Q1)

CODES

1. Form1.Designer.cs  
     
   namespace KafkaChatApp

{

partial class Form1

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

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

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.components = new System.ComponentModel.Container();

this.contextMenuStrip1 = new System.Windows.Forms.ContextMenuStrip(this.components);

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

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

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

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

this.SuspendLayout();

//

// contextMenuStrip1

//

this.contextMenuStrip1.ImageScalingSize = new System.Drawing.Size(24, 24);

this.contextMenuStrip1.Name = "contextMenuStrip1";

this.contextMenuStrip1.Size = new System.Drawing.Size(61, 4);

//

// textBox1

//

this.textBox1.Location = new System.Drawing.Point(192, 205);

this.textBox1.Name = "textBox1";

this.textBox1.Size = new System.Drawing.Size(254, 26);

this.textBox1.TabIndex = 2;

this.textBox1.Text = "Please Enter Your Message Here:";

//

// txtMessage

//

this.txtMessage.Location = new System.Drawing.Point(452, 205);

this.txtMessage.Name = "txtMessage";

this.txtMessage.Size = new System.Drawing.Size(100, 26);

this.txtMessage.TabIndex = 3;

//

// btnSend

//

this.btnSend.Location = new System.Drawing.Point(22, 415);

this.btnSend.Name = "btnSend";

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

this.btnSend.TabIndex = 4;

this.btnSend.Text = "Send";

this.btnSend.UseVisualStyleBackColor = true;

// The line below is the one I fixed. It now points to the correct method name.

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

//

// btnCancel

//

this.btnCancel.Location = new System.Drawing.Point(697, 415);

this.btnCancel.Name = "btnCancel";

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

this.btnCancel.TabIndex = 5;

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(785, 442);

this.Controls.Add(this.btnCancel);

this.Controls.Add(this.btnSend);

this.Controls.Add(this.txtMessage);

this.Controls.Add(this.textBox1);

this.Name = "Form1";

this.Text = "Form1";

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.ContextMenuStrip contextMenuStrip1;

private System.Windows.Forms.TextBox textBox1;

private System.Windows.Forms.TextBox txtMessage;

private System.Windows.Forms.Button btnSend;

private System.Windows.Forms.Button btnCancel;

}

}

1. Form1.cs

using Confluent.Kafka;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KafkaChatApp

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private async Task SendMessageAsync(string message)

{

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

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

{

try

{

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

}

catch (ProduceException<Null, string> e)

{

MessageBox.Show($"Delivery failed: {e.Error.Reason}");

}

}

}

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

{

if (!string.IsNullOrEmpty(txtMessage.Text))

{

await SendMessageAsync(txtMessage.Text);

txtMessage.Clear();

}

}

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

{

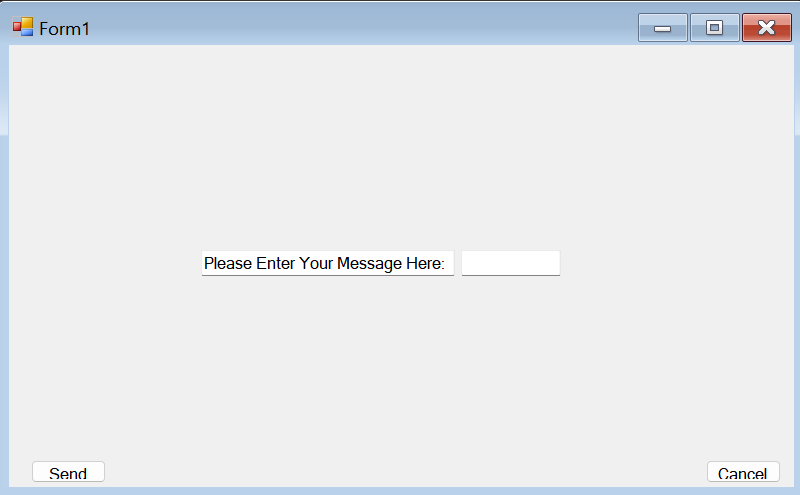
this.Close();

}

}

}

1. Form1.Design



1. Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KafkaChatApp

{

internal static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

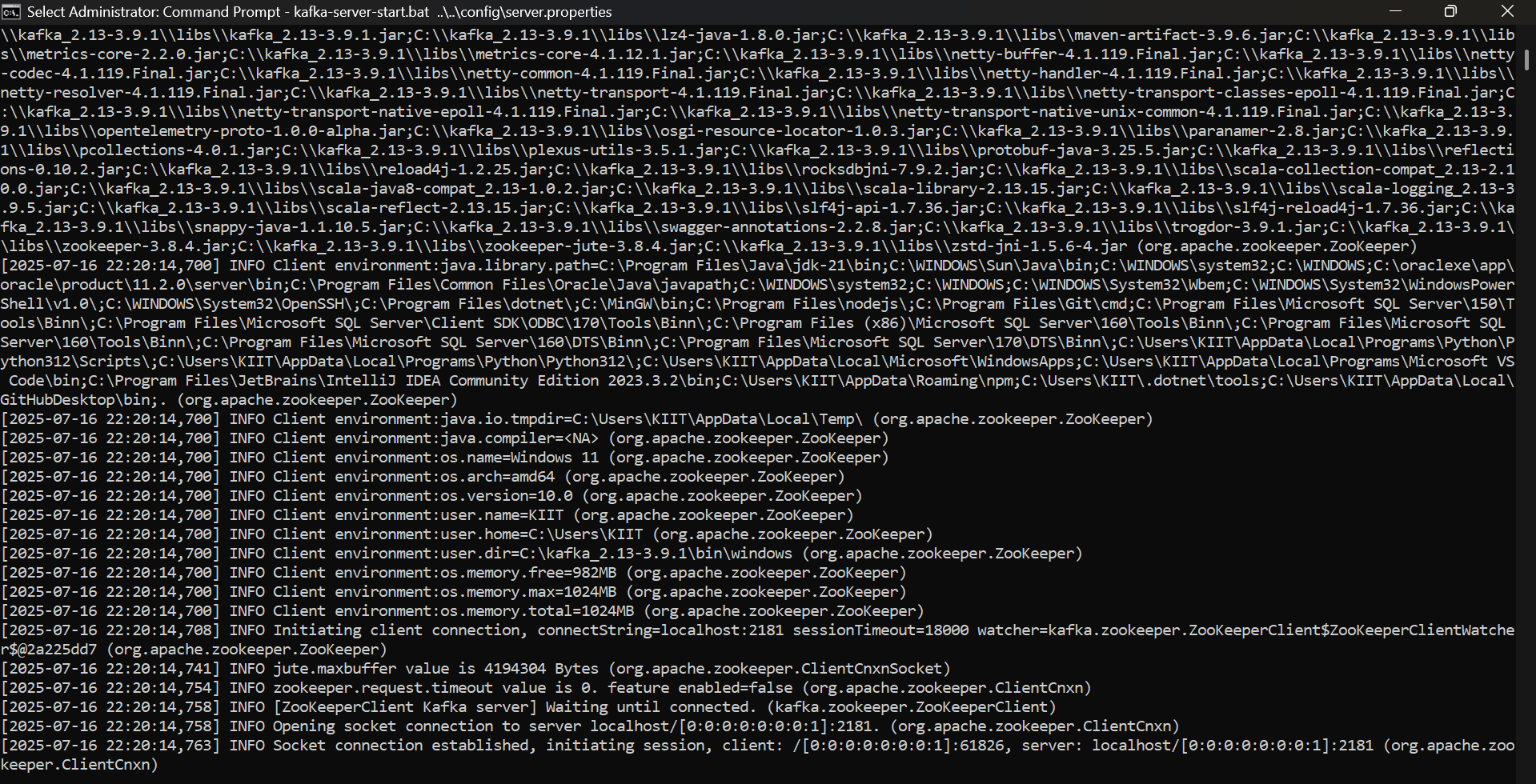
}

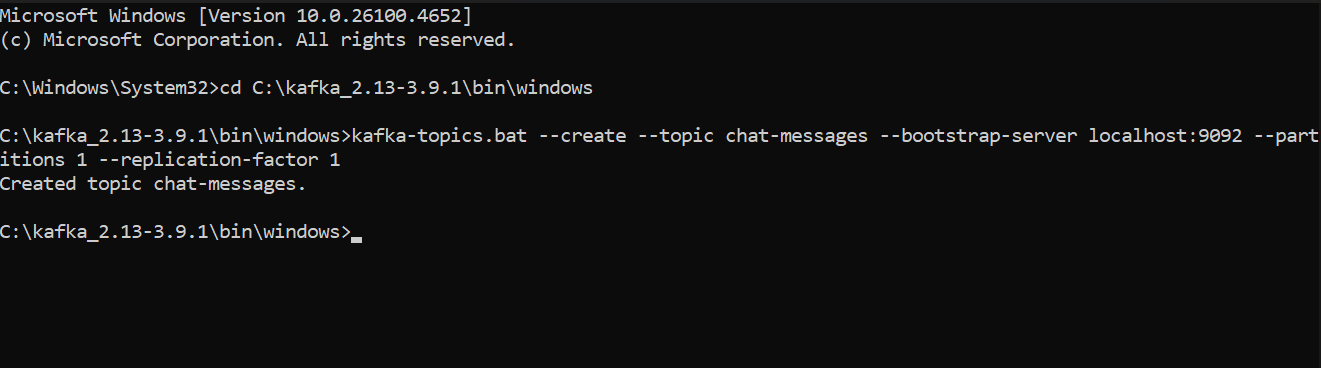
}

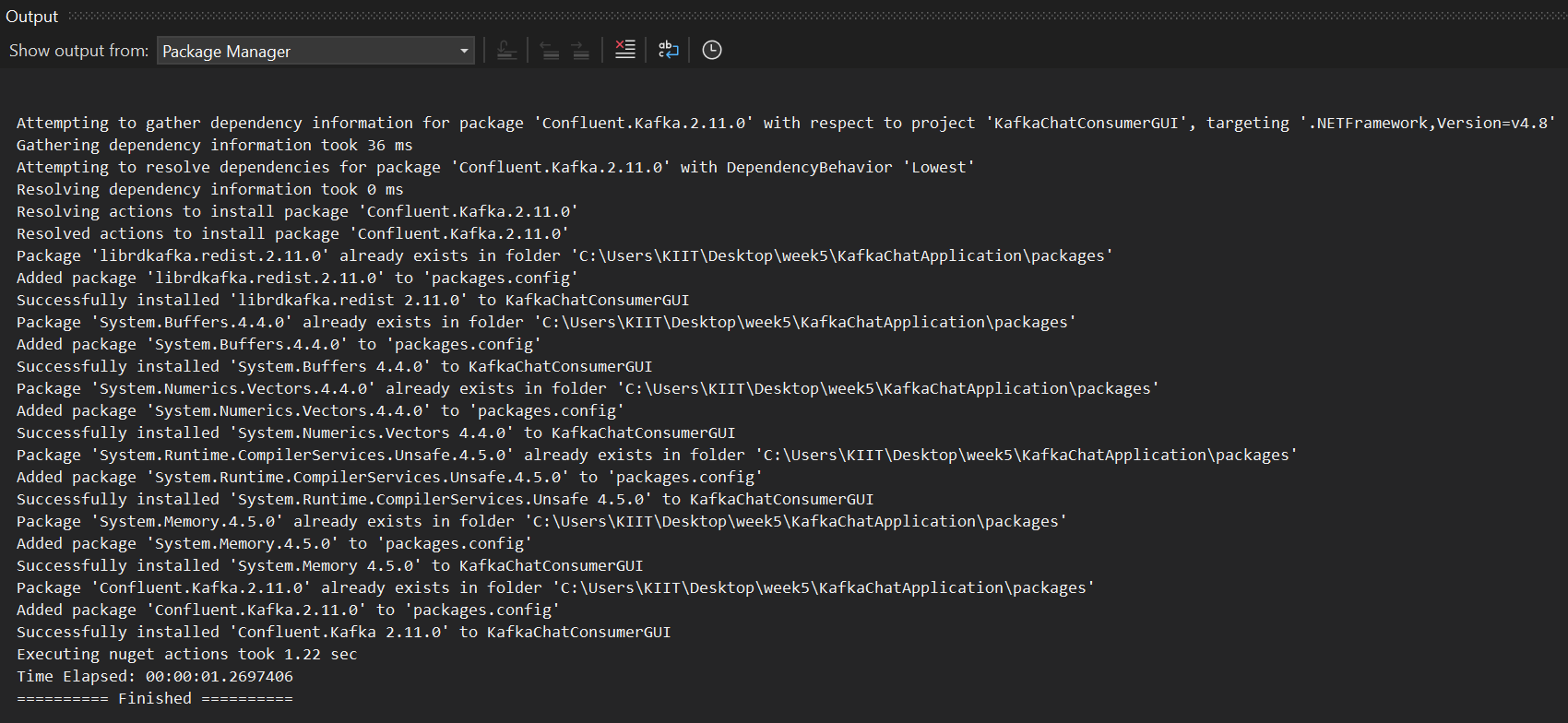
}

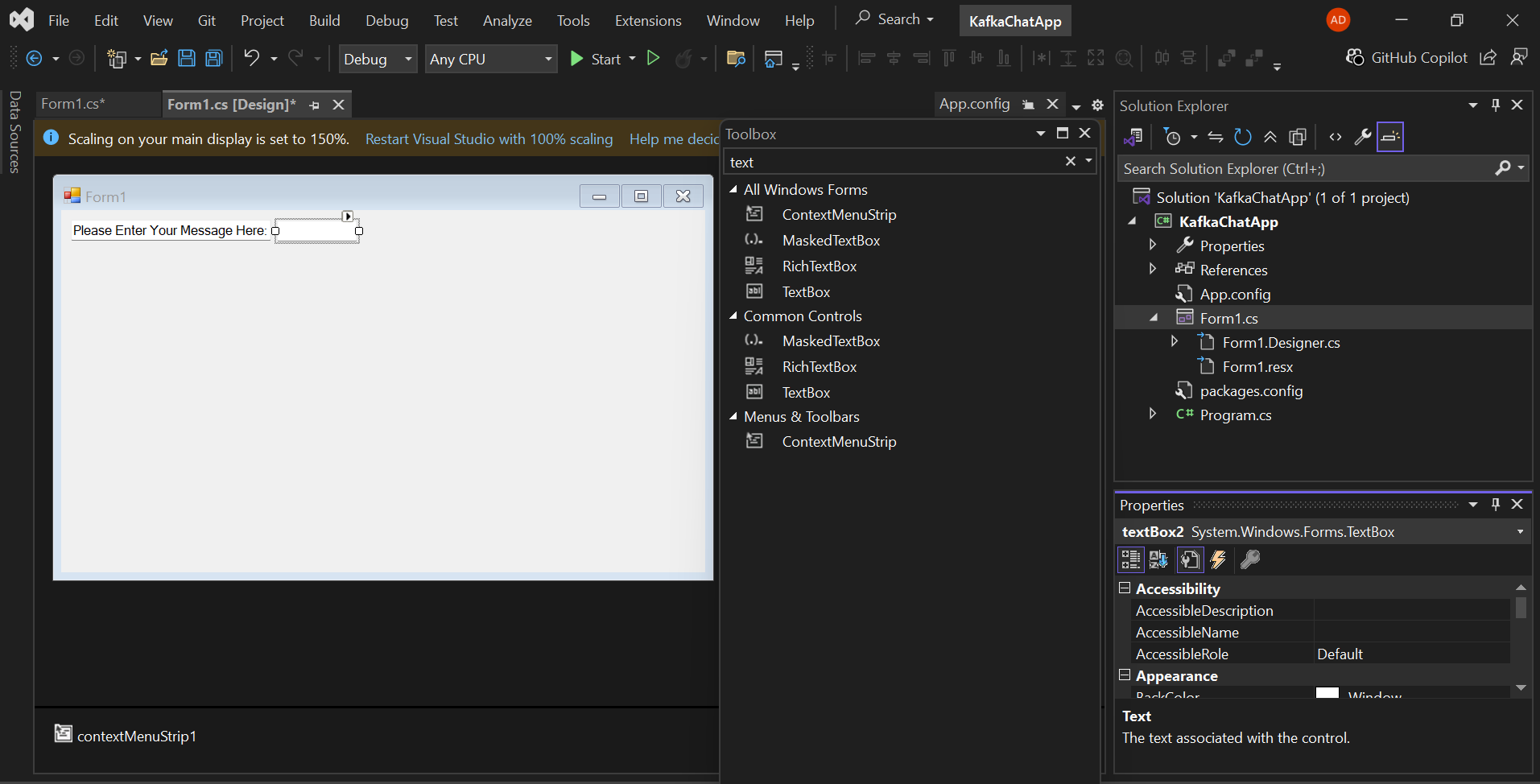
Outputs)

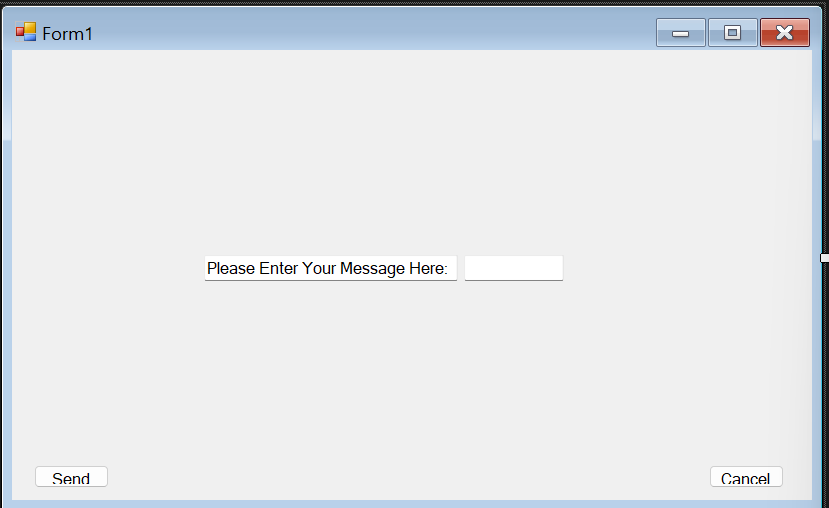


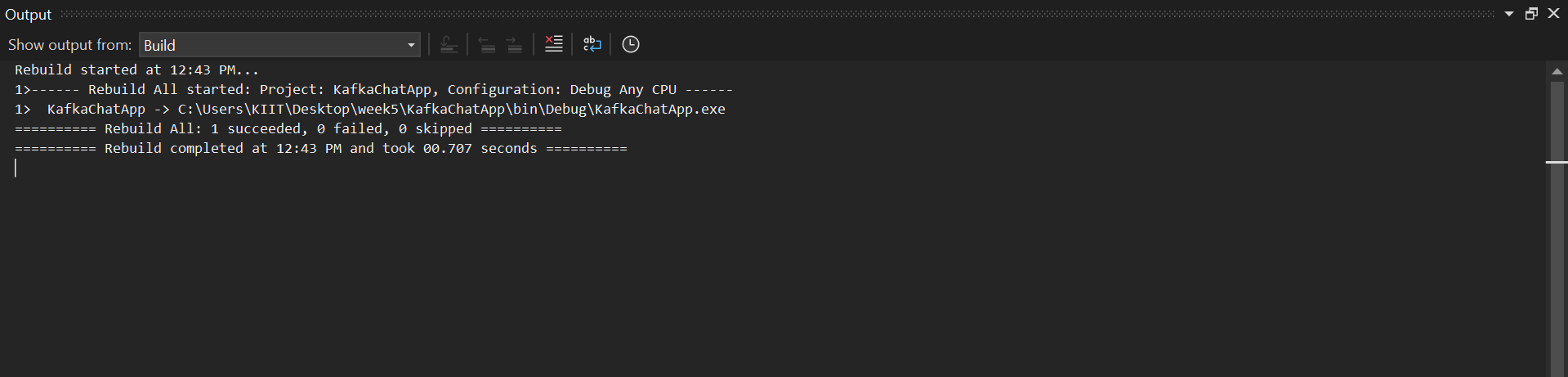


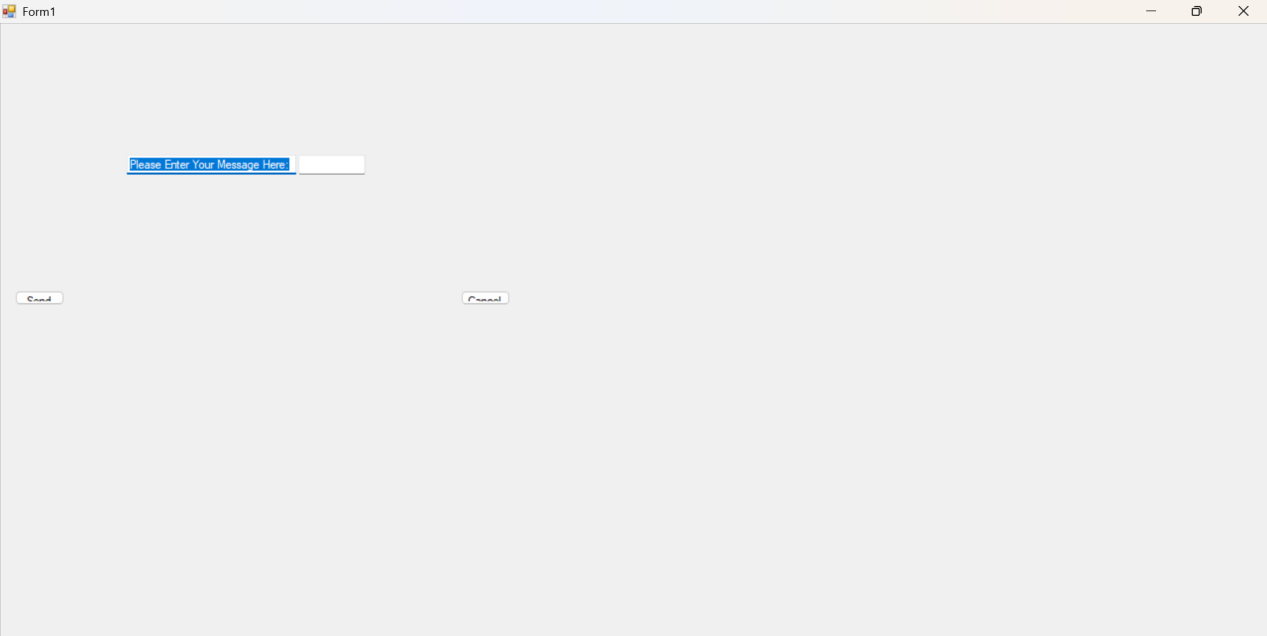


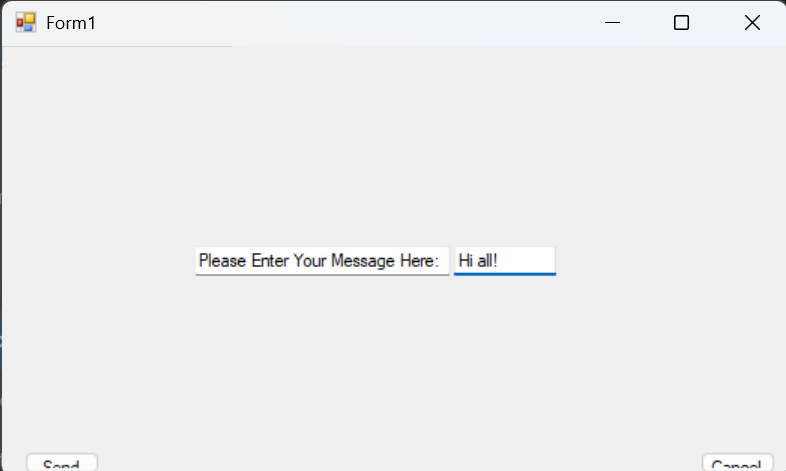


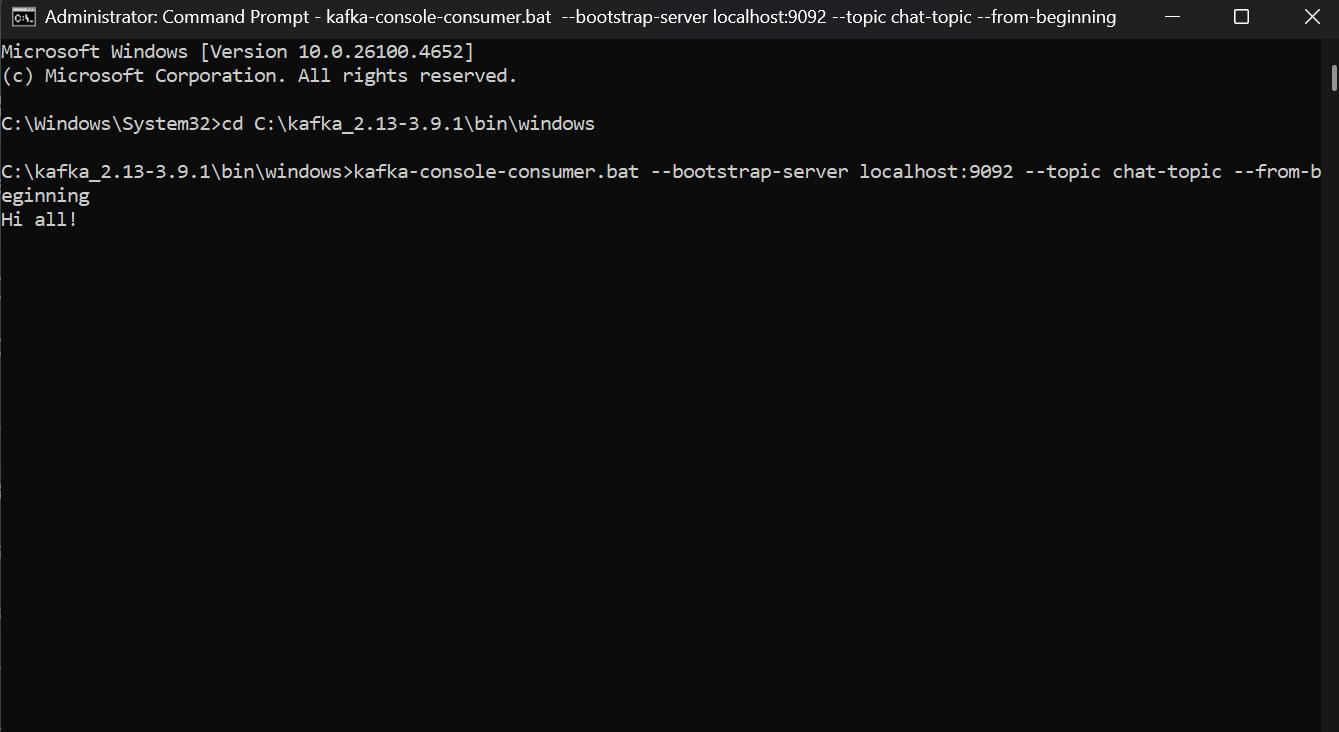












Q2)

CODES

1. WeatherForecastController.cs

using Microsoft.AspNetCore.Authorization; // <-- 1. ADD THIS USING STATEMENT

using Microsoft.AspNetCore.Mvc;

namespace MySecureApi.Controllers

{

[Authorize] // <-- 2. ADD THIS ATTRIBUTE TO PROTECT THE CONTROLLER

[ApiController]

[Route("[controller]")]

public class WeatherForecastController : ControllerBase

{

private static readonly string[] Summaries = new[]

{

"Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching"

};

private readonly ILogger<WeatherForecastController> \_logger;

public WeatherForecastController(ILogger<WeatherForecastController> logger)

{

\_logger = logger;

}

[HttpGet(Name = "GetWeatherForecast")]

public IEnumerable<WeatherForecast> Get()

{

return Enumerable.Range(1, 5).Select(index => new WeatherForecast

{

Date = DateOnly.FromDateTime(DateTime.Now.AddDays(index)),

TemperatureC = Random.Shared.Next(-20, 55),

Summary = Summaries[Random.Shared.Next(Summaries.Length)]

})

.ToArray();

}

}

}

1. AuthController.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

using MySecureApi.Models; // Make sure to include your Models namespace

namespace MySecureApi.Controllers

{

[ApiController]

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

public class AuthController : ControllerBase

{

private readonly IConfiguration \_configuration;

// Inject IConfiguration to access appsettings.json

public AuthController(IConfiguration configuration)

{

\_configuration = configuration;

}

[HttpPost("login")]

public IActionResult Login([FromBody] LoginModel model)

{

// In a real app, you'd validate against a database.

// For this exercise, we'll use a simple hardcoded check.

if (IsValidUser(model))

{

var token = GenerateJwtToken(model.Username);

return Ok(new { Token = token });

}

return Unauthorized("Invalid credentials");

}

private bool IsValidUser(LoginModel model)

{

// Hardcoded user for demonstration purposes

return model.Username == "test" && model.Password == "password";

}

private string GenerateJwtToken(string username)

{

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

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

// Add claims (information about the user)

var claims = new[]

{

new Claim(JwtRegisteredClaimNames.Sub, username),

new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString())

};

var token = new JwtSecurityToken(

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

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

claims: claims,

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

signingCredentials: credentials);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

}

3)Program.cs  
  
using System.Text;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

// =================================================================

// START: Modified Swagger/OpenAPI configuration

// =================================================================

builder.Services.AddSwaggerGen(options =>

{

// Define the security scheme (JWT)

options.AddSecurityDefinition("Bearer", new Microsoft.OpenApi.Models.OpenApiSecurityScheme

{

Name = "Authorization",

Type = Microsoft.OpenApi.Models.SecuritySchemeType.Http,

Scheme = "Bearer",

BearerFormat = "JWT",

In = Microsoft.OpenApi.Models.ParameterLocation.Header,

Description = "JWT Authorization header using the Bearer scheme."

});

// Make sure Swagger includes the token in requests

options.AddSecurityRequirement(new Microsoft.OpenApi.Models.OpenApiSecurityRequirement

{

{

new Microsoft.OpenApi.Models.OpenApiSecurityScheme

{

Reference = new Microsoft.OpenApi.Models.OpenApiReference

{

Type = Microsoft.OpenApi.Models.ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

new string[] {}

}

});

});

// =================================================================

// END: Modified Swagger/OpenAPI configuration

// =================================================================

// Add Authentication Services

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();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

// Add Authentication middleware

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

4) appsetting.json

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"Jwt": {

"Key": "ThisIsASecretKeyForJwtTokenThatIsLongEnough",

"Issuer": "MyAuthServer",

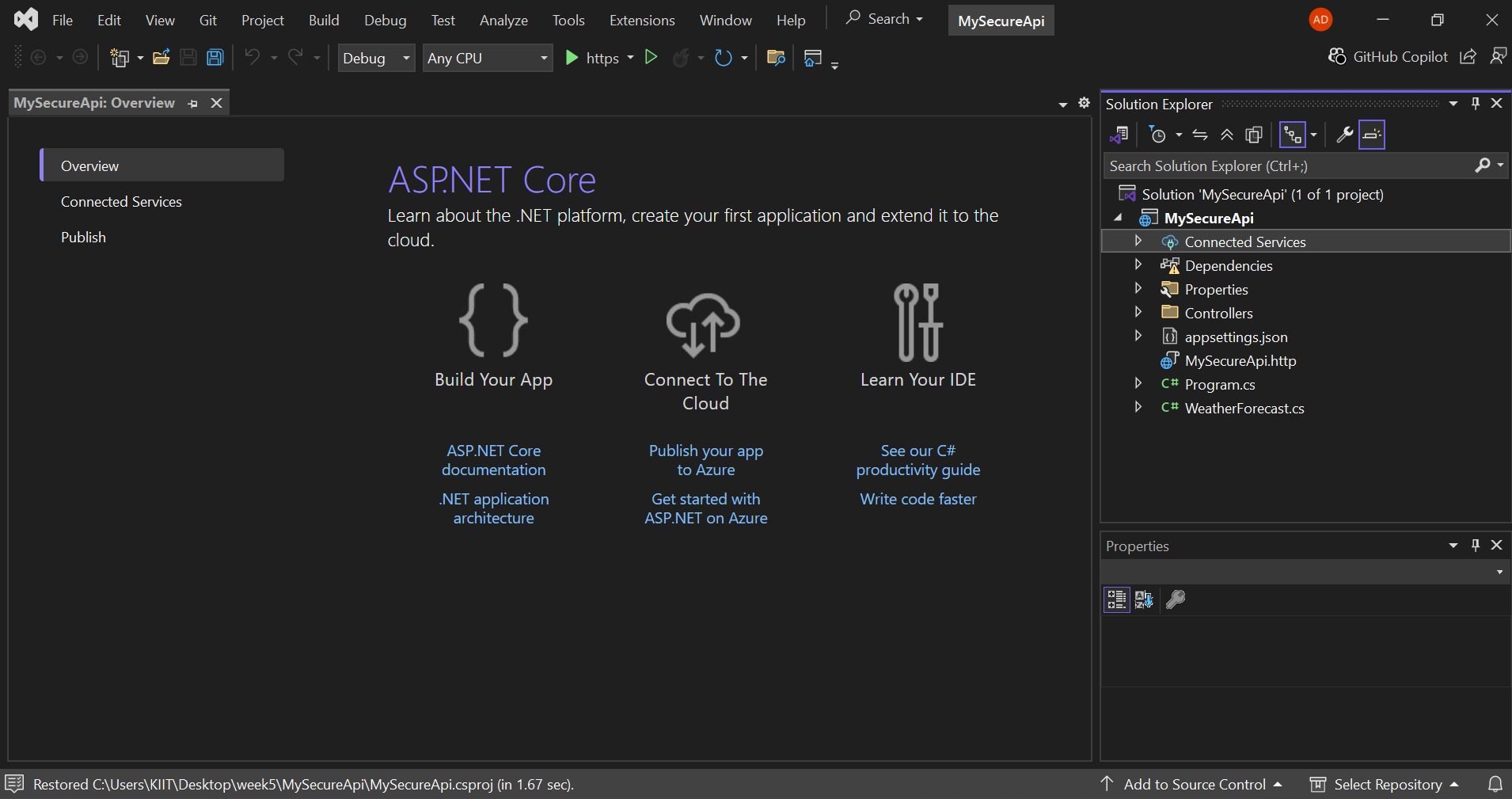
"Audience": "MyApiUsers",

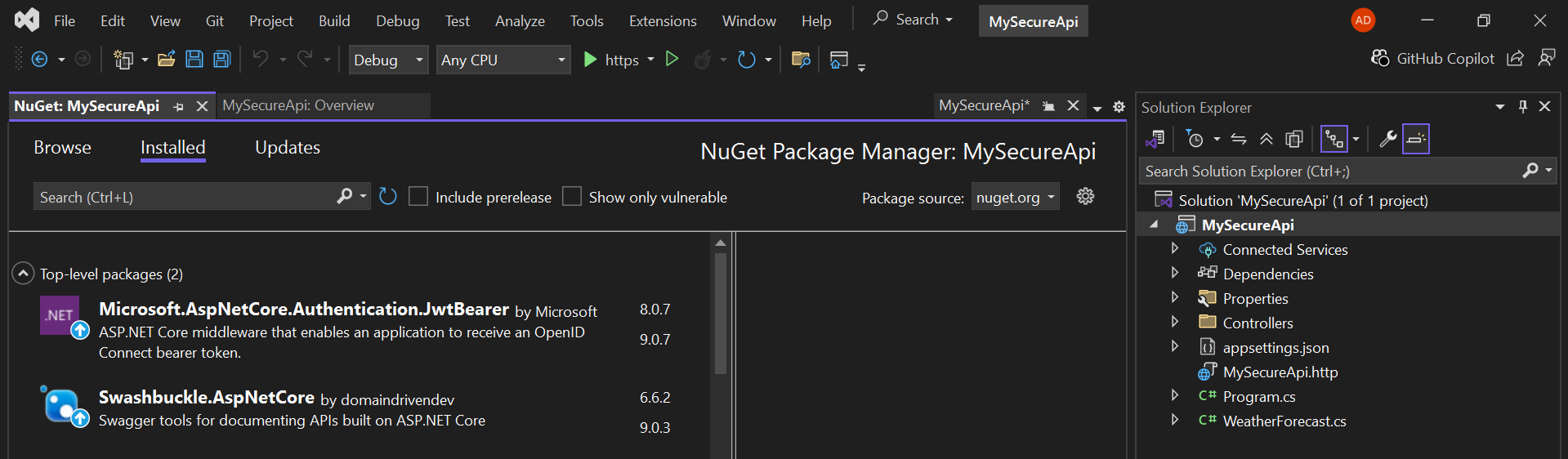
"DurationInMinutes": 60

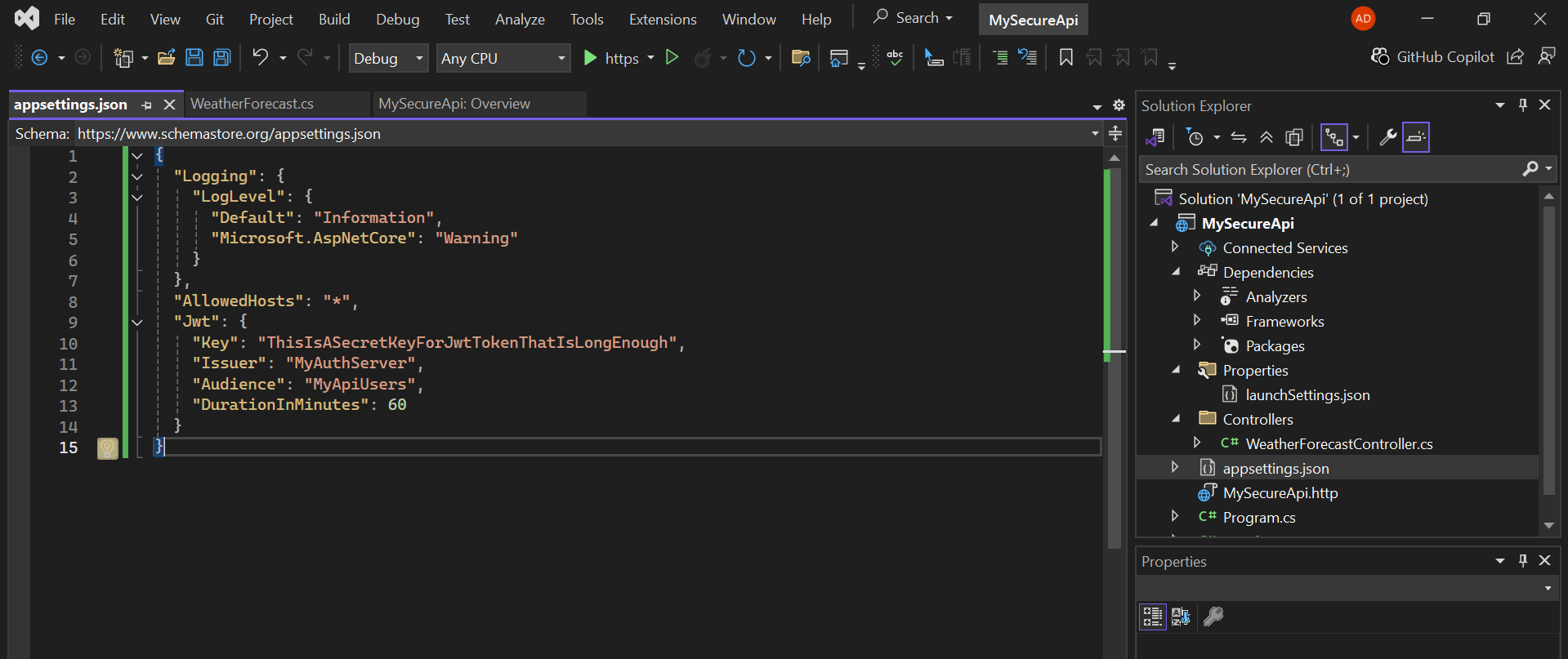
}

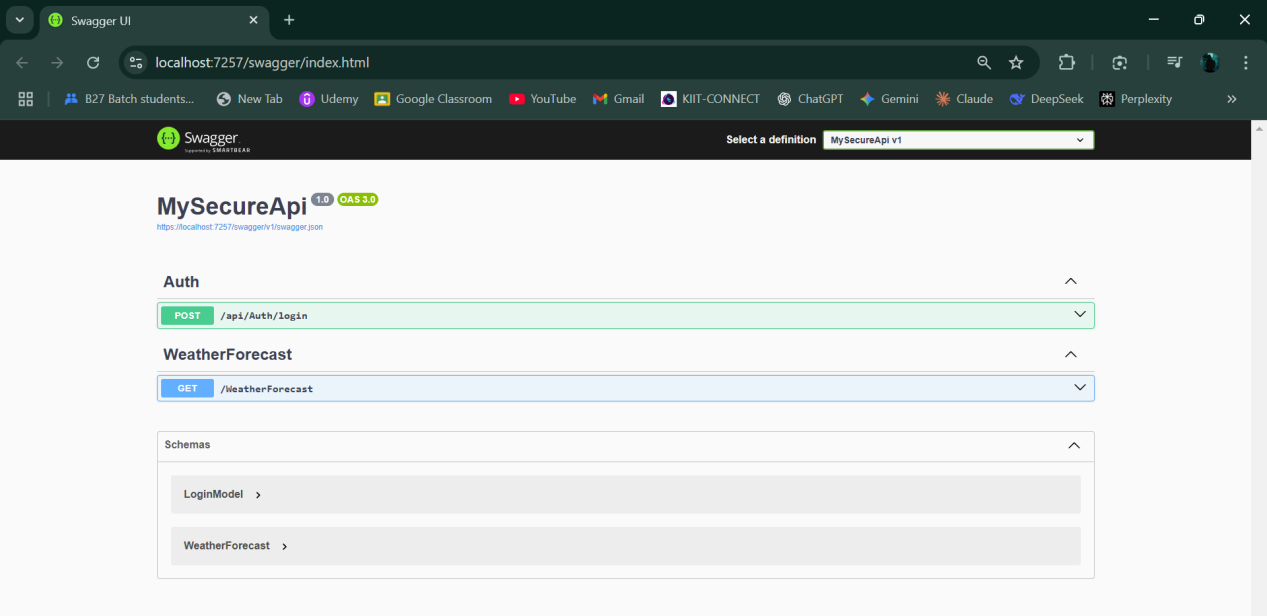
}

OUTPUTS







\

