**Week 5: Hands-On**

**WebAPI Hands-On**

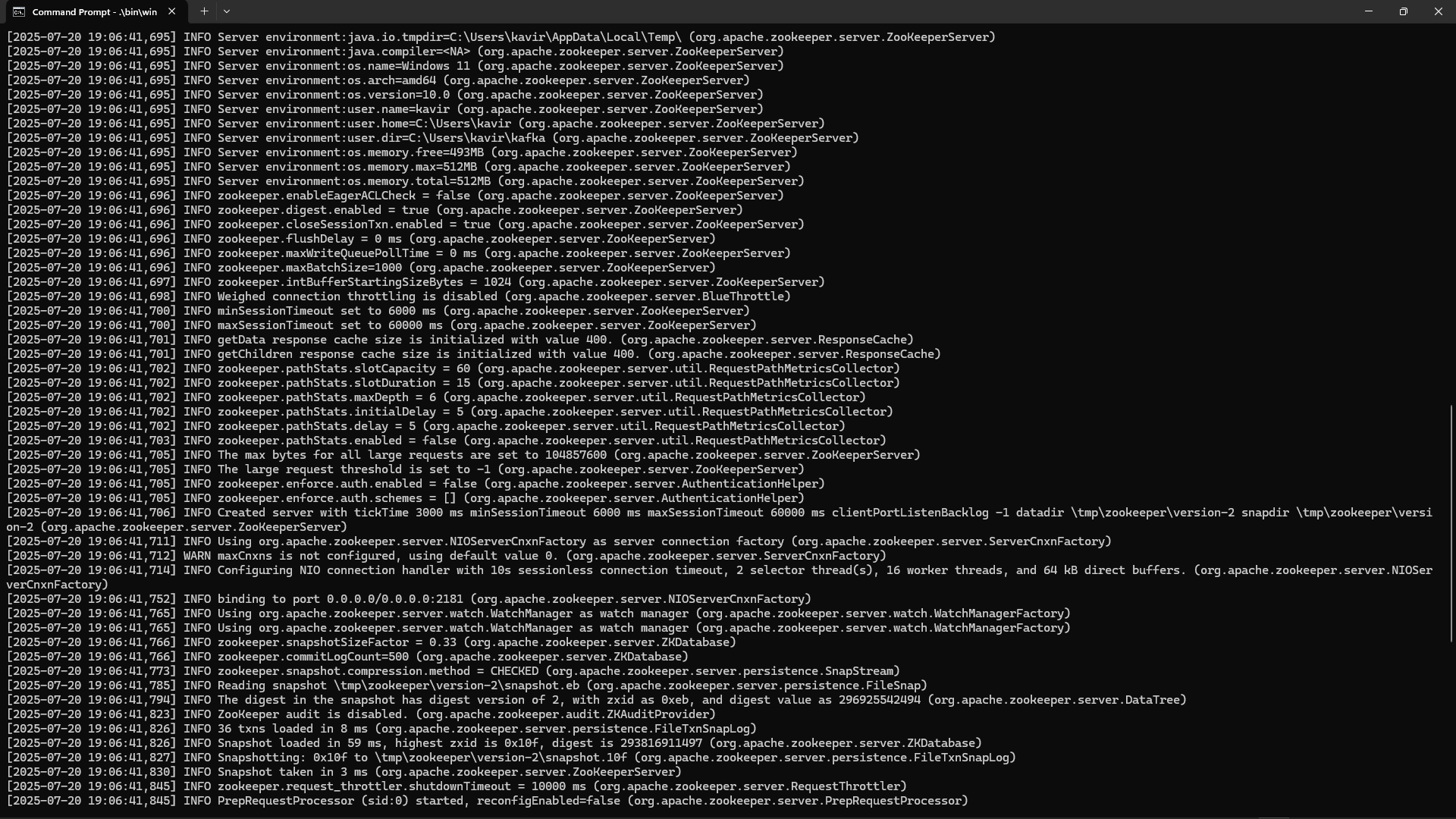
**Objective:**

* **Create a Chat Application which uses Kafka as a streaming platform and consume the chat messages in the command prompt.**
* **Create a Chat Application using C# Windows Application using Kafka and consume the message in different client applications**

**Setup:**

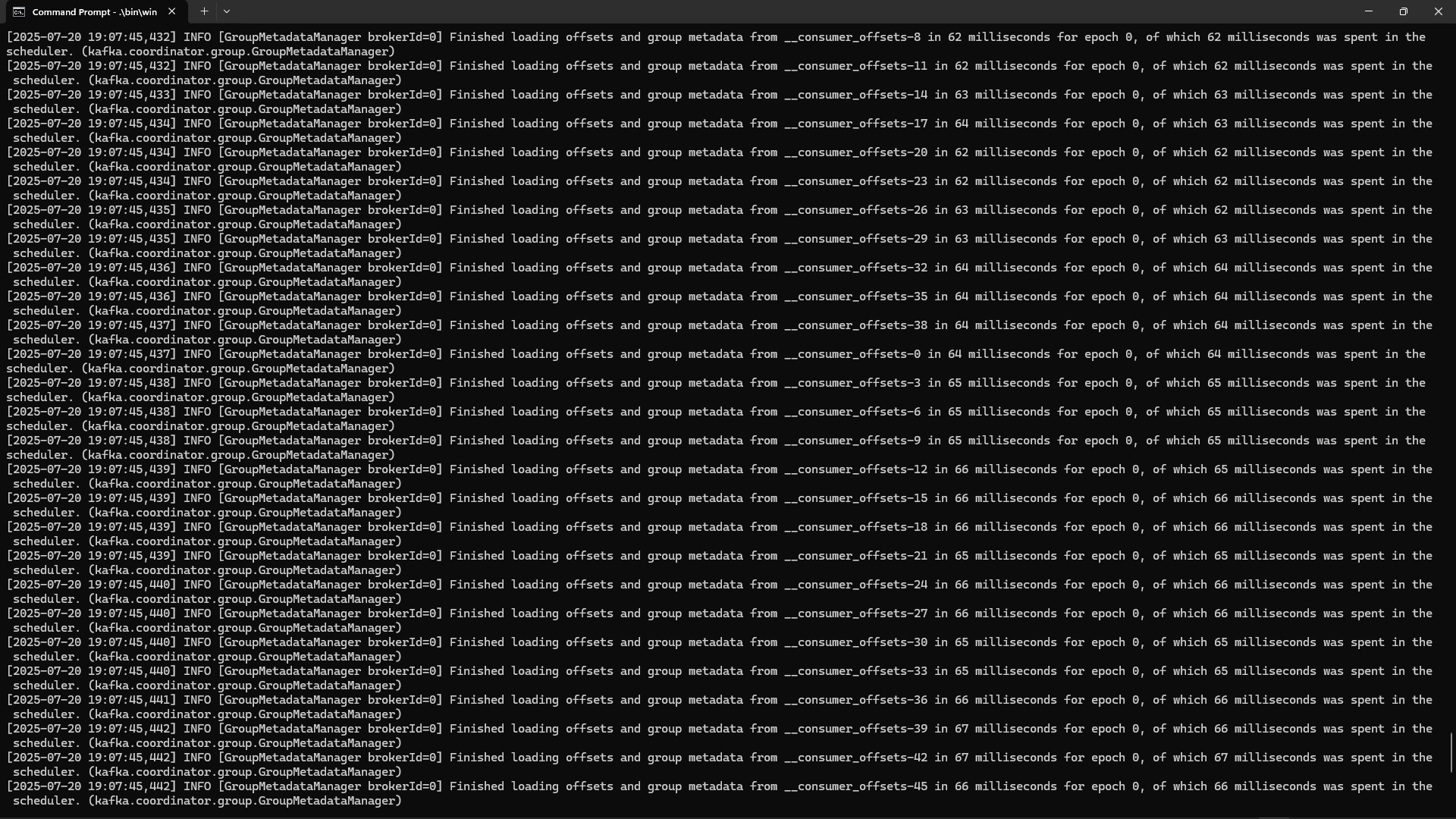
**Starting Zookeeper:**

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

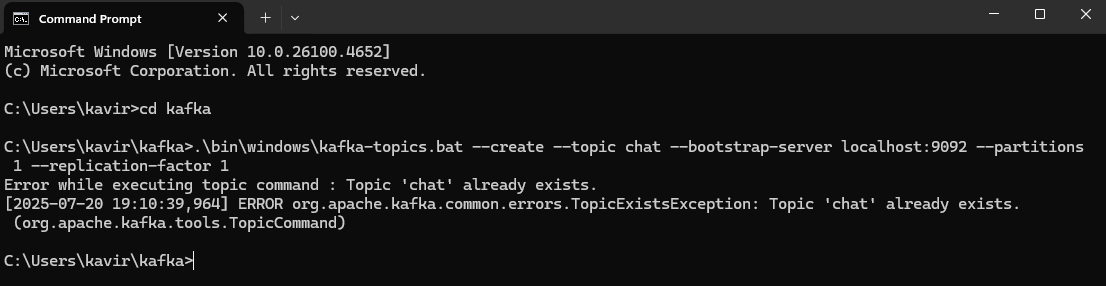


**Starting Kafka Brkoer:**

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



**Creation Topic Chat:**

****

**Project:**

**Kafka Producer:**

**Form1.cs**

using Confluent.Kafka;

using System;

using System.Windows.Forms;

namespace KafkaChat

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

string message = textBox1.Text;

var config = new ProducerConfig

{

BootstrapServers = "localhost:9092"

};

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

{

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

MessageBox.Show($"Message sent to Kafka Topic: {result.TopicPartitionOffset}");

}

}

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

{

this.Close();

}

}

}

**Form1.Designer.cs**

namespace KafkaChat

{

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

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

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

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

this.SuspendLayout();

this.label1.AutoSize = true;

this.label1.Location = new System.Drawing.Point(58, 45);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(250, 20);

this.label1.TabIndex = 0;

this.label1.Text = "Please Enter Your Message Here:";

this.textBox1.Location = new System.Drawing.Point(62, 77);

this.textBox1.Multiline = true;

this.textBox1.Name = "textBox1";

this.textBox1.Size = new System.Drawing.Size(300, 150);

this.textBox1.TabIndex = 1;

this.btnSend.Location = new System.Drawing.Point(244, 259);

this.btnSend.Name = "btnSend";

this.btnSend.Size = new System.Drawing.Size(75, 41);

this.btnSend.TabIndex = 2;

this.btnSend.Text = "Send";

this.btnSend.UseVisualStyleBackColor = true;

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

this.btnCancel.Location = new System.Drawing.Point(86, 259);

this.btnCancel.Name = "btnCancel";

this.btnCancel.Size = new System.Drawing.Size(75, 41);

this.btnCancel.TabIndex = 3;

this.btnCancel.Text = "Cancel";

this.btnCancel.UseVisualStyleBackColor = true;

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

this.AutoScaleDimensions = new System.Drawing.SizeF(9F, 20F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(1265, 755);

this.Controls.Add(this.btnCancel);

this.Controls.Add(this.btnSend);

this.Controls.Add(this.textBox1);

this.Controls.Add(this.label1);

this.Name = "Form1";

this.Text = "Kafka Chat Producer";

this.ResumeLayout(false);

this.PerformLayout();

}

private System.Windows.Forms.Label label1;

private System.Windows.Forms.TextBox textBox1;

private System.Windows.Forms.Button btnSend;

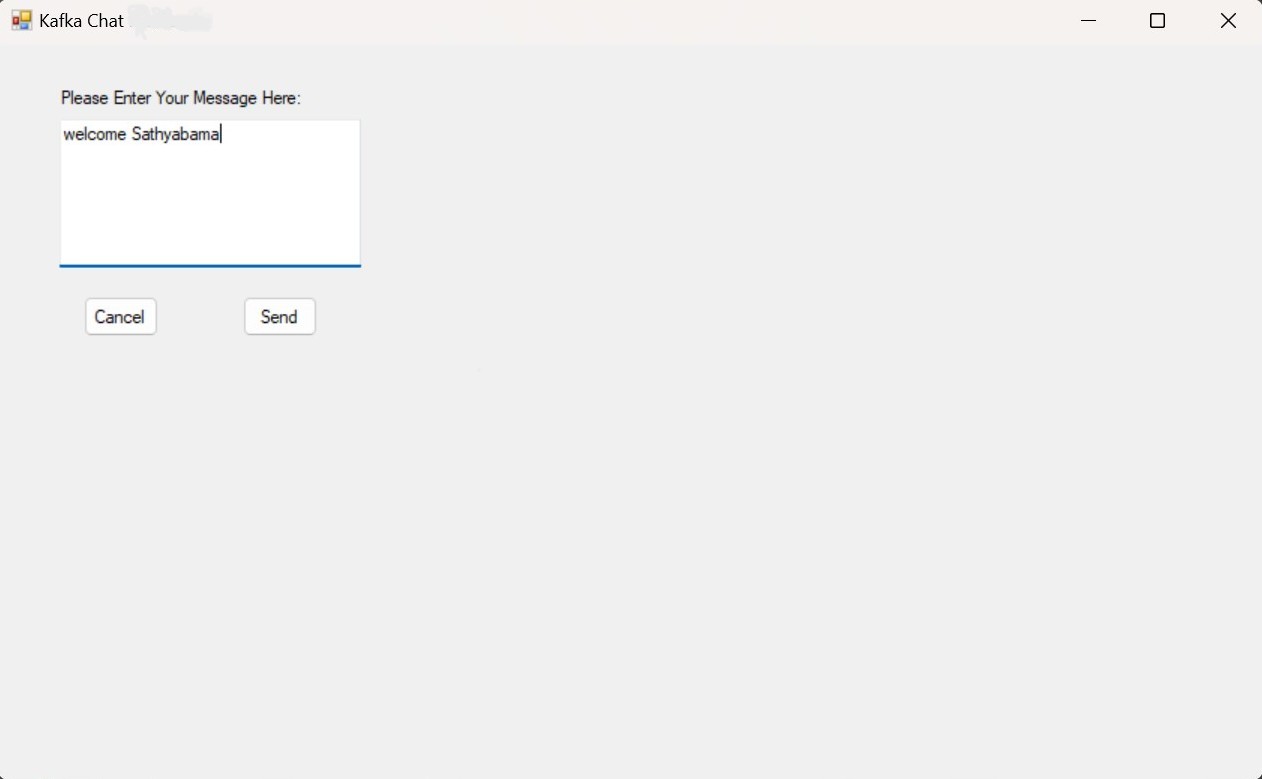
private System.Windows.Forms.Button btnCancel;

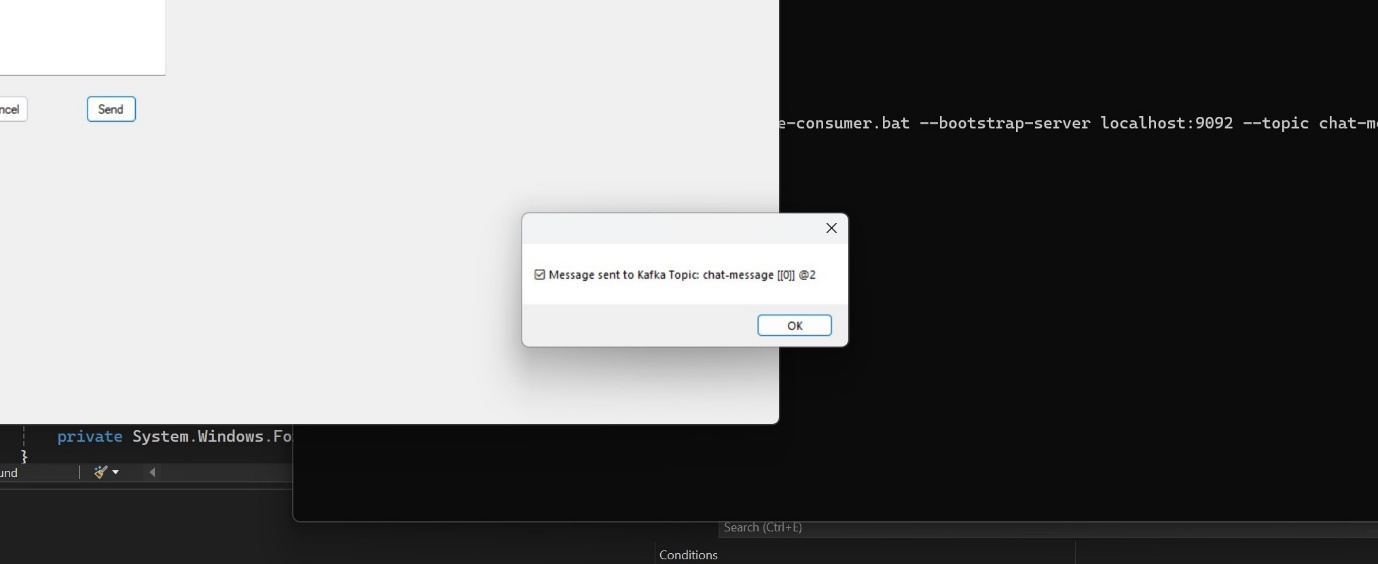
}

}

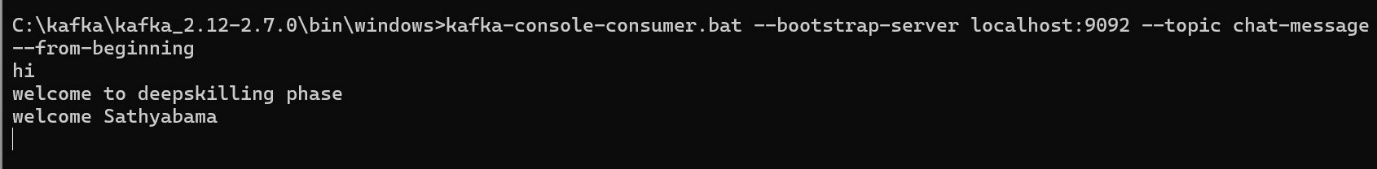
**Kafka Consumer:**

.\bin\windows\kafka-console-consumer.bat --topic chat --from-beginning --bootstrap-server localhost:9092



****

**Output:**

****

**---------------------------------------------------------------------------------------------------------**

**Microservices**

**Authentication and Authorization in ASP.NET Core Web API Microservices**

**Objective:**

Implement and secure ASP.NET Core Web API microservices using JWT authentication and role-based authorization. Learn how to generate and validate JWT tokens, protect endpoints with [Authorize], enforce role-based access, and handle token expiry with custom responses.

**Project:**

**appsettings.json**

{

"Jwt": {

"Key": "ThisIsAVerySecretKeyTokenForAdmin",

"Issuer": "MyAuthServer",

"Audience": "MyApiUsers",

"DurationInMinutes": 60

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

}

**Program.cs**

using JwtAuthDemo.Services;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo { Title = "JWT API", Version = "v1" });

c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

Name = "Authorization",

Type = SecuritySchemeType.Http,

Scheme = "Bearer",

BearerFormat = "JWT",

In = ParameterLocation.Header,

Description = "Enter 'Bearer' [space] and then your token in the text input below."

});

c.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

new string[] {}

}

});

});

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.GetType() == typeof(SecurityTokenExpiredException))

{

if (!context.Response.HasStarted)

{

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

}

}

return Task.CompletedTask;

}

};

});

builder.Services.AddAuthorization();

builder.Services.AddSingleton<TokenService>();

var app = builder.Build();

app.UseSwagger();

app.UseSwaggerUI();

app.UseDeveloperExceptionPage();

app.UseHttpsRedirection();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

**Controllers/AdminController.cs**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace JwtAuthDemo.Controllers

{

[ApiController]

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

public class AdminController : ControllerBase

{

[HttpGet("dashboard")]

[Authorize(Roles = "Admin")]

public IActionResult GetDashboard()

{

return Ok("Welcome Admin!");

}

}

}

**Controllers/AuthControler.cs**

using JwtAuthDemo.Models;

using JwtAuthDemo.Services;

using Microsoft.AspNetCore.Mvc;

namespace JwtAuthDemo.Controllers

{

[ApiController]

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

public class AuthController : ControllerBase

{

private readonly TokenService \_tokenService;

public AuthController(TokenService tokenService)

{

\_tokenService = tokenService;

}

[HttpPost("login")]

public IActionResult Login([FromBody] LoginModel model)

{

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

{

var token = \_tokenService.GenerateToken(model.Username, "Admin");

return Ok(new { Token = token });

}

else

{

return Unauthorized("Invalid credentials");

}

}

}

}

**Controllers/SecureController.cs**

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace JwtAuthDemo.Controllers

{

[ApiController]

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

public class SecureController : ControllerBase

{

[HttpGet("data")]

[Authorize]

public IActionResult GetSecureData()

{

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

}

}

}

**Services/TokenService.cs**

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

using Microsoft.IdentityModel.Tokens;

namespace JwtAuthDemo.Services

{

public class TokenService

{

private readonly IConfiguration \_configuration;

public TokenService(IConfiguration configuration)

{

\_configuration = configuration;

}

public string GenerateToken(string username, string role)

{

var claims = new[]

{

new Claim(ClaimTypes.Name, username),

new Claim(ClaimTypes.Role, role)

};

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(double.Parse(\_configuration["Jwt:DurationInMinutes"])),

signingCredentials: creds

);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

}

**Models/User.cs**

namespace JwtAuthDemo.Models

{

public class LoginModel

{

public string Username { get; set; }

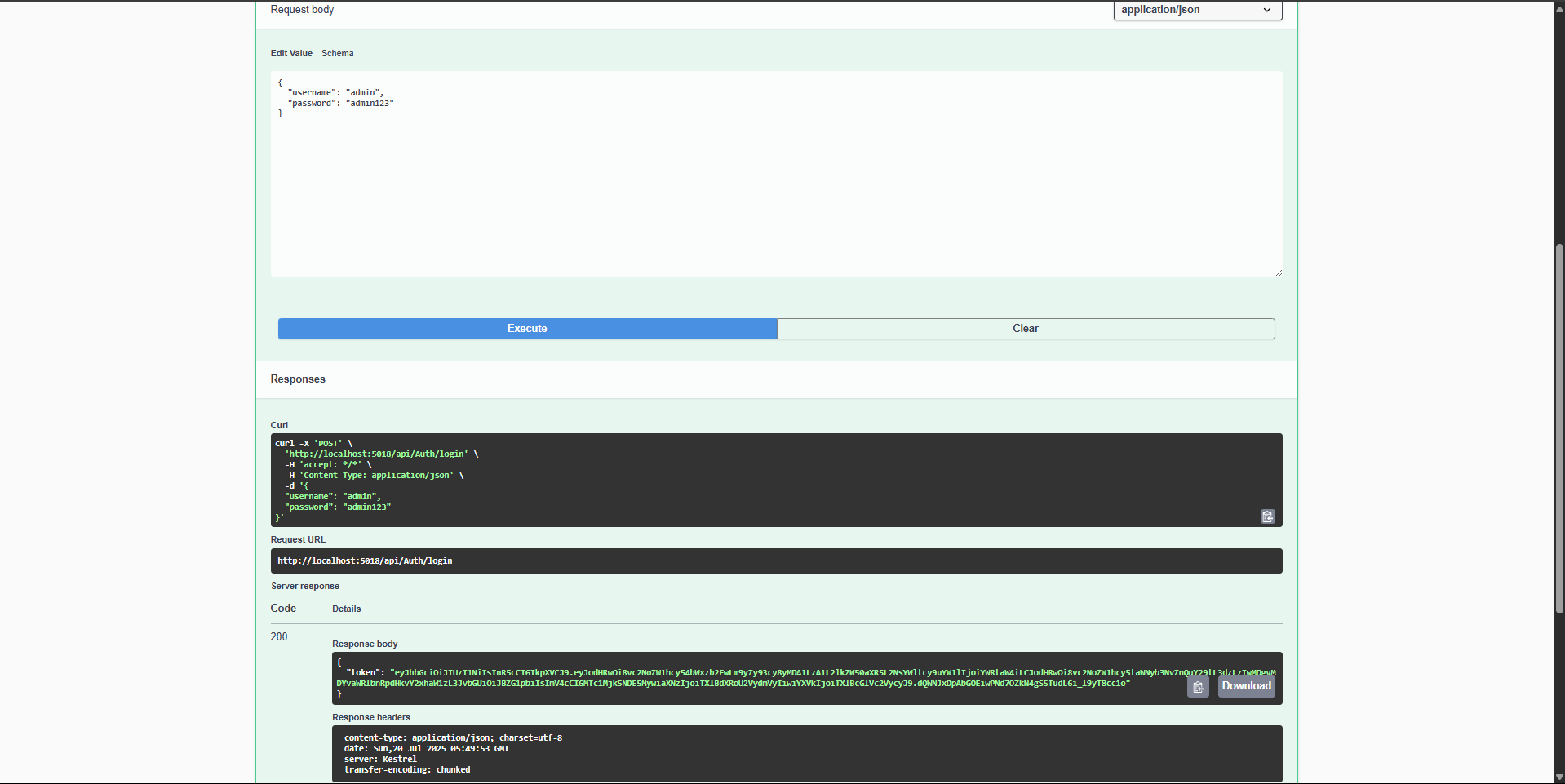
public string Password { get; set; }

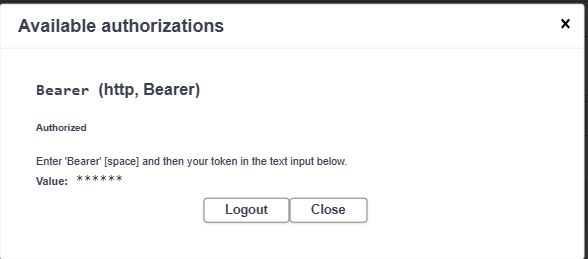
}

}

**Output:**

**Using Swagger:**





**Using POSTMAN:**

