**WEEK 5**

**Module6-ASP.NET Core 8.0 Web API**

**Ex6: WebApi\_Handson**

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

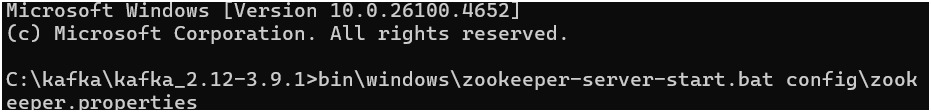
**Set Up Kafka on Windows**

**Step 1: Download and Extract Kafka**

1. Go to: <https://kafka.apache.org/downloads>
2. Download Kafka (e.g., kafka\_2.12-2.2.0)
3. Extract it to D:\Kafka

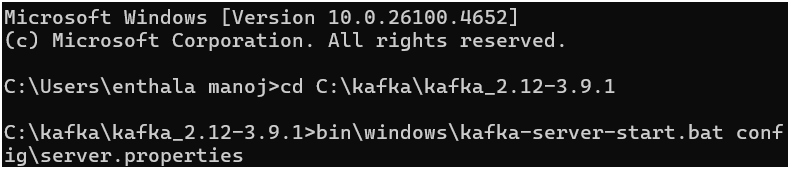
**Step 2: Start Zookeeper**

Open PowerShell or CMD and run:

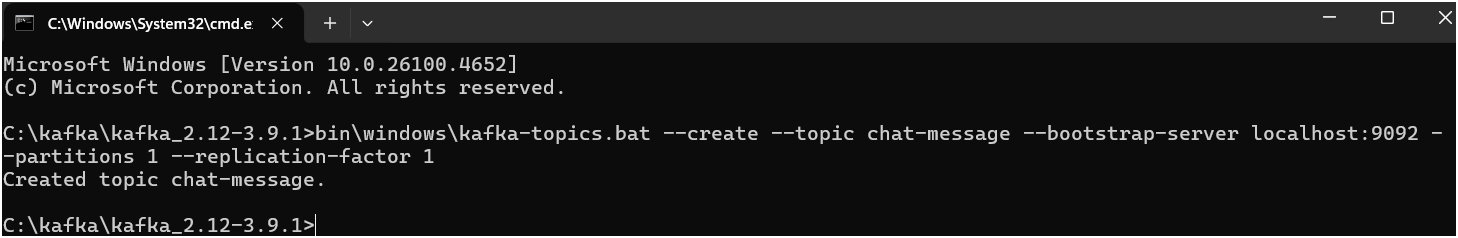


**Step 3: Start Kafka Broker**

In a new PowerShell window:



**PART 4: Create Kafka Topic**

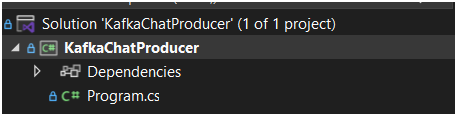
****

**PART 5: Build C# Chat App (Console First)**

Step 1: Create a Console App

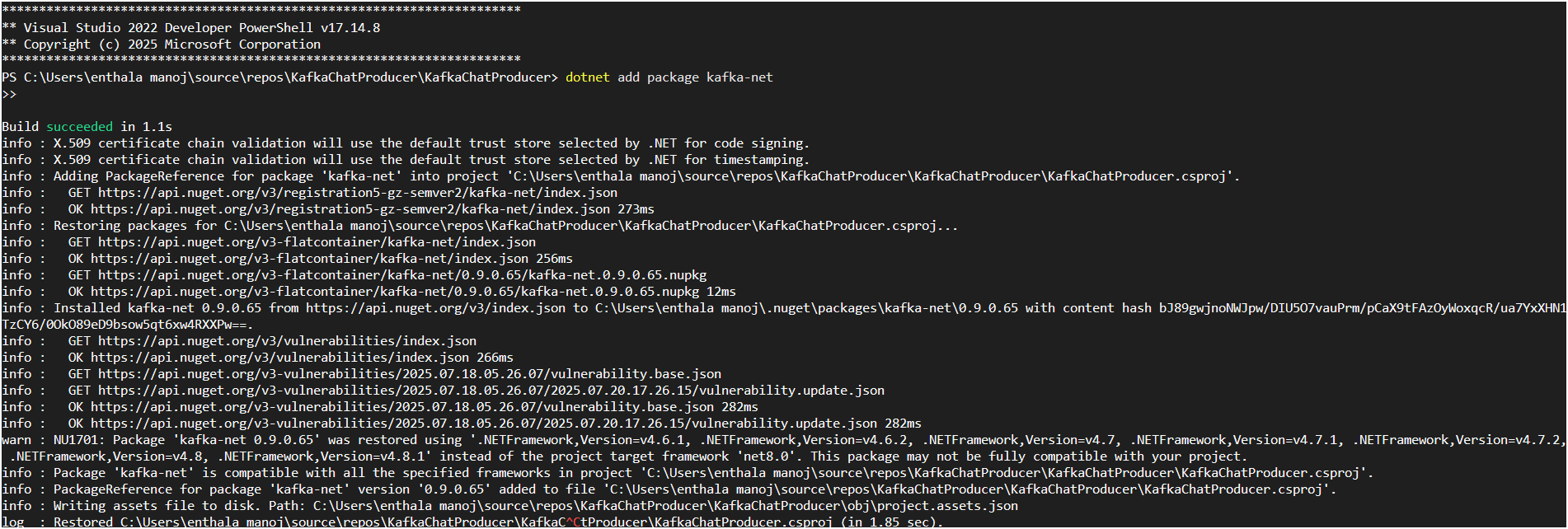
Open Visual Studio → Create New Project → Choose:

* Console App (.NET Core)
* Name: KafkaChatProducer



**Step 2: Add NuGet Package**

Open **Package Manager Console** or run:



**Step 3: Producer Code (Send Messages to Kafka)**

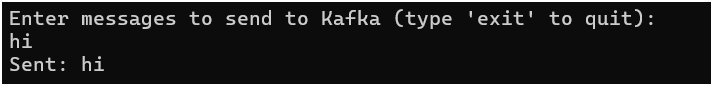
Replace Program.cs with:



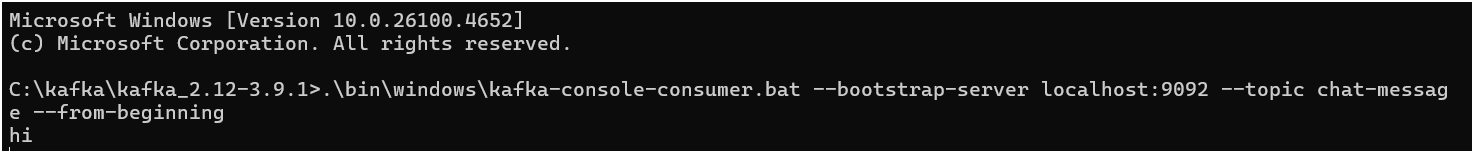
**PART 6: Test Producer**

1. Run Kafka + Zookeeper
2. Run the C# console app
3. Open another terminal and start Kafka Consumer:

Sender:



Recevier:

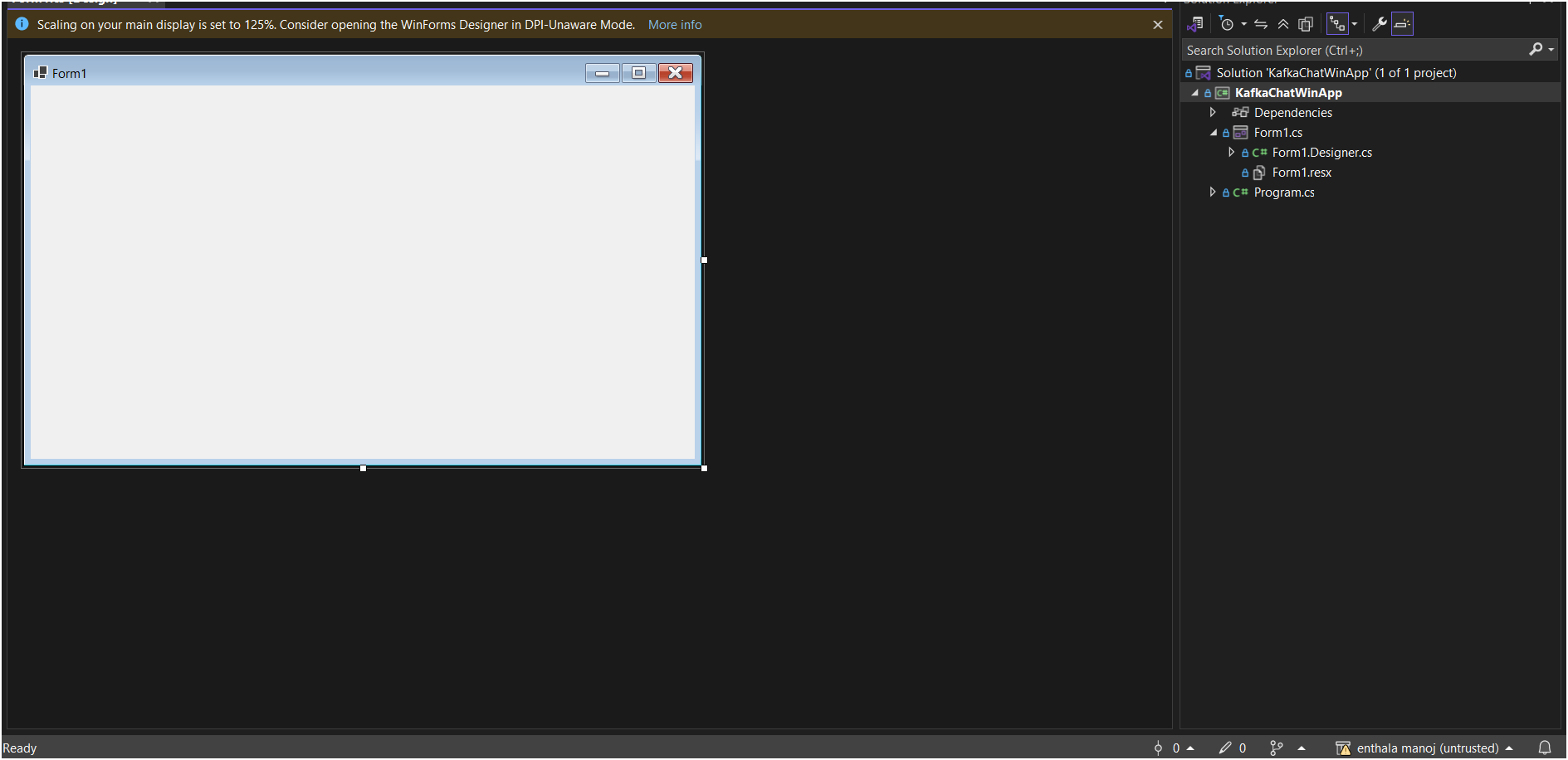


**2. Create a Chat Application using C# Windows Application using Kafka and consume the message in different client applications.**

**Step 1: Create a New Windows Forms App**

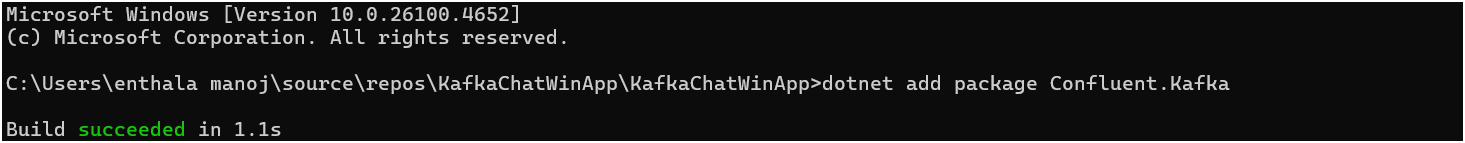
**In Visual Studio:**

* **File > New > Project**
* **Choose Windows Forms App (.NET Core or .NET 6/7)**
* **Name: KafkaChatWinApp**

****

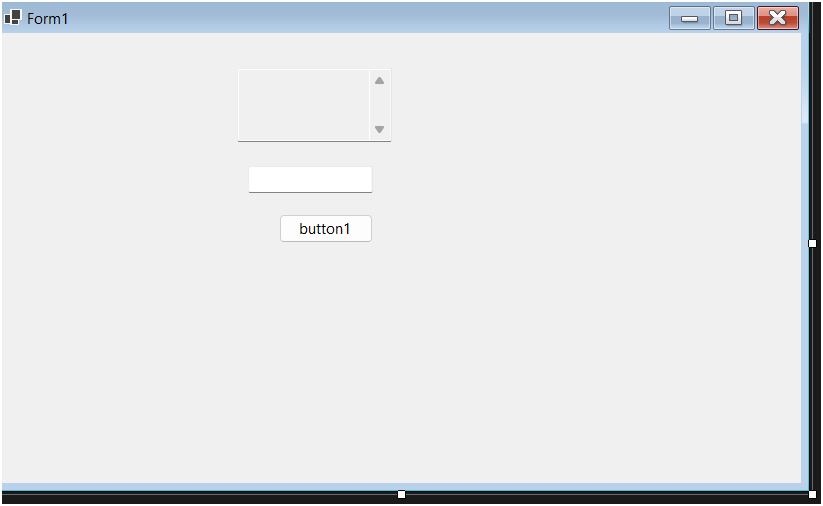
**Step 2: Add Kafka Client Library**

* **Using NuGet Package Manager Console**:

****

**Step 3: Design the Form UI**

* **TextBox (Multiline) → For message display (txtChat)**
* **TextBox → For typing a new message (txtInput)**
* **Button → Send (btnSend)**

****

**Step 4: Kafka Producer & Consumer Code**

**Form1.cs** (Simplified version using Confluent.Kafka):

using System;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

using Confluent.Kafka;

namespace KafkaChatWinApp

{

public partial class Form1 : Form

{

private readonly string topic = "chat-message";

private readonly string bootstrapServers = "localhost:9092";

public Form1()

{

InitializeComponent();

Task.Run(() => StartConsumer());

}

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

{

var config = new ProducerConfig { BootstrapServers = bootstrapServers };

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

await producer.ProduceAsync(topic, new Message<Null, string> { Value = txtInput.Text });

txtChat.AppendText("You: " + txtInput.Text + Environment.NewLine);

txtInput.Clear();

}

private void StartConsumer()

{

var config = new ConsumerConfig

{

BootstrapServers = bootstrapServers,

GroupId = Guid.NewGuid().ToString(),

AutoOffsetReset = AutoOffsetReset.Earliest

};

using var consumer = new ConsumerBuilder<Ignore, string>(config).Build();

consumer.Subscribe(topic);

try

{

while (true)

{

var cr = consumer.Consume();

this.Invoke((MethodInvoker)delegate

{

txtChat.AppendText("Other: " + cr.Message.Value + Environment.NewLine);

});

}

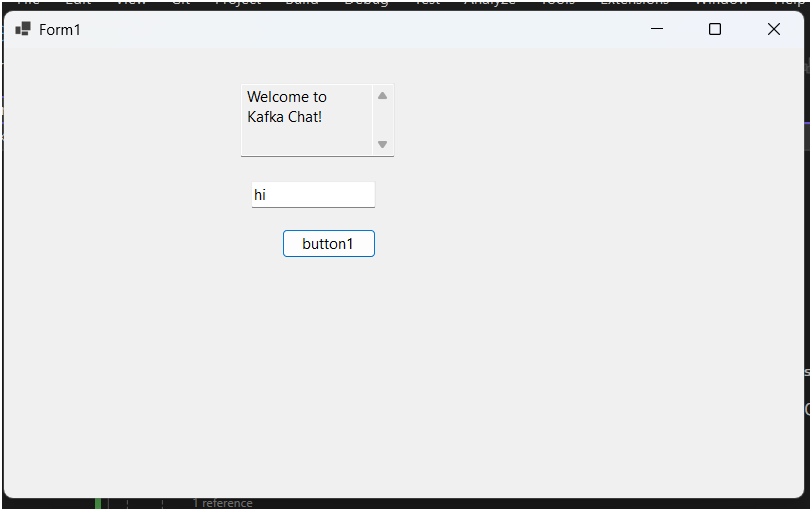
}

catch (OperationCanceledException) { consumer.Close(); }

}

}

}

**Output:**

**1.Microservices – JWT**

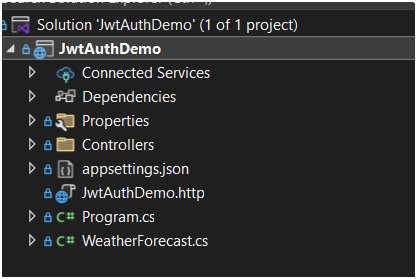
**Exercise1:Implement JWT Authentication in ASP.NET Core Web API**

**Scenario:**

**You are building a microservice that requires secure login. You need to implement JWT-based authentication**

**Step-by-Step in Visual Studio**

1. **Create a new ASP.NET Core Web API project**
   * **Open Visual Studio → Create a new project**
   * **Choose ASP.NET Core Web API → Click Next**
   * **Name: JwtAuthDemo → Click Create**
   * **Select .NET 6.0 or .NET 7.0, uncheck “Use controllers (Minimal API)” if checked → Click Create**

****

1. **Add the NuGet package**

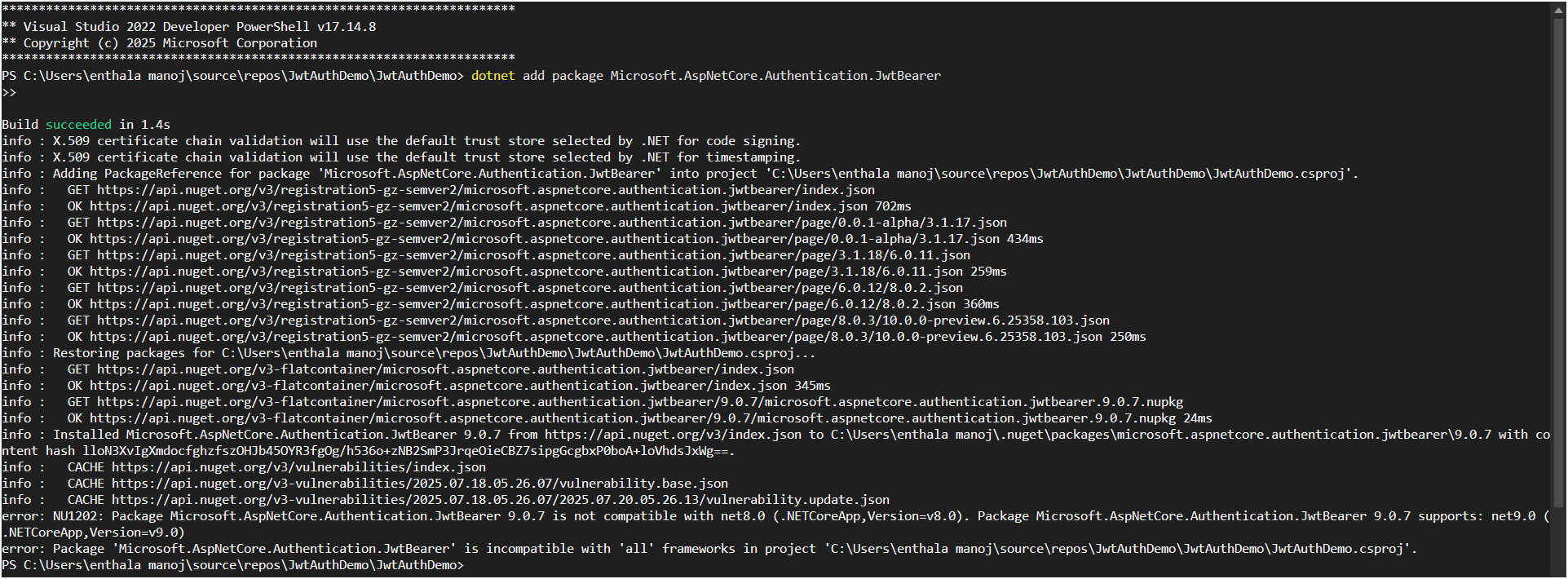
**Tools > NuGet Package Manager > Package Manager Console**

**Run:**

**bash**

**CopyEdit**

**dotnet add package Microsoft.AspNetCore.Authentication.JwtBearer**

**3.Add configuration in appsettings.json**

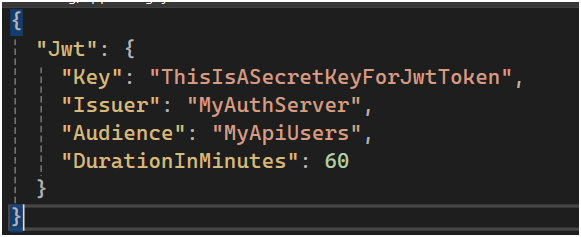
**{**

**"Jwt":**

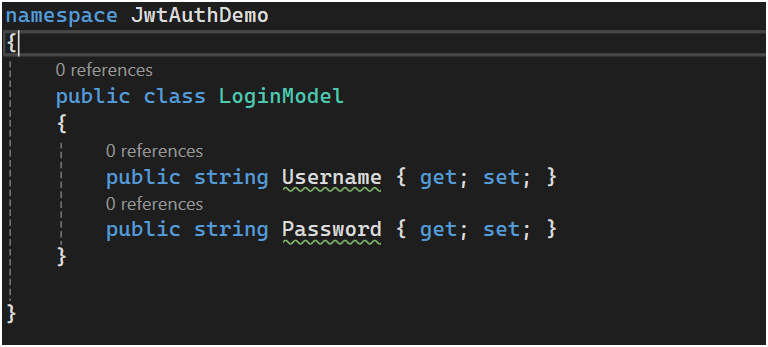
**{**

**"Key": "ThisIsASecretKeyForJwtToken", "Issuer": "MyAuthServer", "Audience": "MyApiUsers", "DurationInMinutes": 60**

**} }**

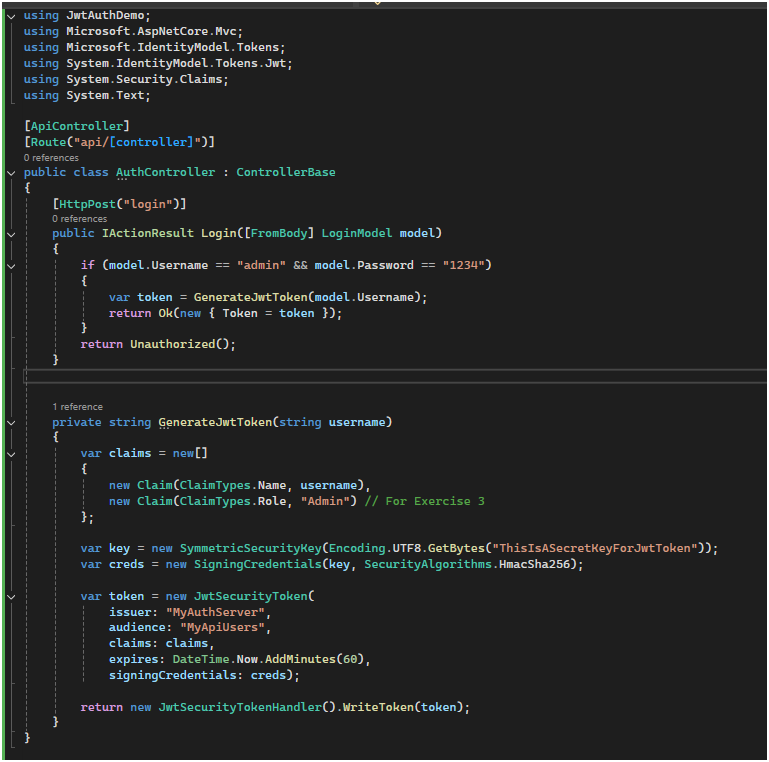
****

**4.Create a Models/LoginModel.cs file**

**5.Configure JWT in Program.cs**

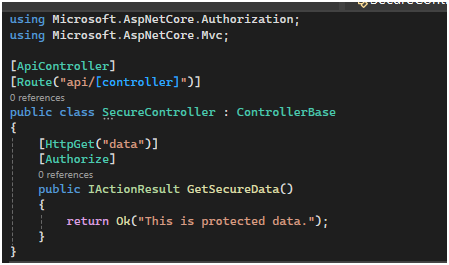
****

**6.Create Controllers/AuthController.cs**

****

**Exercise 2:** **Secure an API Endpoint**

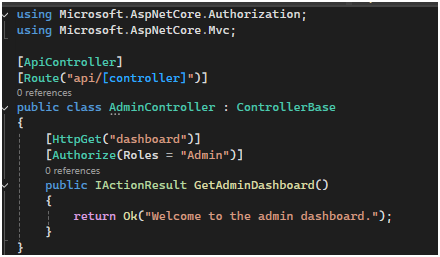
**1.Create Controllers/SecureController.cs**

****

**EXERCISE 3: Role-Based Authorization**

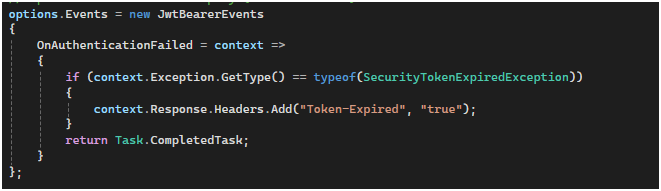
**1.Already added ClaimTypes.Role = "Admin" in token generation above.**

**2.Create Controllers/AdminController.cs**

****

**EXERCISE 4: Token Expiry Handling**

**Already added this part in Program.cs:**

****