**WEEK 5 HANDS ON**

Submitted by:Simran Raghav(6362006)

**ASP.NET CORE 8.0**

**# Install Kafka and Zookeeper**

# Start Zookeeper

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

# Start Kafka Server

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

**# Create Kafka Topic**  
# Create topic named "chat-topic"

## .\bin\windows\kafka-topics.bat --create --topic chat-topic --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1

# List topics.\bin\windows\kafka-topics.bat --list --bootstrap-server localhost:9092  
 **#Producer**

using Confluent.Kafka;

using System;

using System.Threading.Tasks;

class Producer

{

public 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 messages to send to Kafka (type 'exit' to quit):");

string input;

do

{

input = Console.ReadLine();

if (input != "exit")

{

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

Console.WriteLine($"Sent: {input}");

}

} while (input != "exit");

}

}

}

**#Subscriber Code**using Confluent.Kafka;

using System;

using System.Threading;

class Consumer

{

public static void Main(string[] args)

{

var config = new ConsumerConfig

{

GroupId = "chat-consumer-group",

BootstrapServers = "localhost:9092",

AutoOffsetReset = AutoOffsetReset.Earliest

};

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

{

consumer.Subscribe("chat-topic");

Console.WriteLine("Listening for messages... Press Ctrl+C to exit");

try

{

while (true)

{

var cr = consumer.Consume(CancellationToken.None);

Console.WriteLine($"Received: {cr.Message.Value}");

}

}

catch (OperationCanceledException)

{

consumer.Close();

}

}

}

}

**#Form Code (Producer + Consumer in UI)**using System;

using System.Windows.Forms;

using Confluent.Kafka;

using System.Threading.Tasks;

namespace KafkaChatApp

{

public partial class ChatForm : Form

{

private readonly ProducerConfig \_producerConfig = new ProducerConfig { BootstrapServers = "localhost:9092" };

private readonly ConsumerConfig \_consumerConfig = new ConsumerConfig

{

GroupId = "chat-ui-group",

BootstrapServers = "localhost:9092",

AutoOffsetReset = AutoOffsetReset.Earliest

};

public ChatForm()

{

InitializeComponent();

StartConsumer();

}

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

{

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

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

txtChat.AppendText($"You: {txtMessage.Text}\n");

txtMessage.Clear();

}

private void StartConsumer()

{

Task.Run(() =>

{

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

consumer.Subscribe("chat-topic");

try

{

while (true)

{

var cr = consumer.Consume();

Invoke(new Action(() =>

{

txtChat.AppendText($"Friend: {cr.Message.Value}\n");

}));

}

}

catch

{

consumer.Close();

}

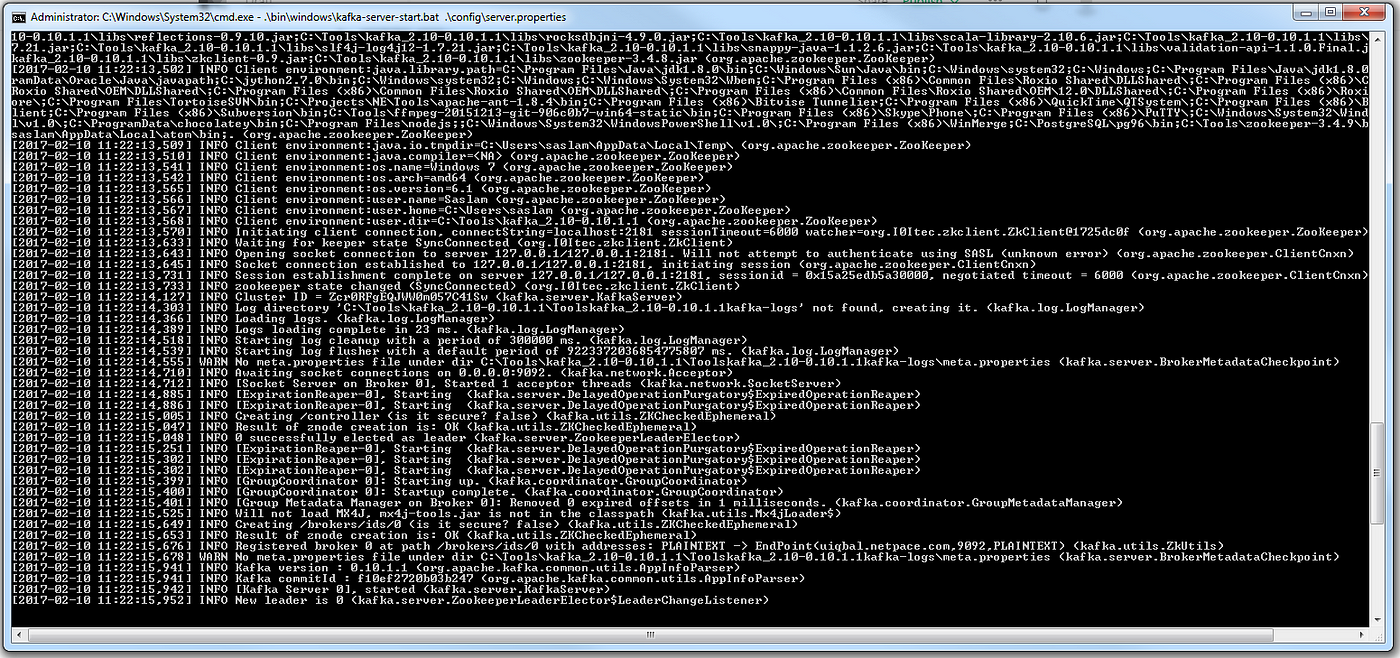
});

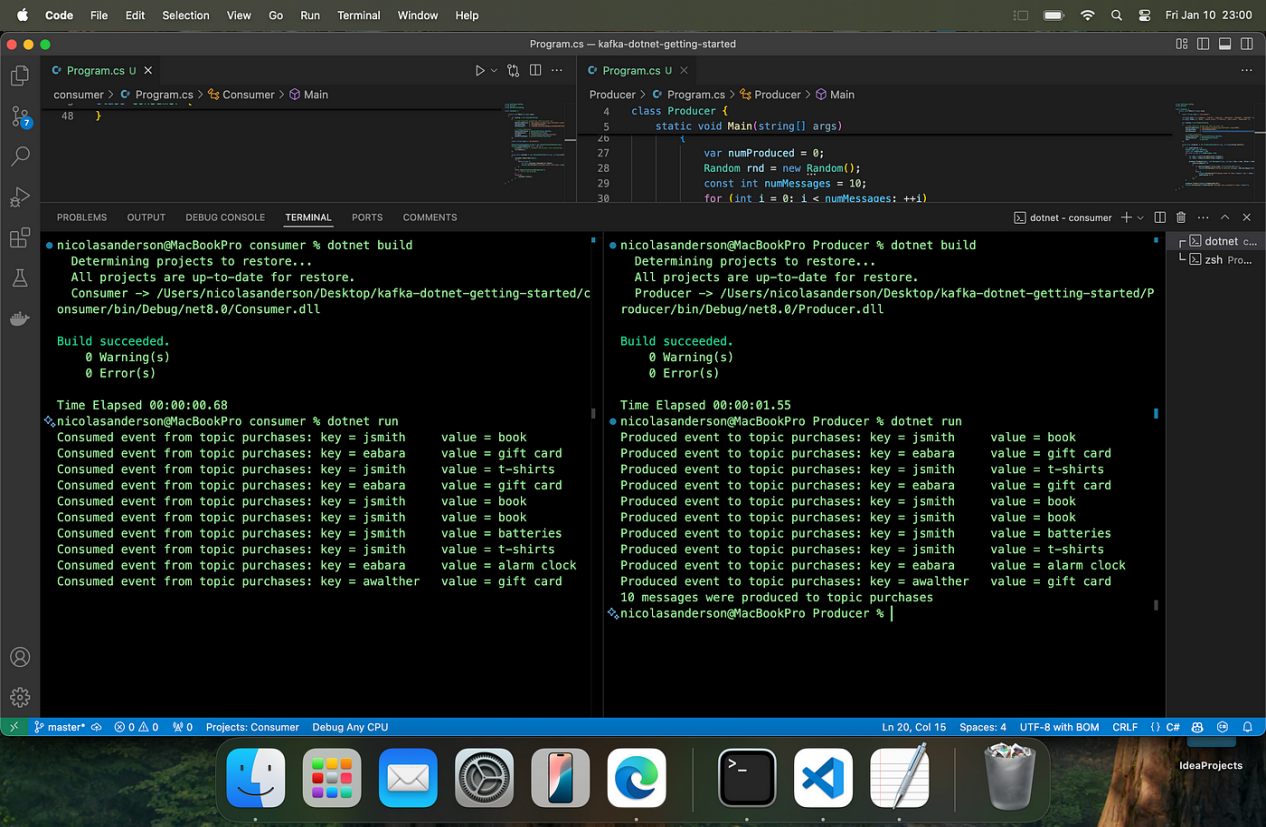
}

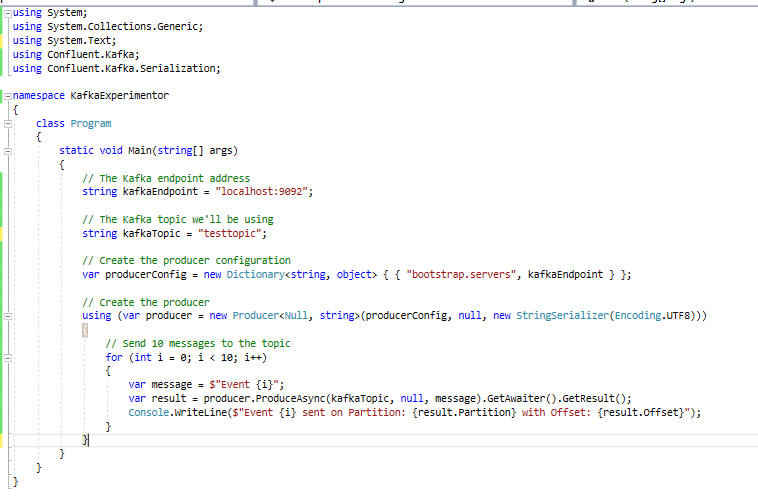
}

}

**OUTPUT**

****

****

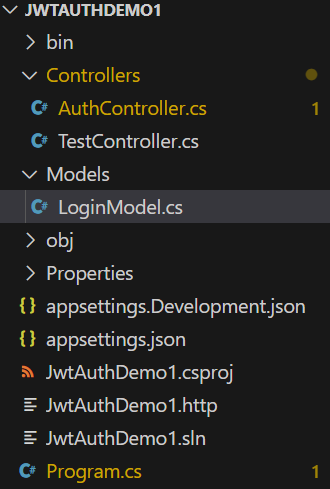
****

**MICROSERVICES-JWT**

**Ques. 1: Implement JWT Authentication in ASP.NET Core Web API**

CODE:

**# Project Structure**



**# AuthController.cs**

using JwtAuthDemo1.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace JwtAuthDemo1.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    public class AuthController : ControllerBase

    {

        private readonly IConfiguration \_configuration;

        public AuthController(IConfiguration configuration)

        {

            \_configuration = configuration;

        }

        [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 (replace with DB check if needed)

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

        }

        private string GenerateJwtToken(string username)

        {

            var claims = new[]

            {

                new Claim(ClaimTypes.Name, username)

            };

            var key = new SymmetricSecurityKey(

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

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

            var token = new JwtSecurityToken(

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

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

                claims: claims,

                expires: DateTime.Now.AddMinutes(60),

                signingCredentials: creds

            );

            return new JwtSecurityTokenHandler().WriteToken(token);

        }

    }

}

**#TestController.cs**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace JwtAuthDemo1.Controllers

{

    [ApiController]

    [Route("api/[controller]")]

    public class TestController : ControllerBase

    {

        [HttpGet("public")]

        public IActionResult Public() => Ok("✅ Public endpoint - No token needed.");

        [Authorize]

        [HttpGet("secure")]

        public IActionResult Secure() => Ok("🔐 Secure endpoint - JWT token required.");

    }

}

**#LoginModel.cs**

namespace JwtAuthDemo1.Models

{

    public class LoginModel

    {

        public string Username { get; set; } = string.Empty;

        public string Password { get; set; } = string.Empty;

    }

}

**#appsettings.json**

{

  "Jwt": {

    "Key": "ThisIsASecretKeyForJwtToken",

    "Issuer": "MyAuthServer",

    "Audience": "MyApiUsers",

    "DurationInMinutes": 60

  },

  "Logging": {

    "LogLevel": {

      "Default": "Information",

      "Microsoft.AspNetCore": "Warning"

    }

  },

  "AllowedHosts": "\*"

}

**#Program.cs**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

// JWT Authentication configuration

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"]))

        };

    });

builder.Services.AddAuthorization();

var app = builder.Build();

app.UseAuthentication(); // This MUST come before UseAuthorization

app.UseAuthorization();

app.MapControllers();

app.Run();

**OUTPUT**

