**Superset-Id: 6364597  
Name: SRINJOY PAUL**

**Week-5 WEB API HANDSON  
6. Kafka Integration with C#**

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

**Answer:**Steps:  
 1) Install Kafka and Zookeeper (Windows):  
 a)Start Zookeeper:  
 **zookeeper-server-start.bat ..\..\config\zookeeper.properties** b)Start Kafka Broker: **kafka-server-start.bat ..\..\config\server.properties**2) Create a Topic: **kafka-topics.bat --create --topic chat-topic --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1**3)Producer Application (Console App):  
a)Create a C# console app.b) Install NuGet: **Confluent.Kafka**c)Write producer logic:



1. Consumer Application (Console App):



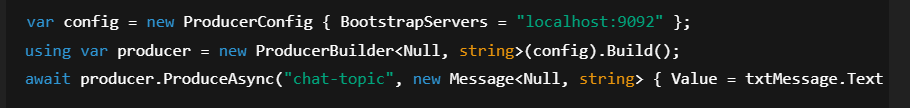
1. **Create a Chat Application using C# Windows Application using Kafka and consume the message in different client applications.**

**Answer:**Steps:  
1)Setup Windows Forms App:  
 a) Create a new C# Windows Forms App (.NET).  
 b) Install **Confluent.Kafka** via NuGet.

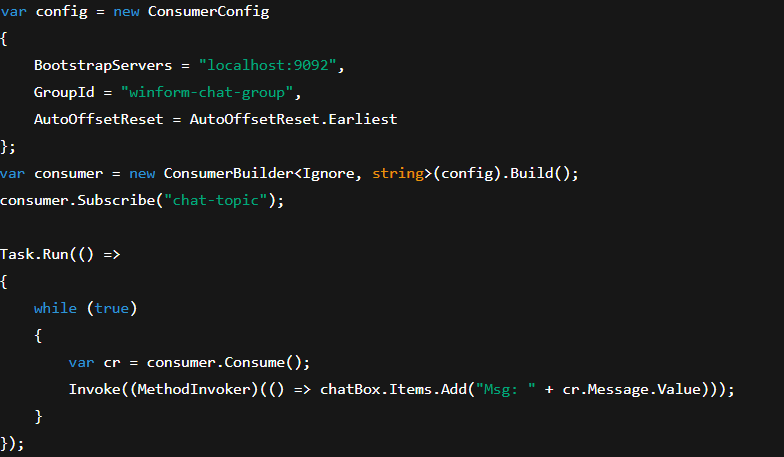
2)Design the UI:  
a) TextBox for message input  
b)ListBox/TextArea for displaying messages

c)Button to send messages

3)Producer Logic (On Button Click):



4)Consumer Logic (Background Task):



**Week-5 MICROSERVICES HANDSON**

**Q1) Implement JWT Authentication in ASP.NET Core Web API  
Answer:**Steps:

1. Create a new ASP.NET Core Web API project.

2. Add a `User` model and a login endpoint.

3. Generate a JWT token upon successful login.

4. Secure an endpoint using `[Authorize]`

Install NuGet Packages:

dotnet add package Microsoft.AspNetCore.Authentication.JwtBearer

appsettings.json:

{

"Jwt": {

"Key": "ThisIsASecretKeyForJwtToken",

"Issuer": "MyAuthServer",

"Audience": "MyApiUsers",

"DurationInMinutes": 60

}

}

Program.cs:

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();

AuthController.cs:

[ApiController]

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

public class AuthController : ControllerBase

{

[HttpPost("login")]

public IActionResult Login([FromBody] LoginModel model)

{

if (IsValidUser(model))

{

var token = GenerateJwtToken(model.Username);

return Ok(new { Token = token });

}

return Unauthorized();

}

private string GenerateJwtToken(string username)

{

var claims = new[]

{

new Claim(ClaimTypes.Name, username)

};

var key = new

SymmetricSecurityKey(Encoding.UTF8.GetBytes("ThisIsASecretKeyForJwtToken"));

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

var token = new JwtSecurityToken(

issuer: "MyAuthServer",

audience: "MyApiUsers",

claims: claims,

expires: DateTime.Now.AddMinutes(60),

signingCredentials: creds);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}