**WEEK 5**

**MICROSERVICES ARCHITECTURE USING ASP.NET CORE WEB API**

**EXERCISE 1: KAFKA INTEGRATION WITH C#**

**Objective**

To create a Kafka-based chat application using C# Windows Forms that allows sending messages to a Kafka topic and consuming them via a console-based Kafka consumer.

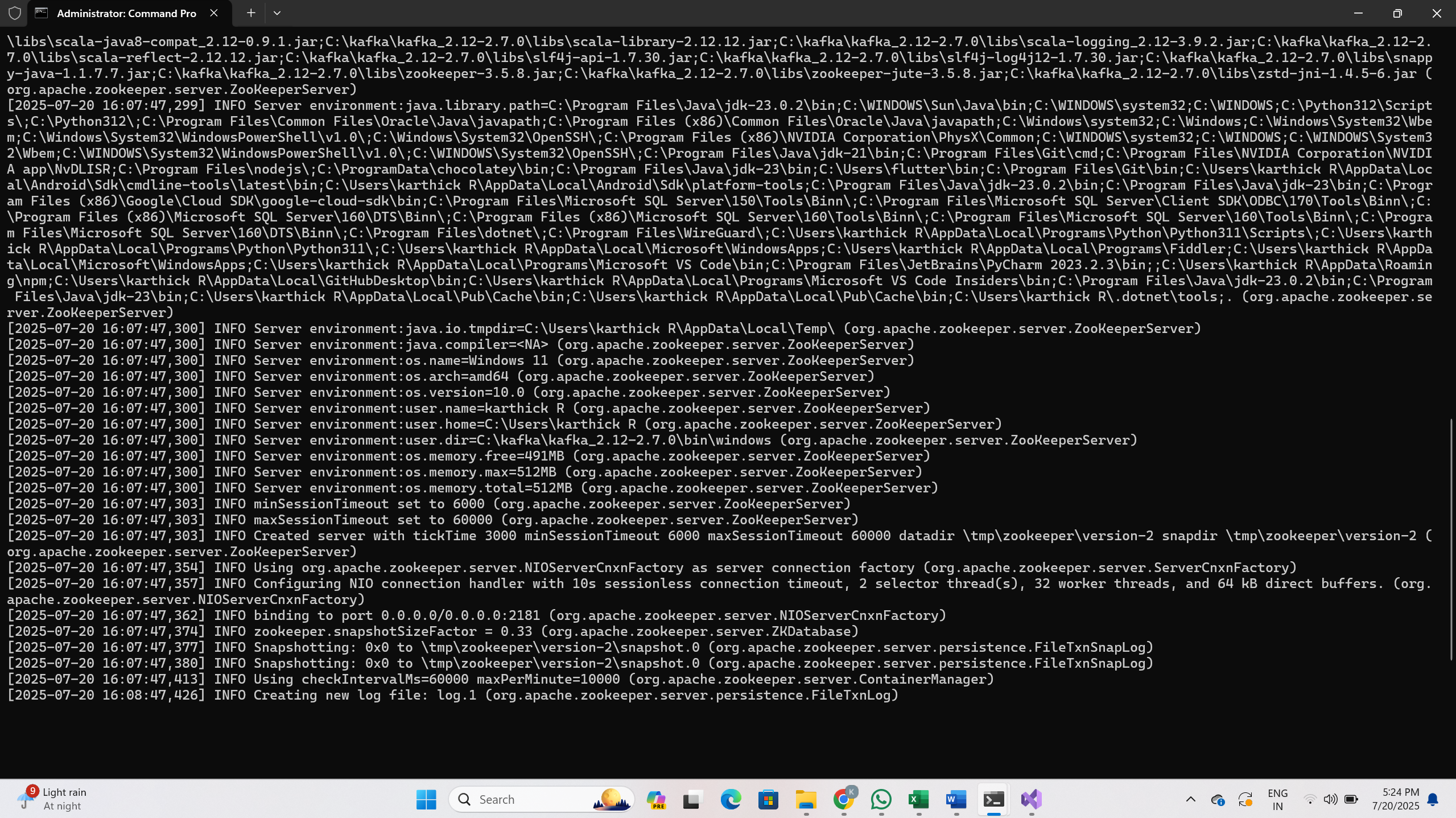
**Setup & Installation**

1. Kafka and Zookeeper Installation

* Extracted Kafka from kafka\_2.12-2.7.0.zip.

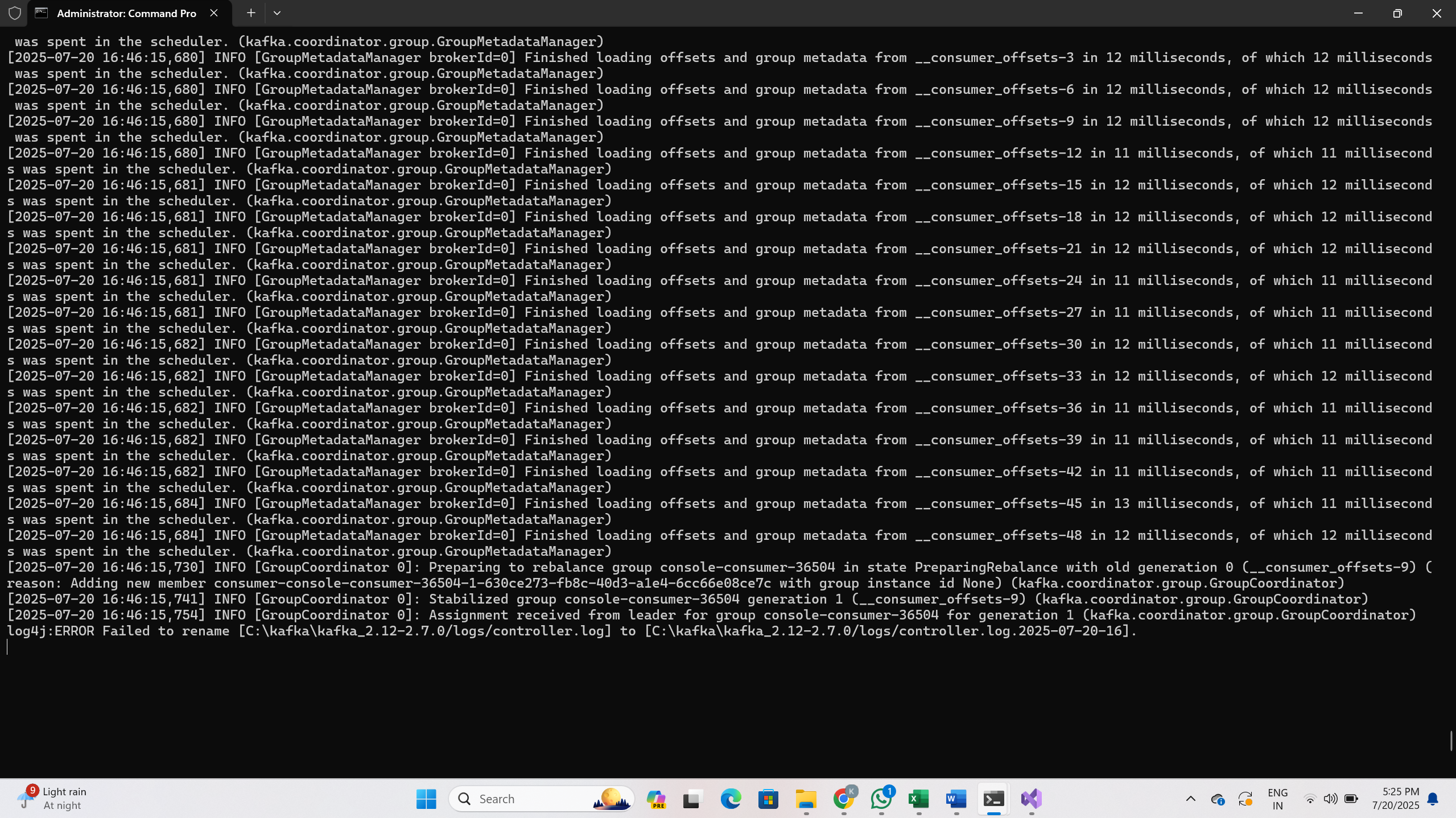
Started Zookeeper using:

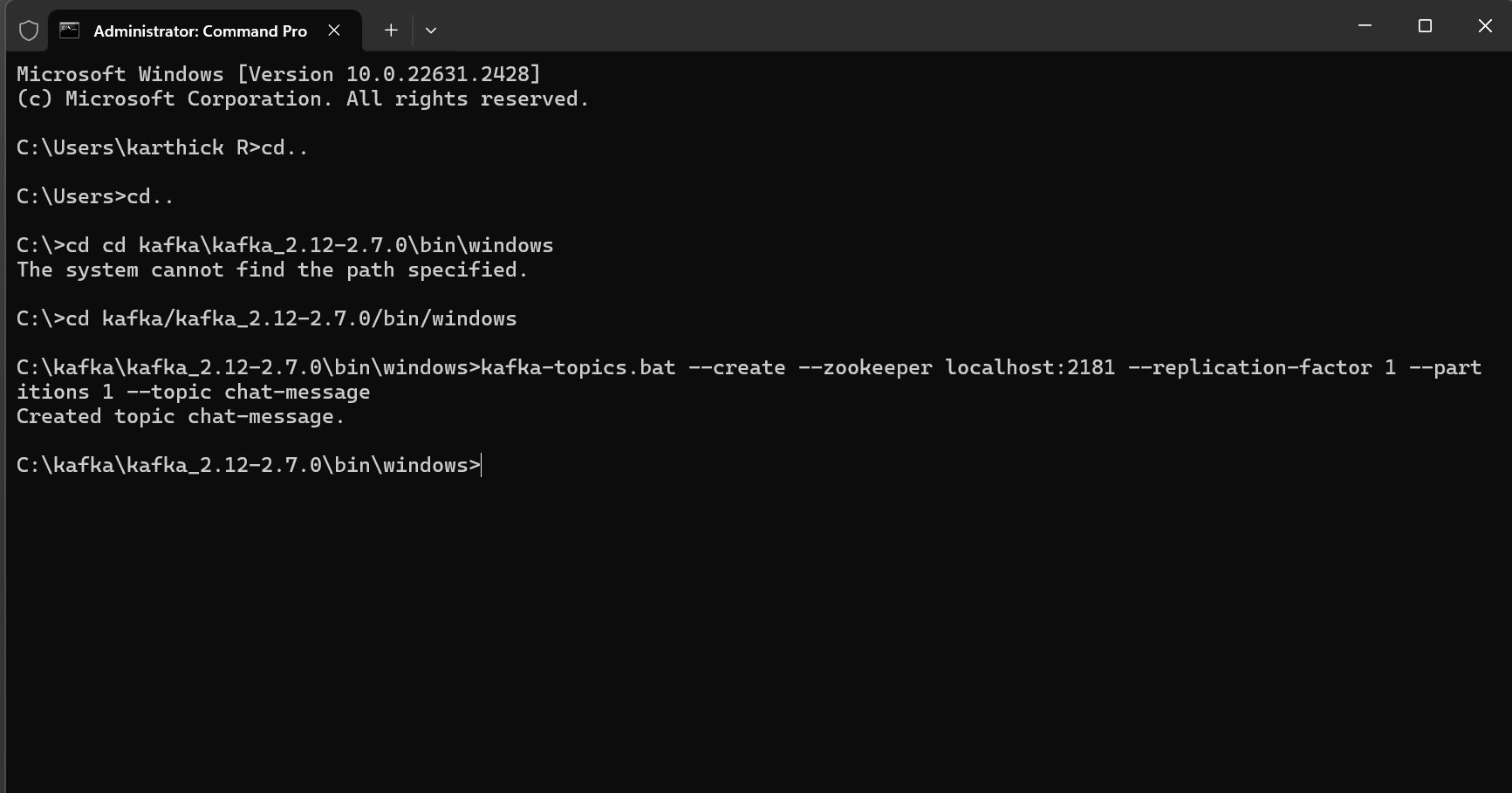
.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties

****

Started Kafka broker using:

.\bin\windows\kafka-server-start.bat .\config\server.properties

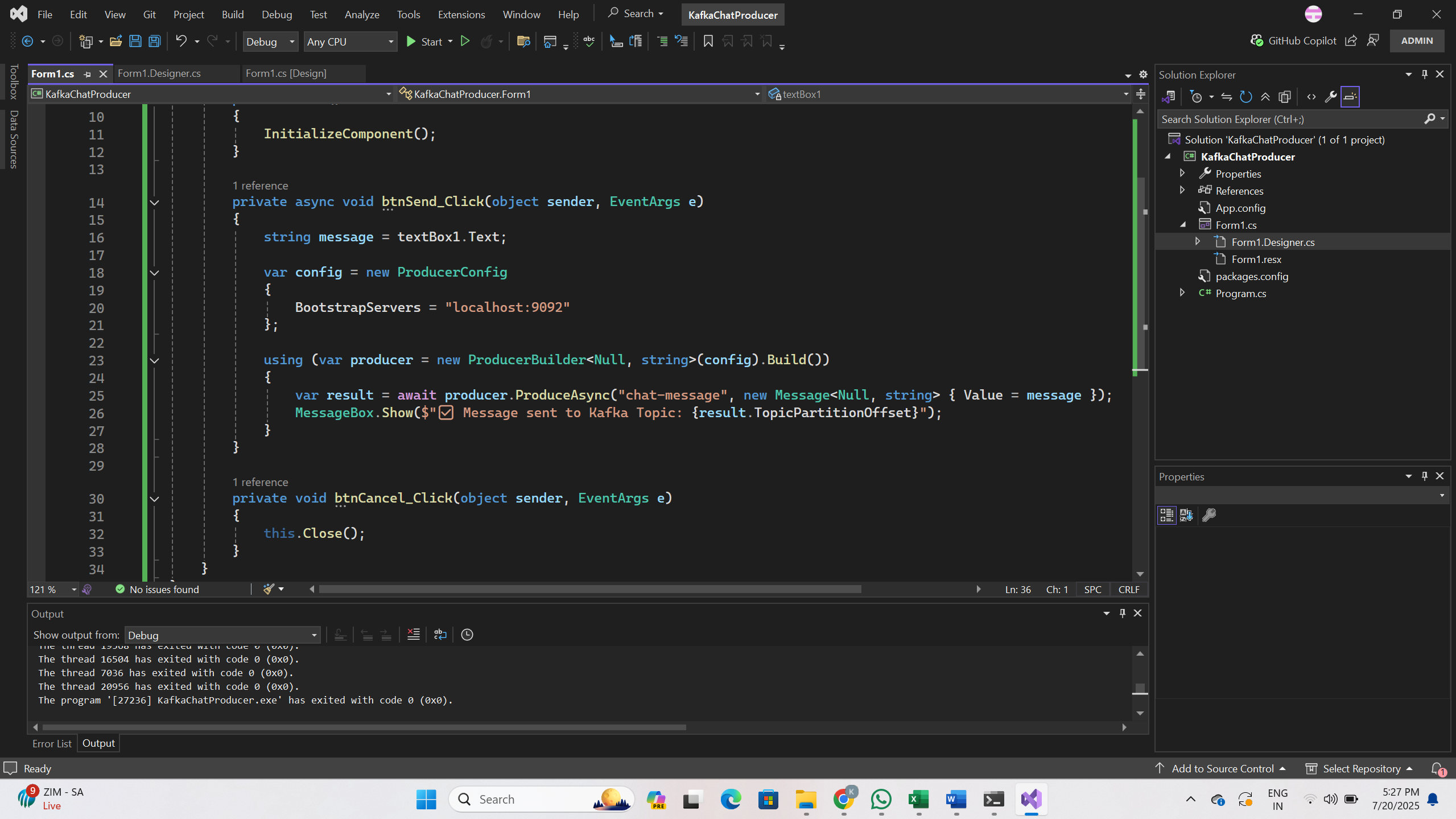
****

Created topic chat-message:  


**Project Setup**

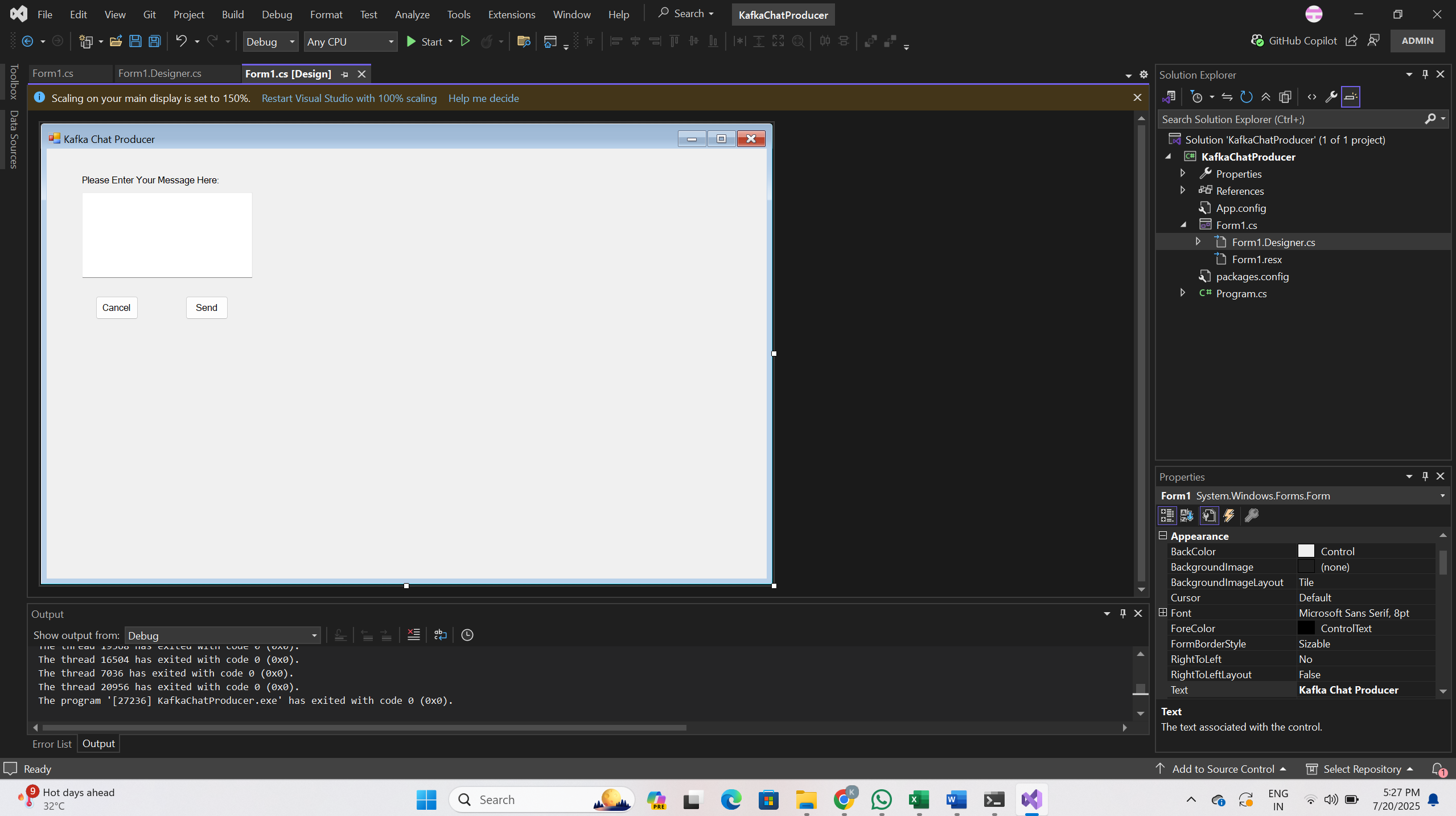
2. Windows Forms Application in Visual Studio

* Created project: KafkaChatProducer
* Type: Windows Forms App (.NET Framework)



3. Designer Elements

* Added a Label with text: “Please Enter Your Message Here:”
* Added a Multiline TextBox: textBox1
* Added two Buttons:
  + btnSend → Send
  + btnCancel → Cancel

****

**Kafka Producer Code**

**Form1.cs**

using Confluent.Kafka;

using System;

using System.Windows.Forms;

namespace KafkaChatProducer

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private async void btnSend\_Click(object sender, EventArgs e)

{

string message = textBox1.Text;

var config = new ProducerConfig

{

BootstrapServers = "localhost:9092"

};

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

{

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

MessageBox.Show($"Message sent to Kafka Topic: {result.TopicPartitionOffset}");

}

}

private void btnCancel\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

**Form1.Designer.cs Code Snippet**

Key UI Controls:

* Label: label1
* TextBox: textBox1
* Buttons: btnSend, btnCancel

namespace KafkaChatProducer

{

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()

{

this.label1 = new System.Windows.Forms.Label();

this.textBox1 = new System.Windows.Forms.TextBox();

this.btnSend = new System.Windows.Forms.Button();

this.btnCancel = new System.Windows.Forms.Button();

this.SuspendLayout();

//

// label1

//

this.label1.AutoSize = true;

this.label1.Location = new System.Drawing.Point(58, 45);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(250, 20);

this.label1.TabIndex = 0;

this.label1.Text = "Please Enter Your Message Here:"

//

// textBox1

//

this.textBox1.Location = new System.Drawing.Point(62, 77);

this.textBox1.Multiline = true;

this.textBox1.Name = "textBox1";

this.textBox1.Size = new System.Drawing.Size(300, 150);

this.textBox1.TabIndex = 1;

//

// btnSend

//

this.btnSend.Location = new System.Drawing.Point(244, 259);

this.btnSend.Name = "btnSend";

this.btnSend.Size = new System.Drawing.Size(75, 41);

this.btnSend.TabIndex = 2;

this.btnSend.Text = "Send";

this.btnSend.UseVisualStyleBackColor = true;

this.btnSend.Click += new System.EventHandler(this.btnSend\_Click);

//

// btnCancel

//

this.btnCancel.Location = new System.Drawing.Point(86, 259);

this.btnCancel.Name = "btnCancel";

this.btnCancel.Size = new System.Drawing.Size(75, 41);

this.btnCancel.TabIndex = 3;

this.btnCancel.Text = "Cancel";

this.btnCancel.UseVisualStyleBackColor = true;

this.btnCancel.Click += new System.EventHandler(this.btnCancel\_Click);

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(9F, 20F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(1265, 755);

this.Controls.Add(this.btnCancel);

this.Controls.Add(this.btnSend);

this.Controls.Add(this.textBox1);

this.Controls.Add(this.label1);

this.Name = "Form1";

this.Text = "Kafka Chat Producer";

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.Label label1;

private System.Windows.Forms.TextBox textBox1;

private System.Windows.Forms.Button btnSend;

private System.Windows.Forms.Button btnCancel;

}

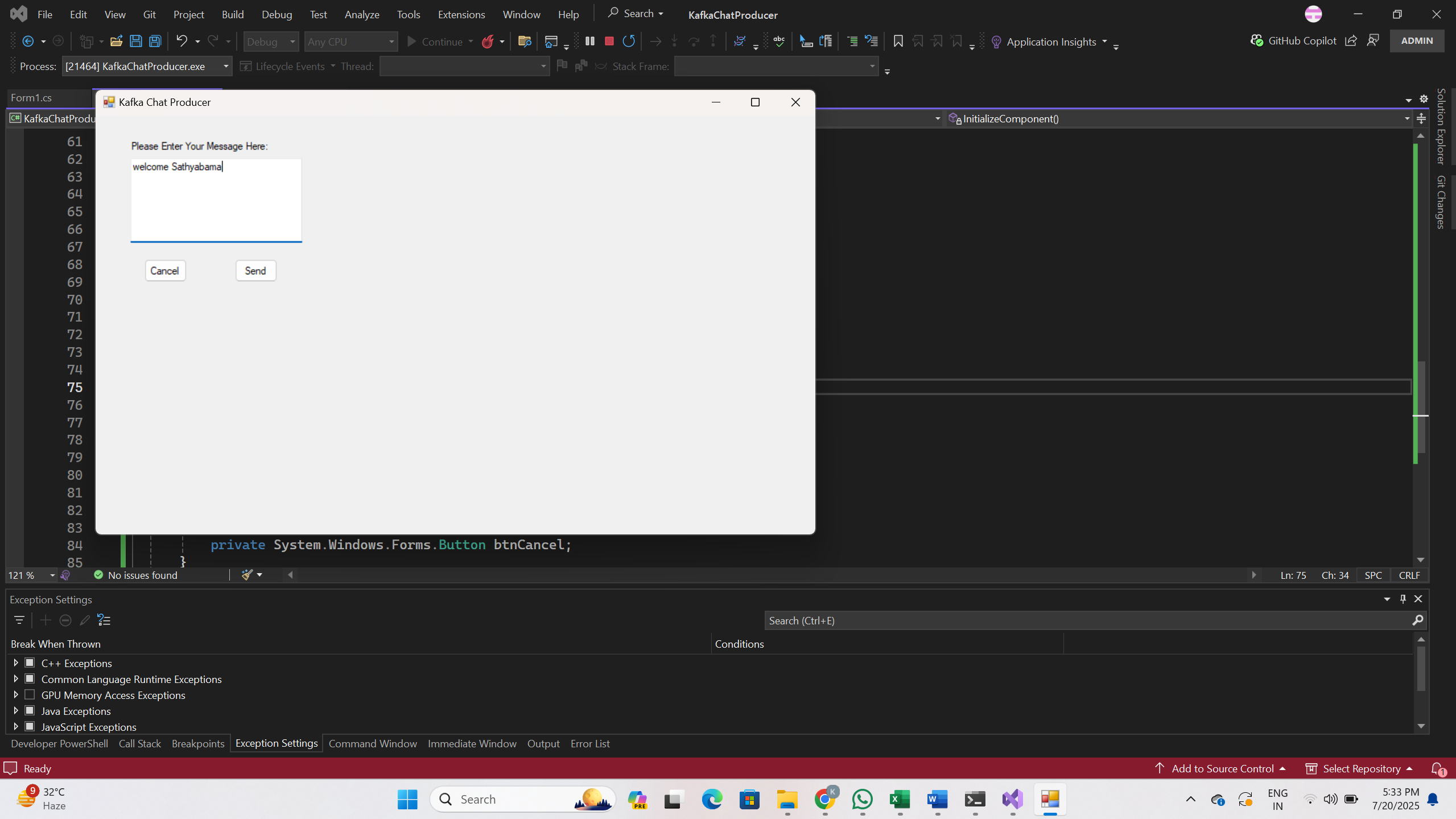
}

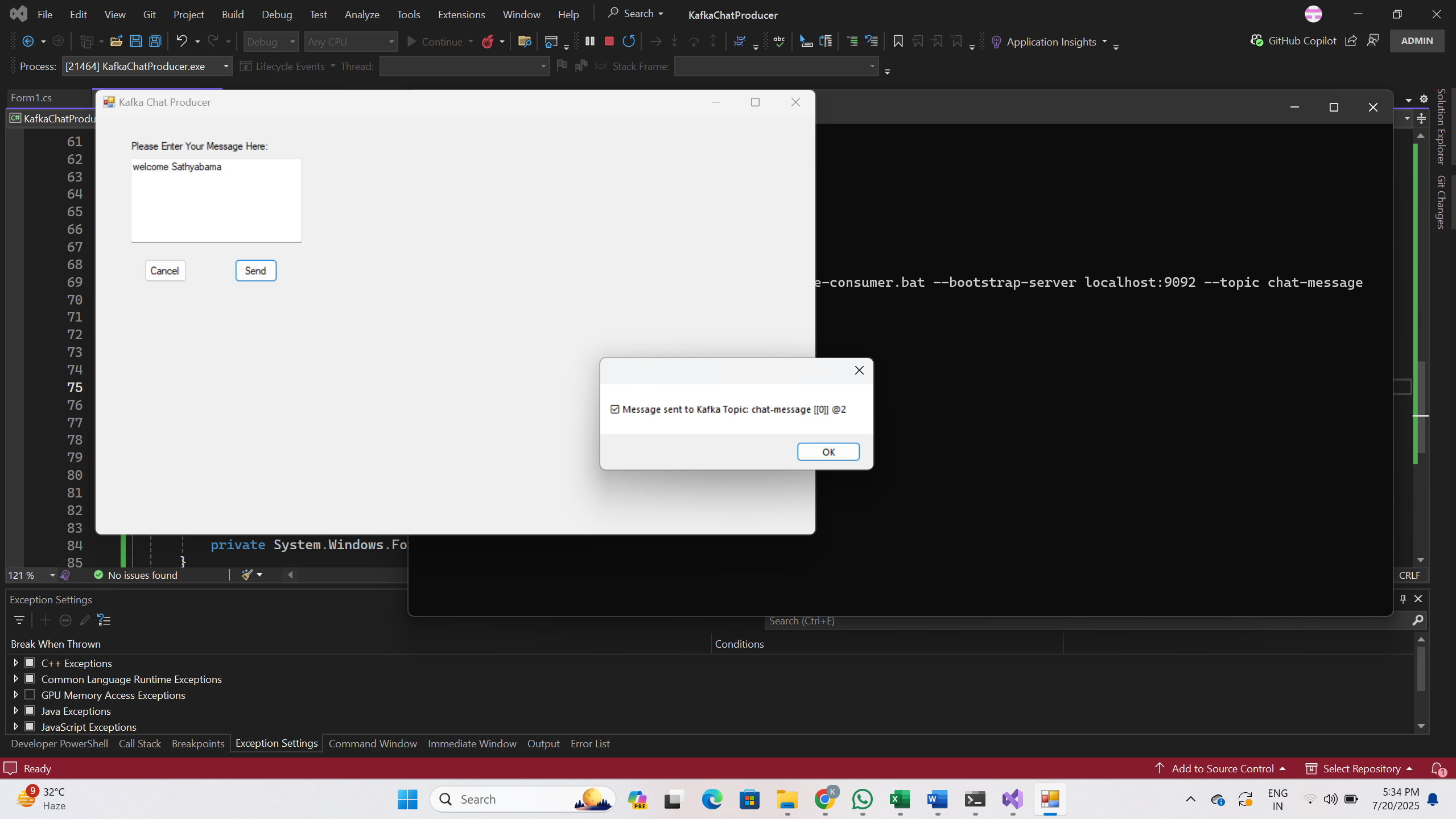
**Kafka Consumer**

Kafka Consumer Started in CMD

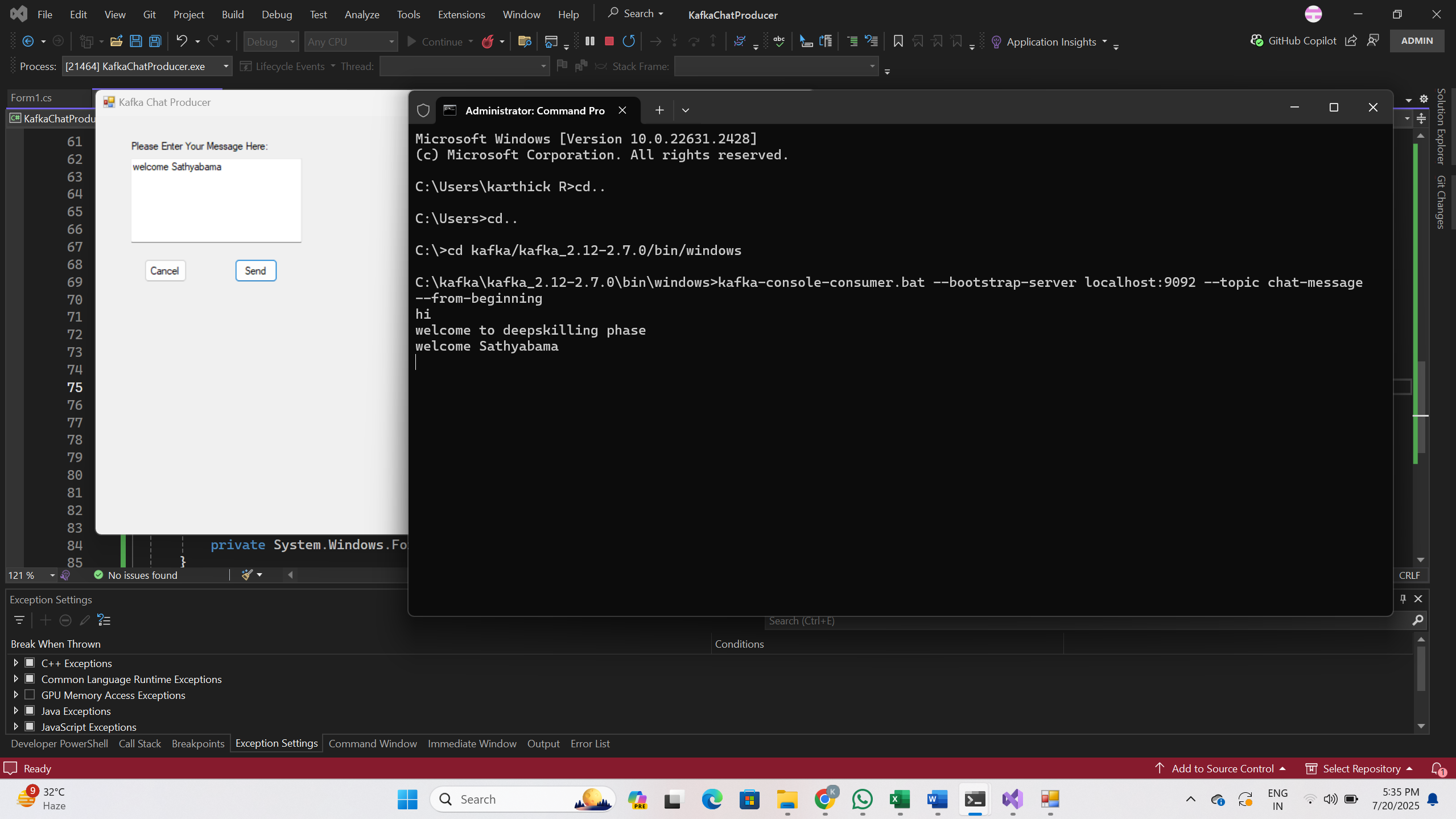
kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic chat-message --from-beginning

This terminal listens to all messages sent by the producer.



****

**OUTPUT:**

****

**MICROSERVICES JWT**

**EXERCISE 1: IMPLEMENT JWT AUTHENTICATION IN ASP.NET CORE WEB API**

Program.cs

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

// Add JWT Authentication

builder.Services.AddAuthentication("Bearer")

.AddJwtBearer("Bearer", 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"]))

};

});

// Add authorization

builder.Services.AddAuthorization();

// Add controllers

builder.Services.AddControllers();

// Add Swagger and configure JWT Auth support

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(options =>

{

// Swagger Authorization configuration

options.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

Name = "Authorization",

Type = SecuritySchemeType.ApiKey,

Scheme = "Bearer",

BearerFormat = "JWT",

In = ParameterLocation.Header,

Description = "Enter 'Bearer <your JWT token here>'"

});

options.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

new string[] {}

}

});

});

var app = builder.Build();

// Middleware pipeline

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthentication(); // ✅ Important

app.UseAuthorization();

app.MapControllers();

app.Run();

AuthController.cs

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

{

private readonly IConfiguration \_config;

public AuthController(IConfiguration config)

{

\_config = config;

}

[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)

{

// Dummy check for testing

return model.Username == "admin" && model.Password == "admin123";

}

private string GenerateJwtToken(string username)

{

var claims = new[]

{

new Claim(ClaimTypes.Name, username)

};

var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(\_config["Jwt:Key"]));

var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

var token = new JwtSecurityToken(

issuer: \_config["Jwt:Issuer"],

audience: \_config["Jwt:Audience"],

claims: claims,

expires: DateTime.Now.AddMinutes(Convert.ToDouble(\_config["Jwt:DurationInMinutes"])),

signingCredentials: creds

);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

SecureController.cs

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace JwtAuthDemo.Controllers

{

[ApiController]

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

public class SecureController : ControllerBase

{

[HttpGet("data")]

[Authorize] // ✅ Protected endpoint

public IActionResult GetSecretData()

{

return Ok("🔐 You are authorized to see this protected data!");

}

}

}

Appsettings.json

{

"Jwt": {

"Key": "ThisIsAStrongSecretKeyForJwtToken123456!",

"Issuer": "MyAuthServer",

"Audience": "MyApiUsers",

"DurationInMinutes": 60

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

LoginModel.cs

public class LoginModel

{

public string Username { get; set; }

public string Password { get; set; }

}

OUTPUT:

