**WEEK-5 ASP.NET Core 8.0 Web API**

**6. WebApi\_Handson**

**Hands On:**

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

Start Zookeeper (CMD 1)  
cd C:\kafka  
bin\windows\zookeeper-server-start.bat config\zookeeper.properties

Start Kafka (CMD 2)  
cd C:\kafka  
bin\windows\kafka-server-start.bat config\server.properties

Create Topic (CMD 3)  
cd C:\kafka  
bin\windows\kafka-topics.bat --create --topic chat-topic --bootstrap-server localhost:9092

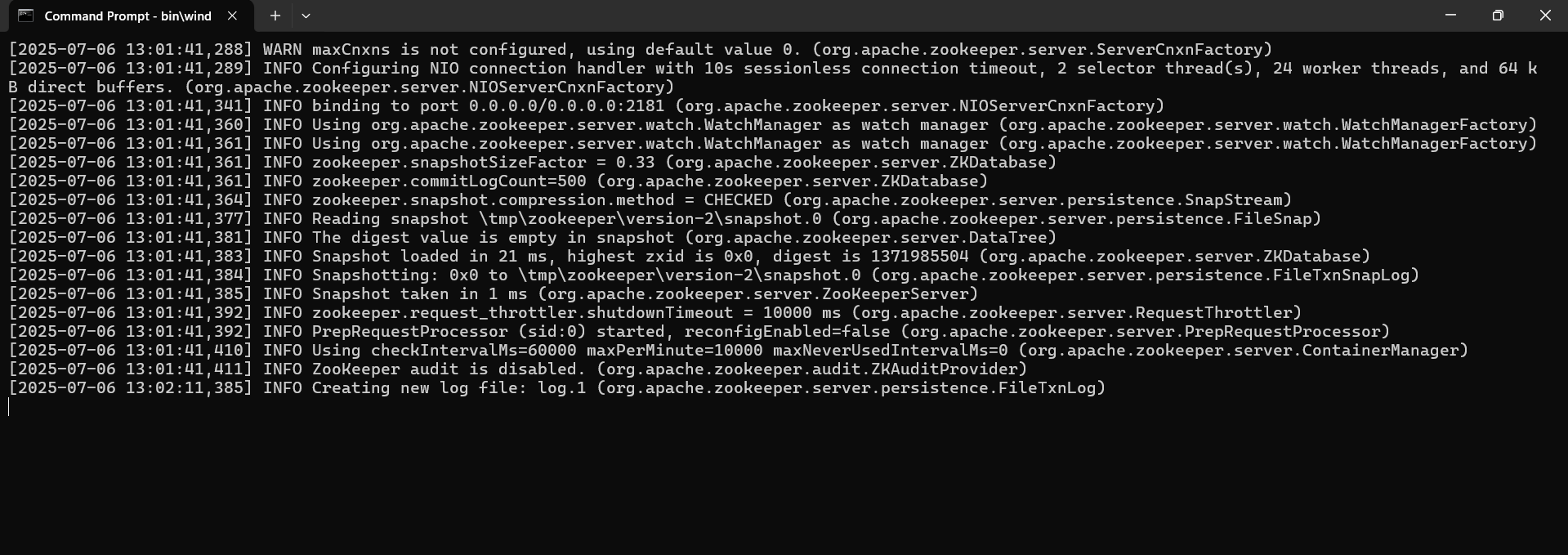
**KafkaPublisher**

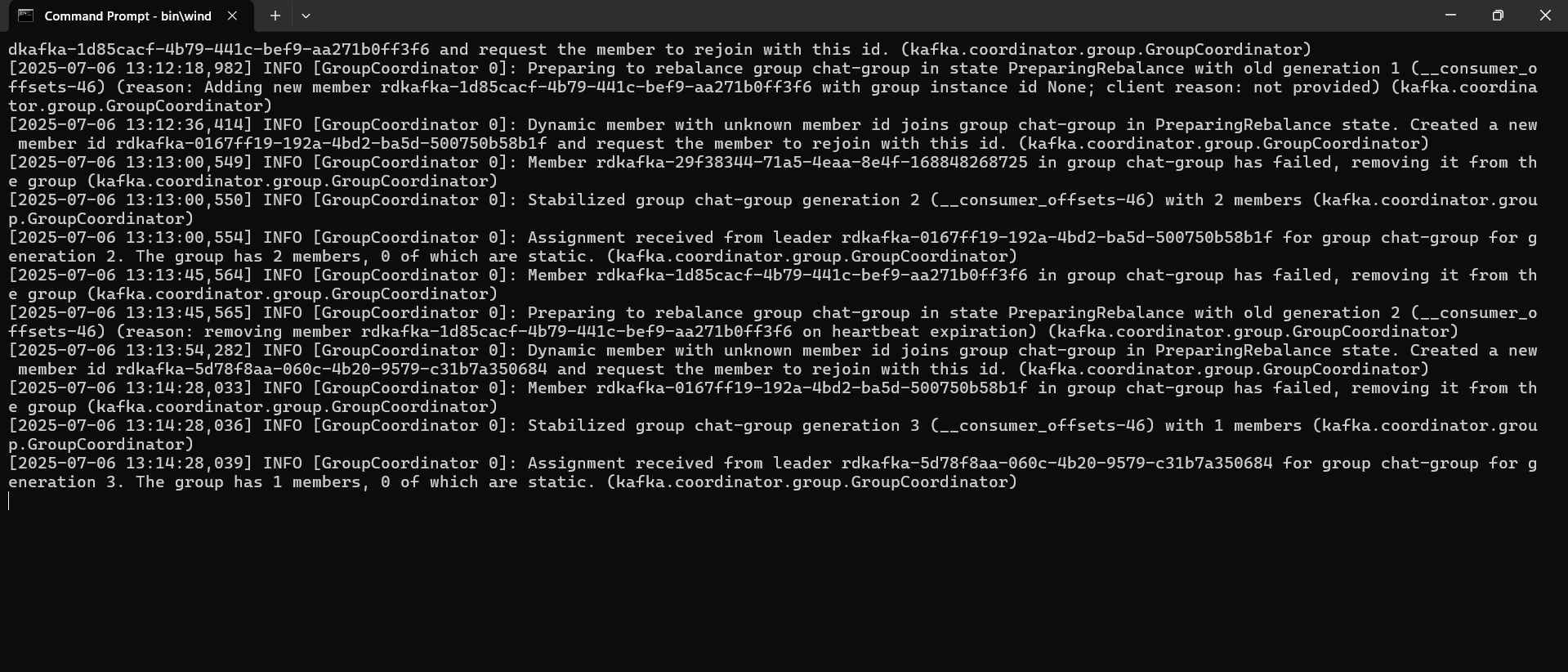
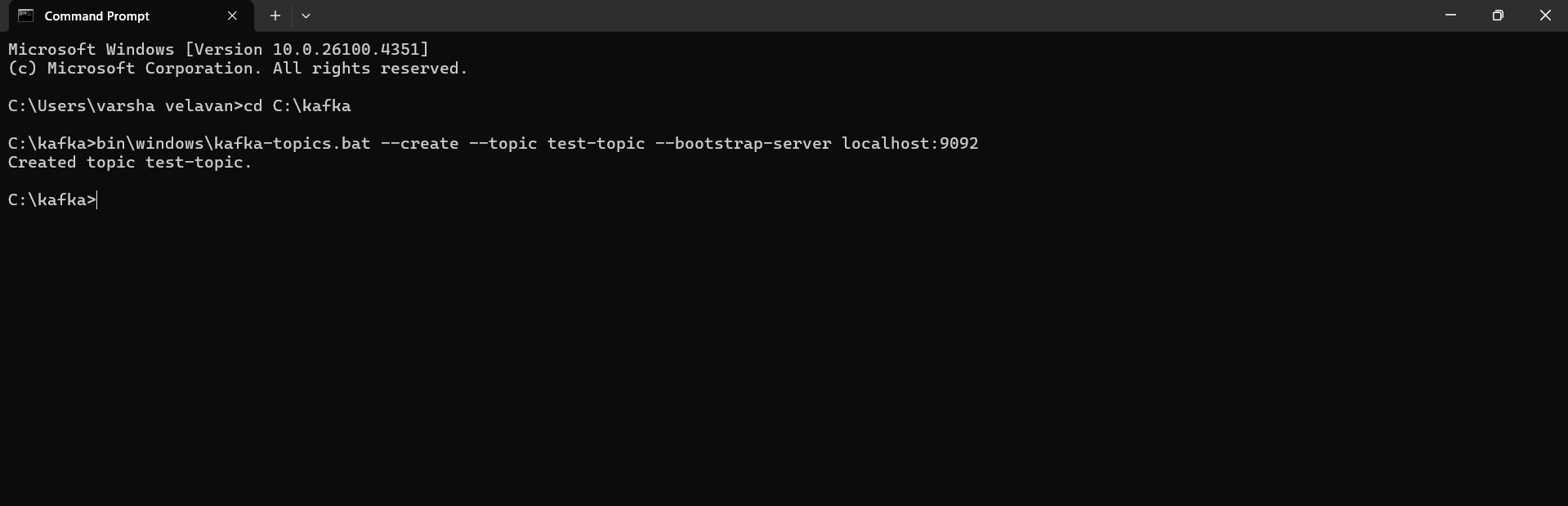
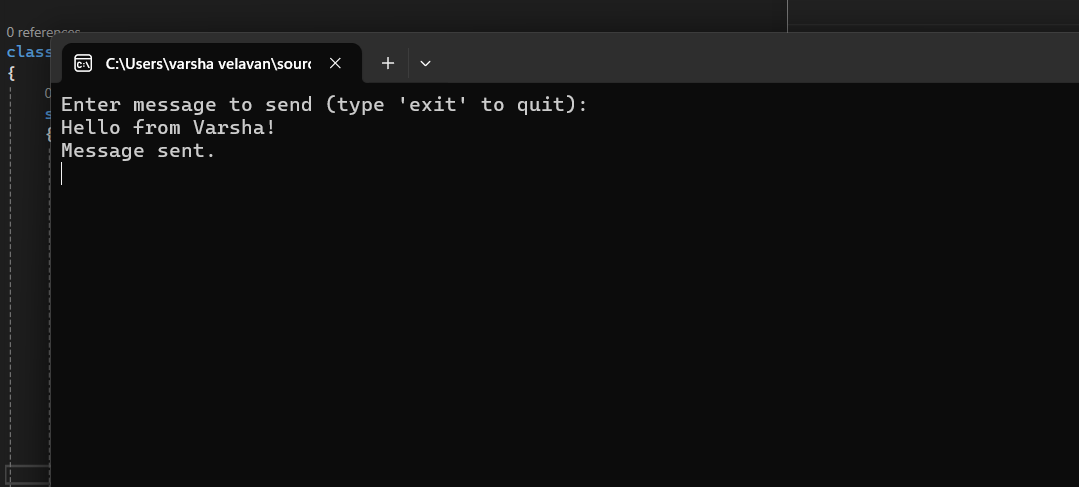
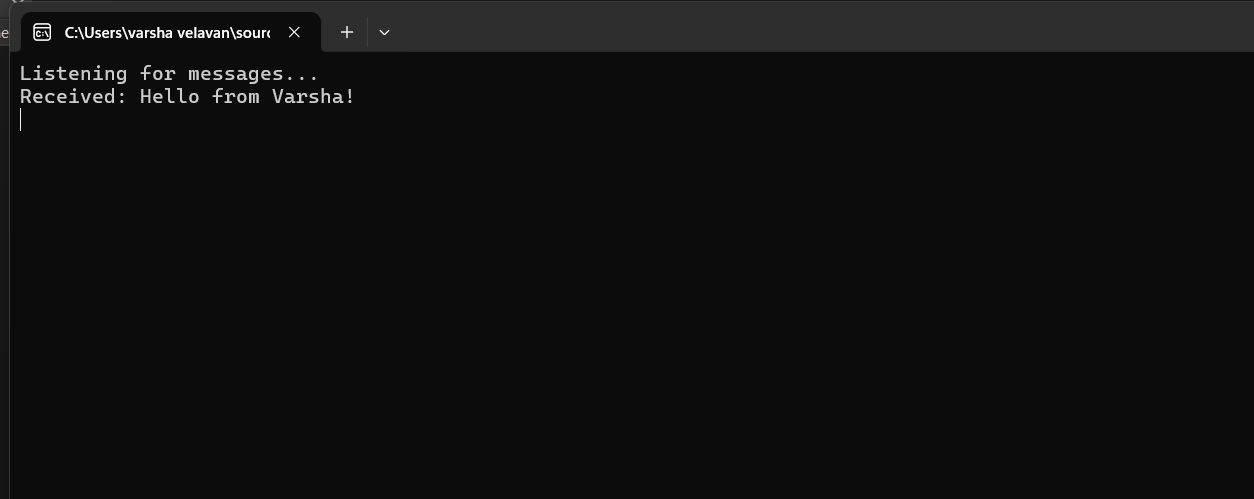
**Program.cs**

using Confluent.Kafka;  
  
class Program  
{  
 static async Task Main(string[] args)  
 {  
 var config = new ProducerConfig { BootstrapServers = "localhost:9092" };  
 using var producer = new ProducerBuilder<Null, string>(config).Build();  
  
 Console.WriteLine("Enter message to send (type 'exit' to quit):");  
  
 while (true)  
 {  
 var input = Console.ReadLine();  
 if (input == "exit") break;  
  
 await producer.ProduceAsync("chat-topic", new Message<Null, string> { Value = input });  
 Console.WriteLine("Message sent.");  
 }  
 }  
}

**KafkaConsumer**

**Program.cs**using Confluent.Kafka;  
  
class Program  
{  
 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(" Listening for messages...");  
  
 while (true)  
 {  
 var msg = consumer.Consume();  
 Console.WriteLine($" Received: {msg.Message.Value}");  
 }  
 }  
}

**Output:  
  
**

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

Open two separate command prompts:  
  
Zookeeper:  
cd C:\kafka\bin\windows  
zookeeper-server-start.bat ..\..\config\zookeeper.properties  
  
Kafka:  
cd C:\kafka\bin\windows  
kafka-server-start.bat ..\..\config\server.properties

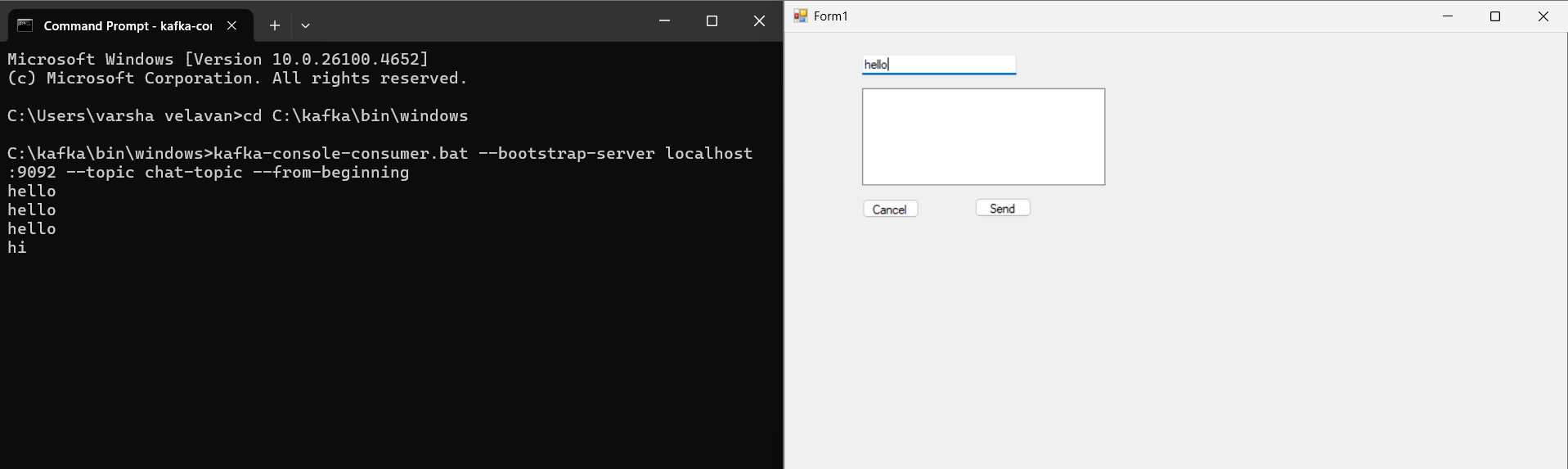
**Step 2: Create Kafka Topic**

In another terminal:  
cd C:\kafka\bin\windows  
kafka-topics.bat --create --topic chat-topic --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1

**Step 3: Design Windows Form**

Add the following controls:  
- TextBox (Name: txtMessage)  
- Button (Name: btnSend, Text: Send)  
- Button (Name: btnCancel, Text: Cancel)  
- ListBox (Name: lstMessages)  
  
Ensure Form Load event is connected to Form1\_Load

**Form1.cs**  
using System;  
using System.Windows.Forms;  
using Confluent.Kafka;  
using System.Threading;  
  
namespace KafkaChatApp  
{  
 public partial class Form1 : Form  
 {  
 private Thread consumerThread;  
  
 public Form1()  
 {  
 InitializeComponent();  
 }  
  
 private void Form1\_Load(object sender, EventArgs e)  
 {  
 consumerThread = new Thread(new ThreadStart(ConsumeMessages));  
 consumerThread.IsBackground = true;  
 consumerThread.Start();  
 }  
  
 private async void btnSend\_Click(object sender, EventArgs e)  
 {  
 var config = new ProducerConfig { BootstrapServers = "localhost:9092" };  
  
 using (var producer = new ProducerBuilder<Null, string>(config).Build())  
 {  
 var message = txtMessage.Text.Trim();  
 if (!string.IsNullOrEmpty(message))  
 {  
 await producer.ProduceAsync("chat-topic", new Message<Null, string> { Value = message });  
 txtMessage.Clear();  
 }  
 }  
 }  
  
 private void btnCancel\_Click(object sender, EventArgs e)  
 {  
 this.Close(); // Close the form  
 }  
  
 private void ConsumeMessages()  
 {  
 var config = new ConsumerConfig  
 {  
 GroupId = "chat-group",  
 BootstrapServers = "localhost:9092",  
 AutoOffsetReset = AutoOffsetReset.Earliest  
 };  
  
 using (var consumer = new ConsumerBuilder<Ignore, string>(config).Build())  
 {  
 consumer.Subscribe("chat-topic");  
  
 while (true)  
 {  
 try  
 {  
 var cr = consumer.Consume();  
  
 if (lstMessages.InvokeRequired)  
 {  
 lstMessages.Invoke((MethodInvoker)delegate {  
 lstMessages.Items.Add($"Received: {cr.Message.Value}");  
 });  
 }  
 else  
 {  
 lstMessages.Items.Add($"Received: {cr.Message.Value}");  
 }  
 }  
 catch (Exception ex)  
 {  
 Console.WriteLine("Error consuming: " + ex.Message);  
 }  
 }  
 }  
 }  
  
 private void txtMessage\_TextChanged(object sender, EventArgs e)  
 {  
  
 }  
  
 private void lstMessages\_SelectedIndexChanged(object sender, EventArgs e)  
 {  
  
 }  
 }  
}

**Output:  
**