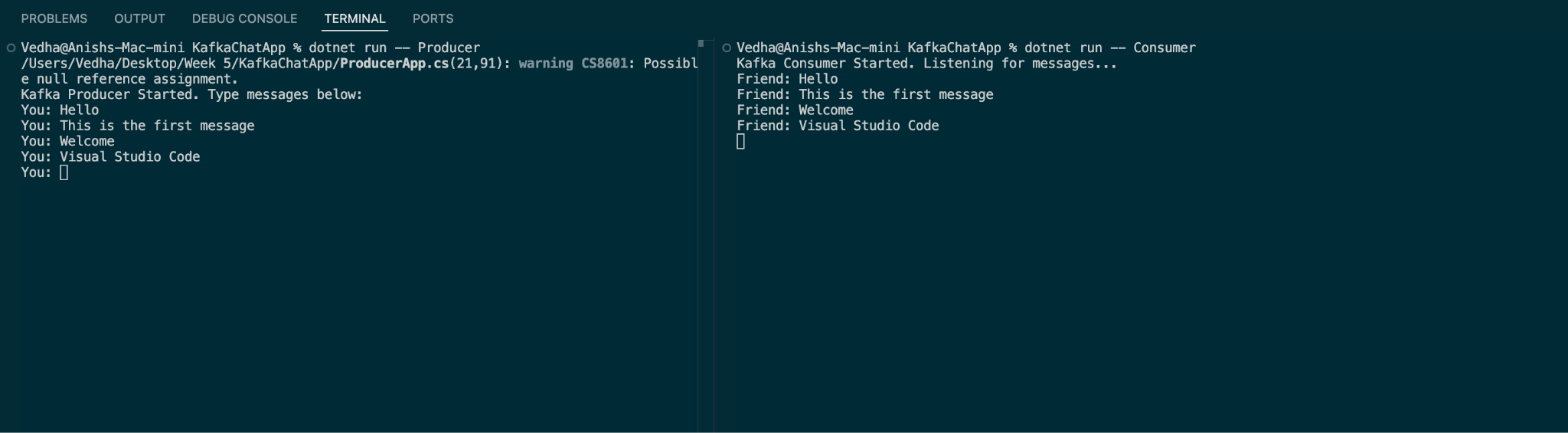
}

**Output:**

**Producer:**



**Consumer:**

****

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

**ConsumerApp.cs**

using Confluent.Kafka;

using System;

using System.Threading;

public class ConsumerApp

{

public static void RunConsumer()

{

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 Consumer Started. Listening for messages...");

while (true)

{

var consumeResult = consumer.Consume(CancellationToken.None);

Console.WriteLine($"Friend: {consumeResult.Message.Value}");

}

**MainWindow.axaml.cs**

using Avalonia.Controls;

using Avalonia.Interactivity;

using Confluent.Kafka;

using System;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using Avalonia.Threading;

namespace KafkaChatGui

{

public partial class MainWindow : Window

{

private readonly string topic = "chat-topic";

private IProducer<Null, string> \_producer;

private IConsumer<Ignore, string> \_consumer;

private CancellationTokenSource \_cts = new();

public MainWindow()

{

InitializeComponent();

SetupKafka();

StartConsumer();

}

private void SetupKafka()

{

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

\_producer = new ProducerBuilder<Null, string>(config).Build();

}

private void StartConsumer()

{

Task.Run(() =>

{

var config = new ConsumerConfig

{

BootstrapServers = "localhost:9092",

GroupId = "chat-group-ui",

AutoOffsetReset = AutoOffsetReset.Earliest

};

\_consumer = new ConsumerBuilder<Ignore, string>(config).Build();

\_consumer.Subscribe(topic);

try

{

while (!\_cts.Token.IsCancellationRequested)

{

var cr = \_consumer.Consume(\_cts.Token);

Dispatcher.UIThread.Post(() =>

{

ChatLog.Text += $"Friend: {cr.Message.Value}\n";

});

}

}

catch (OperationCanceledException) { }

});

}

private async void SendButton\_Click(object? sender, RoutedEventArgs e)

{

var message = MessageBox.Text;

if (!string.IsNullOrWhiteSpace(message))

{

await \_producer.ProduceAsync(topic, new Message<Null, string> { Value = message });

ChatLog.Text += $"You: {message}\n";

MessageBox.Text = "";

}

}

protected override void OnClosed(EventArgs e)

{

base.OnClosed(e);

\_cts.Cancel();

\_consumer?.Close();

}

}

}

**MainWindow.axaml**

<Window xmlns="https://github.com/avaloniaui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="https://github.com/avaloniaui"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

x:Class="KafkaChatGui.MainWindow"

Width="400" Height="500" Title="Kafka Chat GUI">

<StackPanel Margin="10">

<TextBlock Text="Kafka Chat" FontSize="18" HorizontalAlignment="Center" Margin="0,0,0,10"/>

<TextBox Name="MessageBox" Watermark="Type your message..." Margin="0,0,0,10"/>

<Button Name="SendButton" Content="Send" Click="SendButton\_Click" Width="80" HorizontalAlignment="Left" Margin="0,0,0,10"/>

<TextBlock Text="Messages:" FontWeight="Bold"/>

<ScrollViewer Height="300">

<TextBlock Name="ChatLog" TextWrapping="Wrap" />

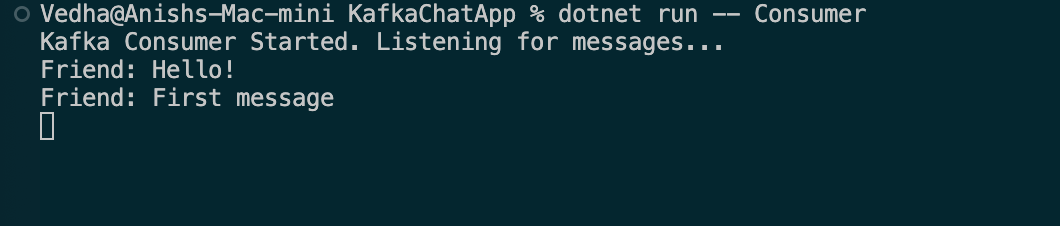
</ScrollViewer>

</StackPanel>

</Window>

**Output:**

**Consumer:**

****

**GUI:**

