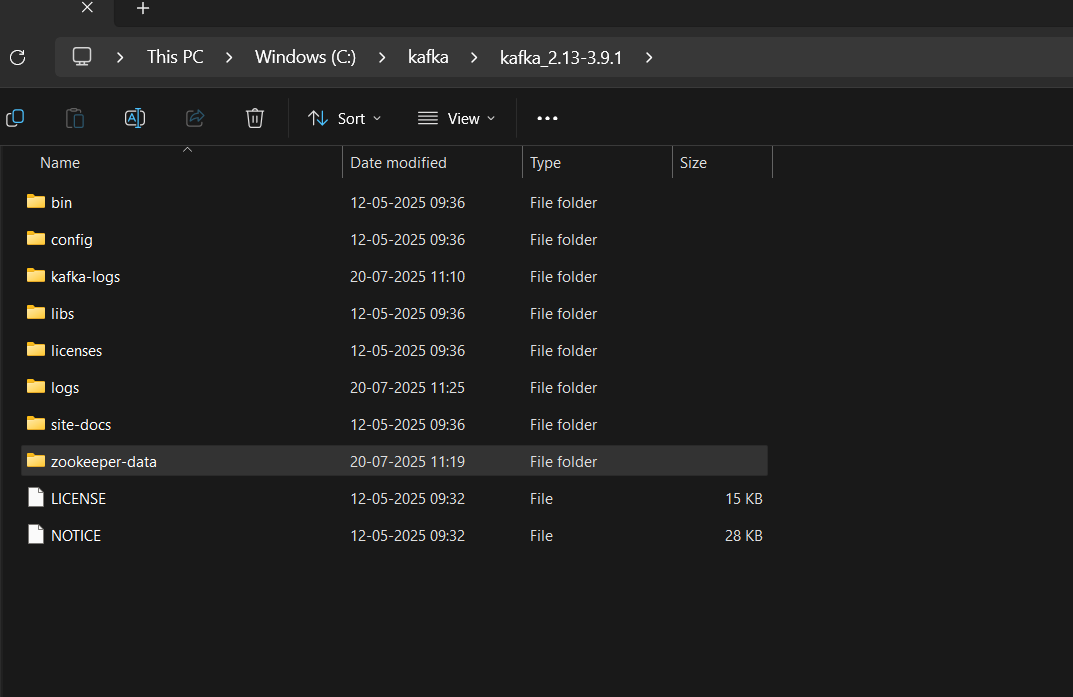
# WEEK – 5 ( WEB API AND MICROSERVICES)

1. **CREATE A CHAT APPLICATION WHICH USES KAFKA AS A STREAMING PLATFORM AND CONSUME THE CHAT MESSAGES IN THE COMMAND PROMPT**.

**Step 1 : install kafka and save it in a folder**



**Step 2:**

**Configure the below two files**

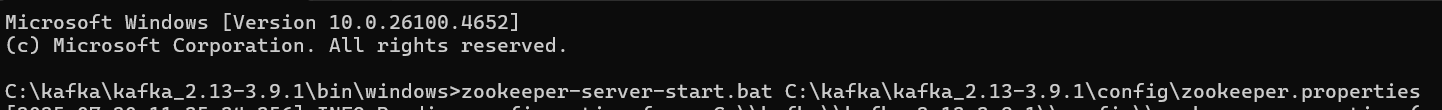
1. **Server.properties**
2. **Zookeeper.properties**

**Path : C: kafka\config\ zookeeper**

*Note : we need to create the folders server-logs and zookeeper-data in kafka main folder for storing the logs of messages and texts*

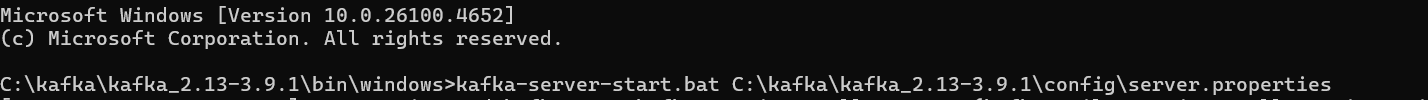
**Step 3:**

**Initiate the zookeeper properties by below command**

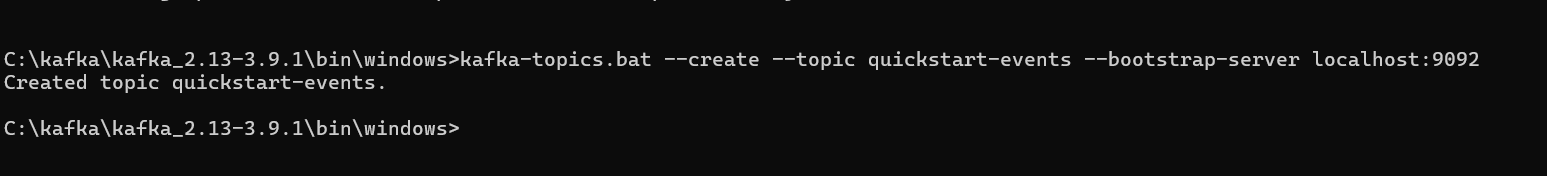
****

**Step 4:**

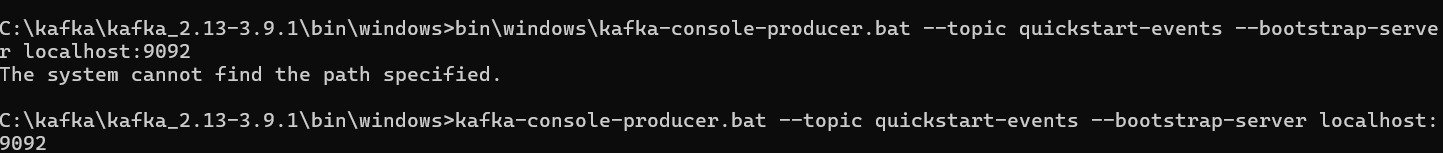
**Initiate the server by the below command**



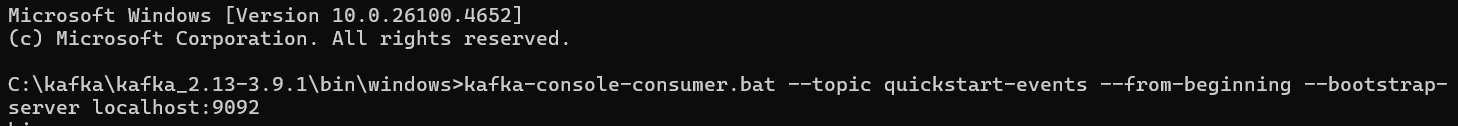
**Step 5 : create the topics for intiating chats**



Step 6 : create and initiate the producer programmer

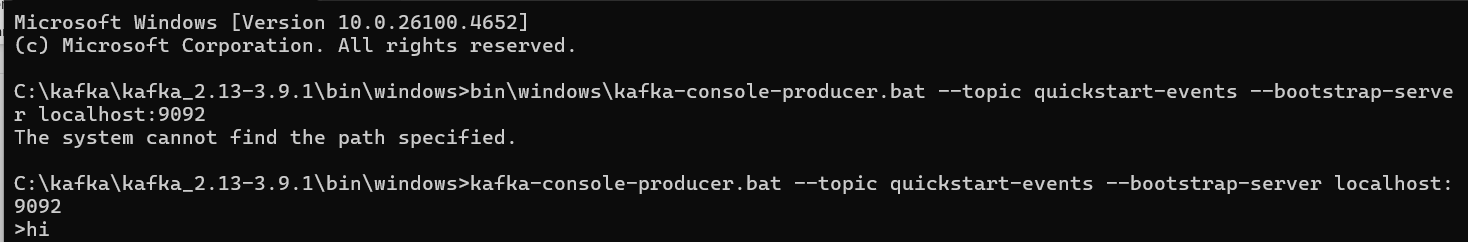


Step 7 : create and initiate the consumer program

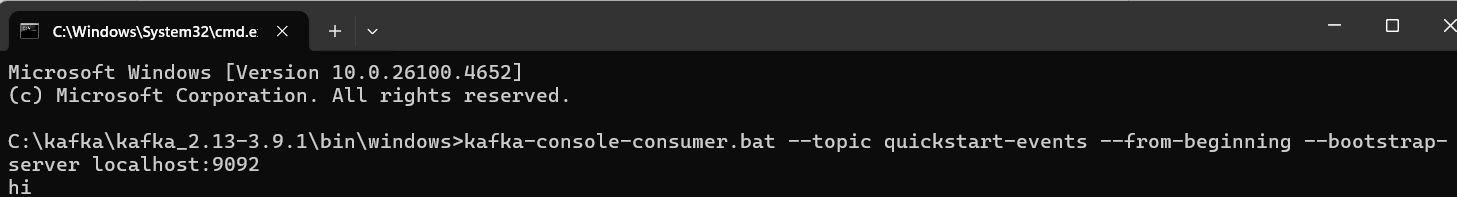


Step 8 : the output of chat using kafka and producing and consuming messages through command prompt

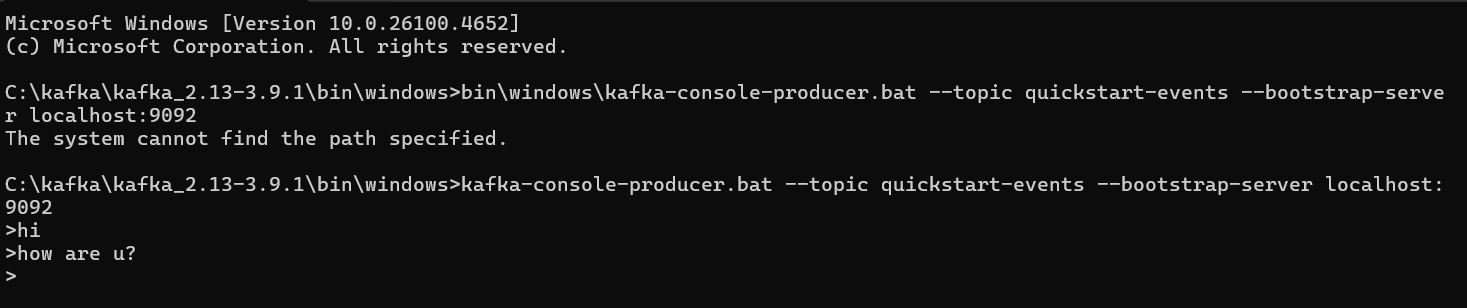
Producer :



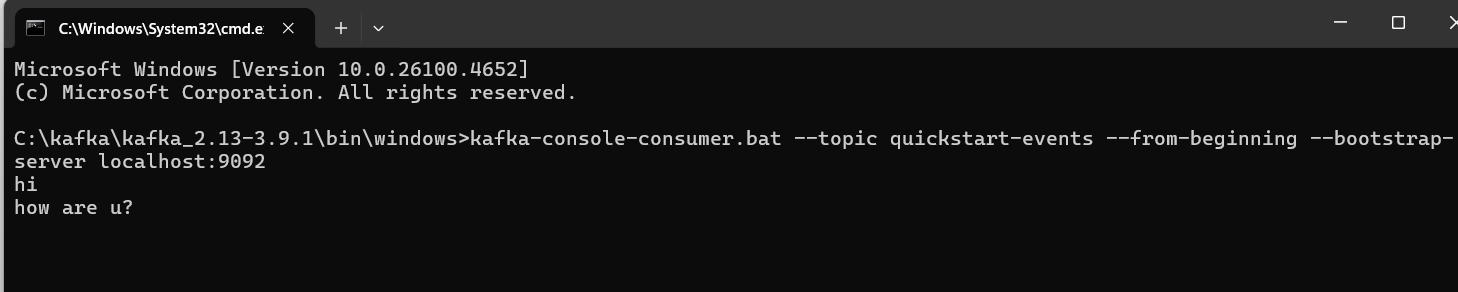
Consumer :



Producer :



Consumer :



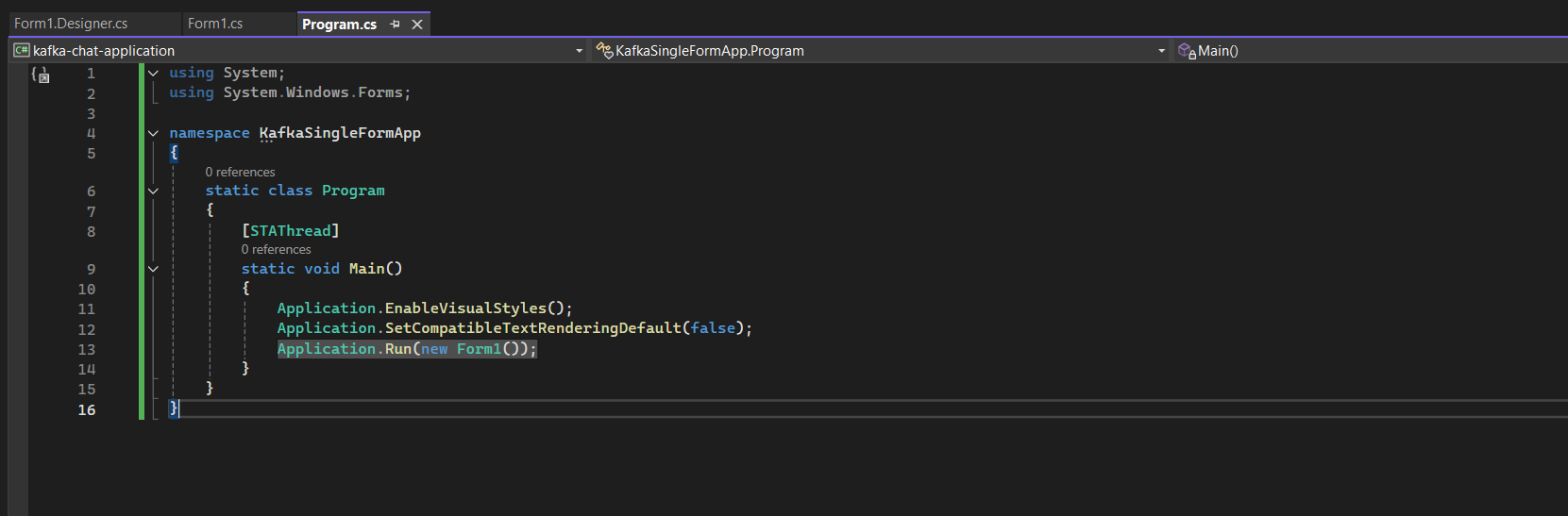
**2) CREATE A CHAT APPLICATION USING C# WINDOWS APPLICATION USING KAFKA AND CONSUME THE MESSAGE IN DIFFERENT CLIENT APPLICATIONS.**

**First create an Web Windows App in Visual studio code**

**Then name the project kafka-chat-application**

**Create a form**

**Form1.cs**

****

**Form.designer.cs**

*namespace KafkaSingleFormApp*

*{*

*partial class Form1*

*{*

*private System.ComponentModel.IContainer components = null;*

*protected override void Dispose(bool disposing)*

*{*

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

*{*

*components.Dispose();*

*}*

*base.Dispose(disposing);*

*}*

*private void InitializeComponent()*

*{*

*this.txtMessage = new System.Windows.Forms.TextBox();*

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

*this.lstMessages = new System.Windows.Forms.ListBox();*

*this.txtUsername = new System.Windows.Forms.TextBox();*

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

*this.label2 = new System.Windows.Forms.Label();*

*this.btnStartConsumer = new System.Windows.Forms.Button();*

*this.SuspendLayout();*

*// txtMessage*

*this.txtMessage.Location = new System.Drawing.Point(85, 41);*

*this.txtMessage.Multiline = true;*

*this.txtMessage.Name = "txtMessage";*

*this.txtMessage.Size = new System.Drawing.Size(300, 80);*

*this.txtMessage.TabIndex = 1;*

*// btnSend*

*this.btnSend.Location = new System.Drawing.Point(391, 41);*

*this.btnSend.Name = "btnSend";*

*this.btnSend.Size = new System.Drawing.Size(75, 80);*

*this.btnSend.TabIndex = 2;*

*this.btnSend.Text = "Send";*

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

*// lstMessages*

*this.lstMessages.FormattingEnabled = true;*

*this.lstMessages.ItemHeight = 15;*

*this.lstMessages.Location = new System.Drawing.Point(12, 127);*

*this.lstMessages.Name = "lstMessages";*

*this.lstMessages.Size = new System.Drawing.Size(454, 214);*

*this.lstMessages.TabIndex = 3;*

*// txtUsername*

*this.txtUsername.Location = new System.Drawing.Point(85, 12);*

*this.txtUsername.Name = "txtUsername";*

*this.txtUsername.Size = new System.Drawing.Size(300, 23);*

*this.txtUsername.TabIndex = 0;*

*// label1*

*this.label1.AutoSize = true;*

*this.label1.Location = new System.Drawing.Point(12, 15);*

*this.label1.Name = "label1";*

*this.label1.Size = new System.Drawing.Size(63, 15);*

*this.label1.Text = "Username:";*

*// label2*

*this.label2.AutoSize = true;*

*this.label2.Location = new System.Drawing.Point(12, 44);*

*this.label2.Name = "label2";*

*this.label2.Size = new System.Drawing.Size(56, 15);*

*this.label2.Text = "Message:";*

*// btnStartConsumer*

*this.btnStartConsumer.Location = new System.Drawing.Point(391, 12);*

*this.btnStartConsumer.Name = "btnStartConsumer";*

*this.btnStartConsumer.Size = new System.Drawing.Size(75, 23);*

*this.btnStartConsumer.TabIndex = 4;*

*this.btnStartConsumer.Text = "Start Consumer";*

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

*// Form1*

*this.ClientSize = new System.Drawing.Size(478, 353);*

*this.Controls.Add(this.btnStartConsumer);*

*this.Controls.Add(this.label2);*

*this.Controls.Add(this.label1);*

*this.Controls.Add(this.txtUsername);*

*this.Controls.Add(this.lstMessages);*

*this.Controls.Add(this.btnSend);*

*this.Controls.Add(this.txtMessage);*

*this.Name = "Form1";*

*this.Text = "Kafka Producer/Consumer";*

*this.ResumeLayout(false);*

*this.PerformLayout();*

*}*

*private System.Windows.Forms.TextBox txtMessage;*

*private System.Windows.Forms.Button btnSend;*

*private System.Windows.Forms.ListBox lstMessages;*

*private System.Windows.Forms.TextBox txtUsername;*

*private System.Windows.Forms.Label label1;*

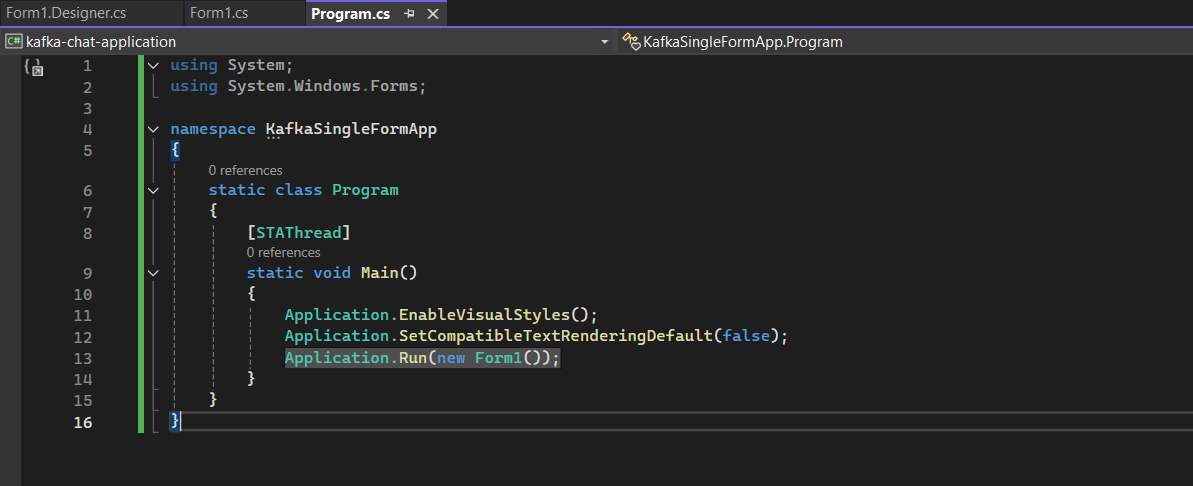
*private System.Windows.Forms.Label label2;*

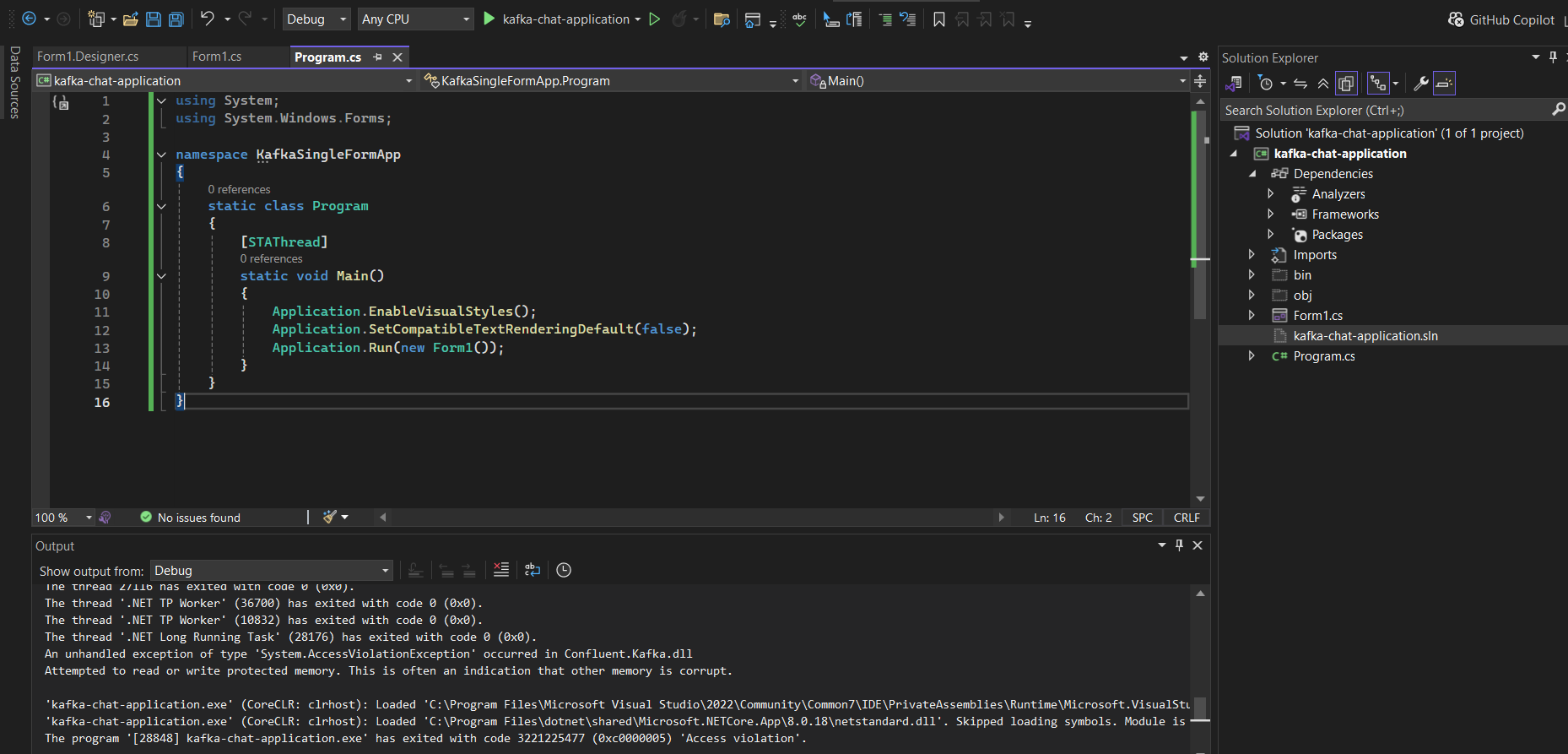
*private System.Windows.Forms.Button btnStartConsumer;*

*}*

*}*

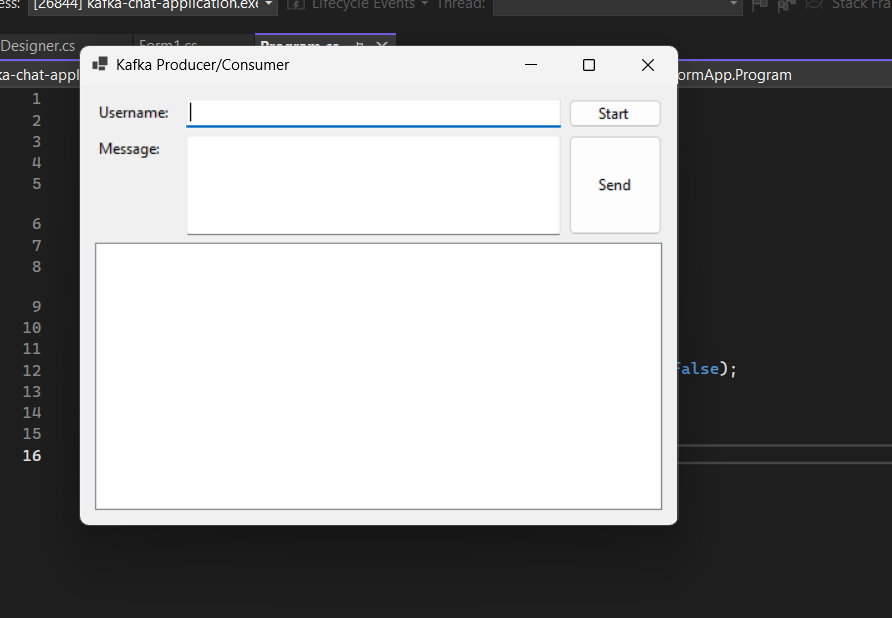
**Program.cs**

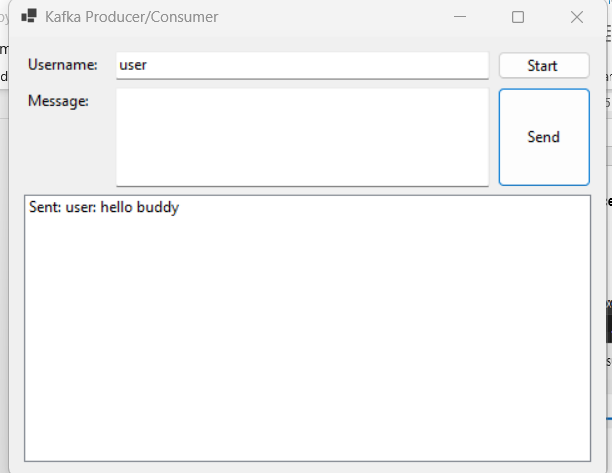
****

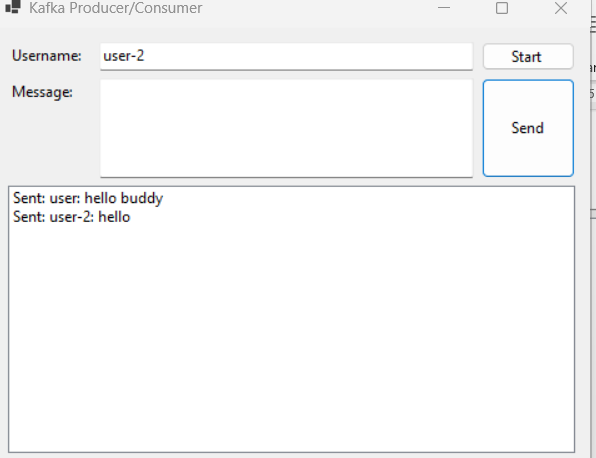
****

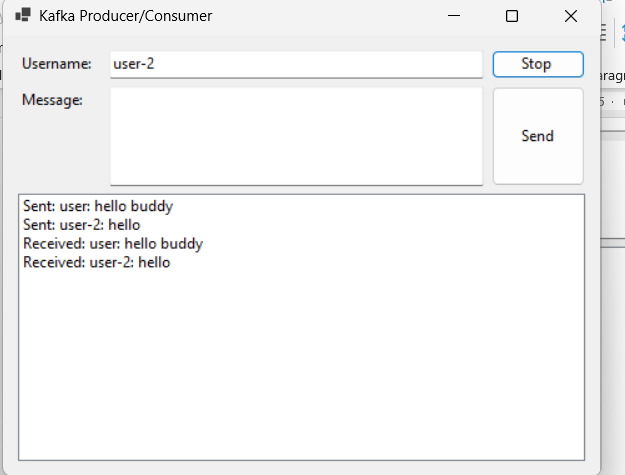
**Above Fig shows the code space for the chat application**

**Output :**

****

****

****

****

Microservices

1. Implement JWT Authentication in ASP.NET Core Web API

* At beginning let us define jwt in appsettings.json

{

  "Jwt": {

    "Key": "KfA1w8pAZA+M9T/8zXnyN/zM6R8xLDbdFZlP2JepuD0=",

    "Issuer": "MyAuthServer",

    "Audience": "MyApiUsers"

  },

  "Logging": {

    "LogLevel": {

      "Default": "Information",

      "Microsoft.AspNetCore": "Warning"

    }

  },

  "AllowedHosts": "\*"

}

* In the next step,Create the jwtBearer in program.cs file

Program.cs

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"]!))

        };

        options.Events = new JwtBearerEvents

        {

            OnAuthenticationFailed = context =>

            {

                if (context.Exception is SecurityTokenExpiredException)

                {

                    context.Response.Headers.Add("Token-Expired", "true");

                }

                return Task.CompletedTask;

            }

        };

    });

builder.Services.AddAuthorization();

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

* Now create the Controller

1.For AdminController.cs

2.AuthController.cs

3.SecureController.cs

AdminController.cs

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace JwtAuthApi.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    public class AdminController : ControllerBase

    {

        [HttpGet("dashboard")]

        [Authorize(Roles = "Admin")]

        public IActionResult GetAdminDashboard()

        {

            return Ok("Welcome to the admin dashboard.");

        }

    }

}

AuthController.cs

using JwtAuthApi.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace JwtAuthApi.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    public class AuthController : ControllerBase

    {

        private readonly IConfiguration \_config;

        public AuthController(IConfiguration config)

        {

            \_config = config;

        }

        [HttpPost("login")]

        public IActionResult Login([FromBody] User user)

        {

            if (user.Username == "admin" && user.Password == "admin123")

            {

                var token = GenerateJwtToken(user.Username);

                return Ok(new { token });

            }

            return Unauthorized();

        }

        private string GenerateJwtToken(string username)

        {

            var securityKey = new SymmetricSecurityKey(

                Encoding.UTF8.GetBytes(\_config["Jwt:Key"]!));

            var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);

            var claims = new[]

            {

                new Claim(ClaimTypes.Name, username),

                new Claim(ClaimTypes.Role, "Admin") // Role-based authorization

            };

            var token = new JwtSecurityToken(

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

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

                claims: claims,

                expires: DateTime.Now.AddMinutes(60),

                signingCredentials: credentials

            );

            return new JwtSecurityTokenHandler().WriteToken(token);

        }

    }

}

Secure Controller.cs

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace JwtAuthApi.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    public class SecureController : ControllerBase

    {

        [HttpGet("data")]

        [Authorize]

        public IActionResult GetSecureData()

        {

            return Ok("This is protected data.");

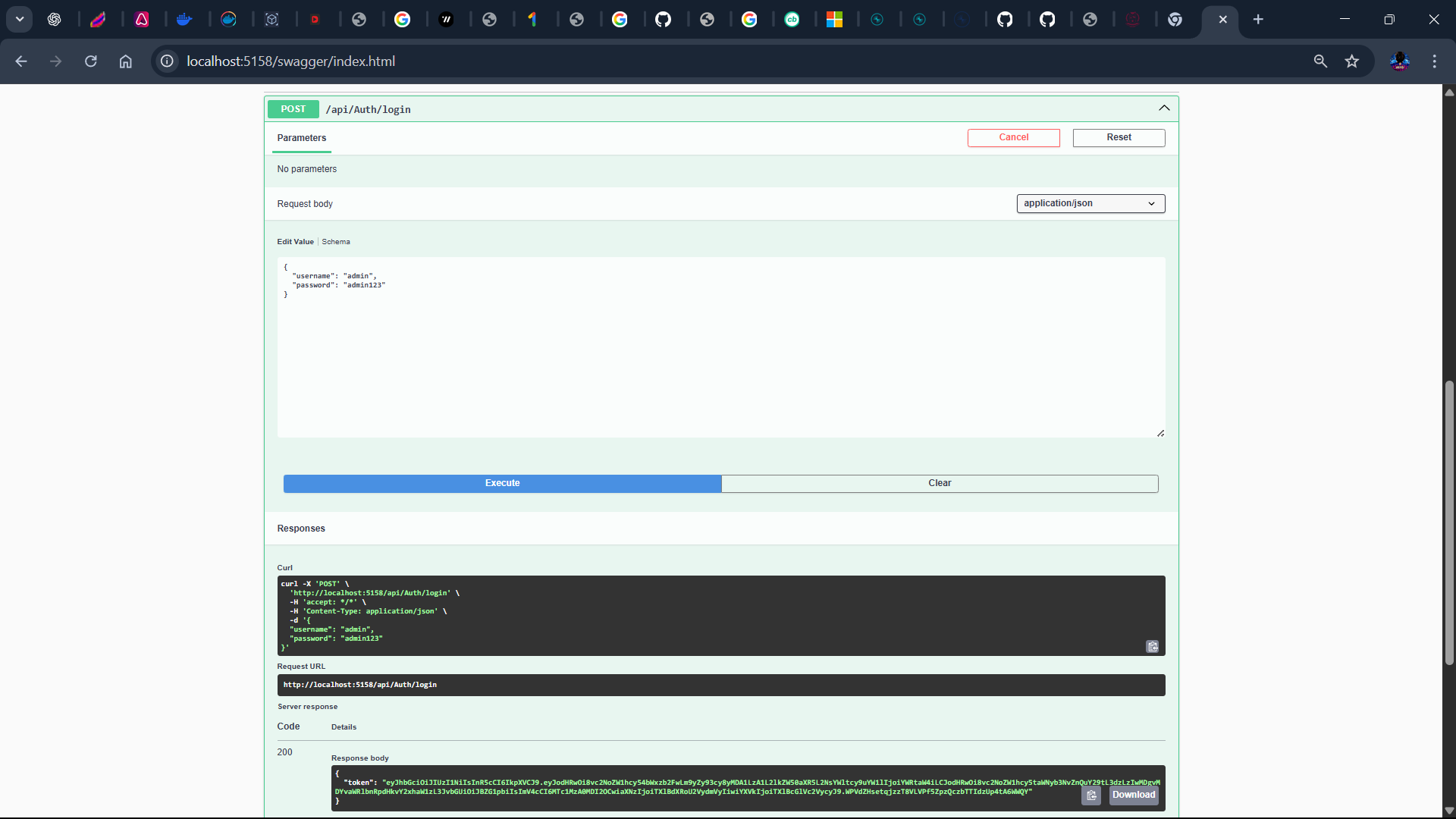
        }

    }

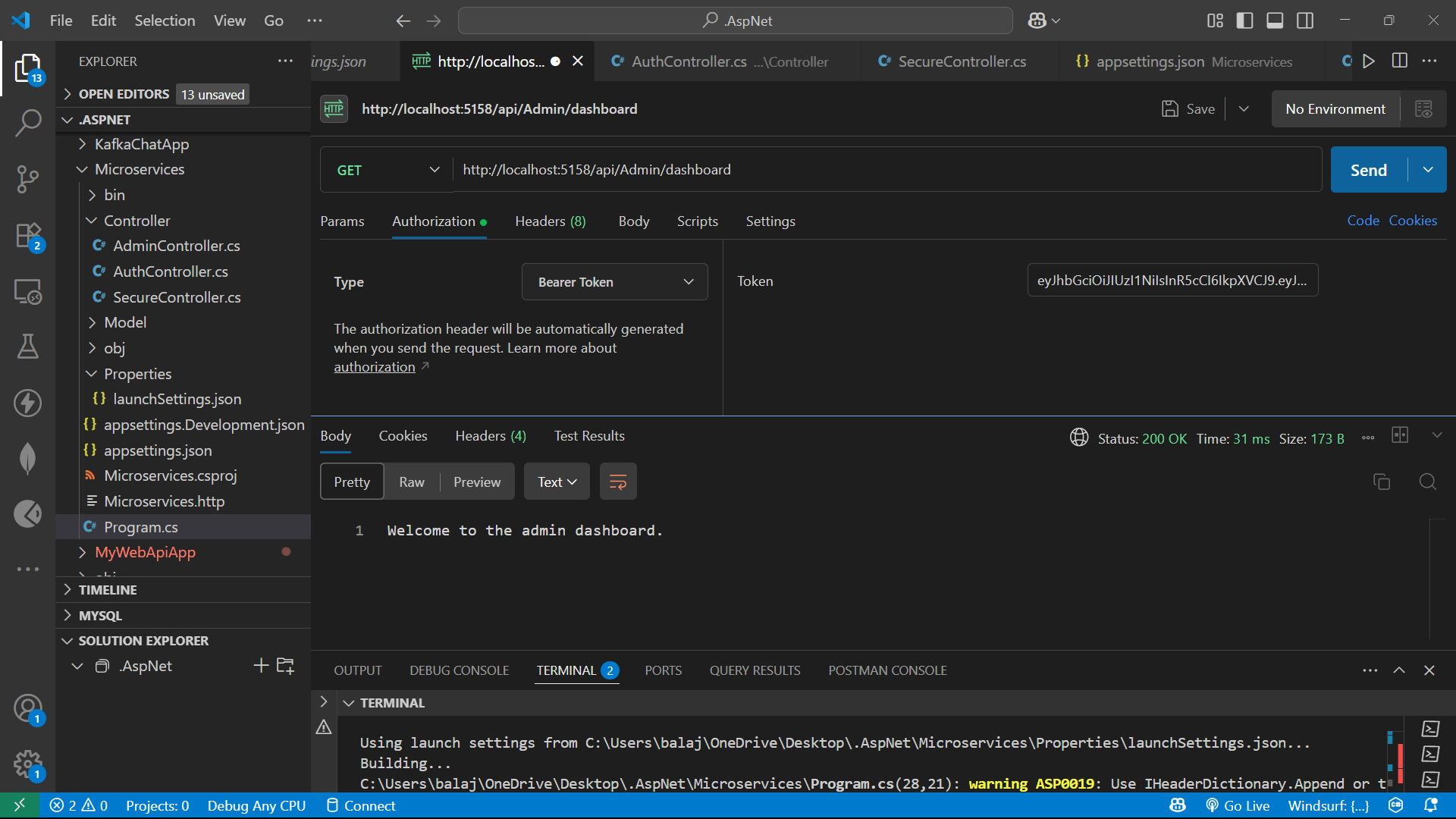
}

The Final Output will be:

Using Swagger Ui



Now using Postman to Check for the Get Request for the Secure Controller



For Secure Controller

