Microservices

Question 1: 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.

Steps:

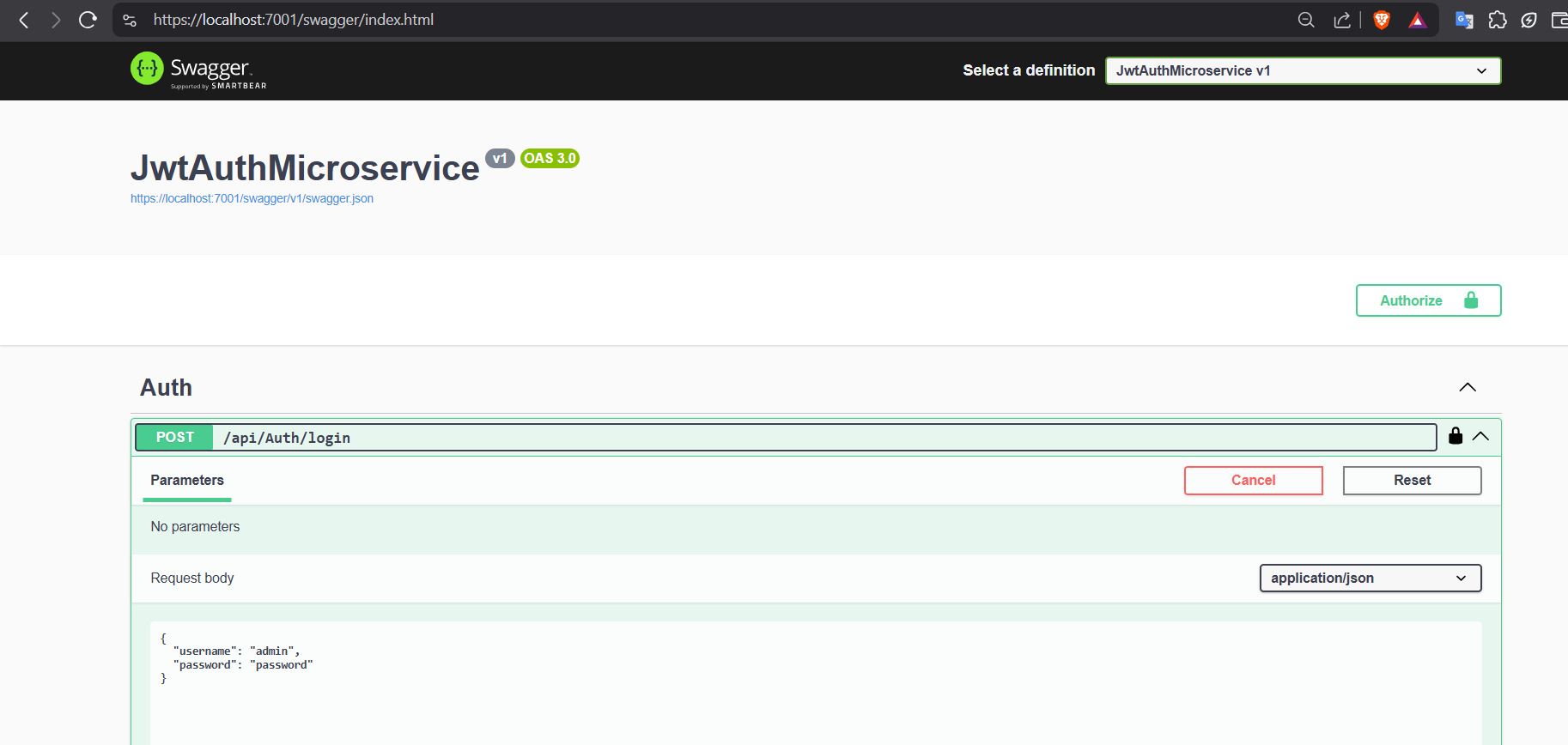
1. Create a new ASP.NET Core Web API project.

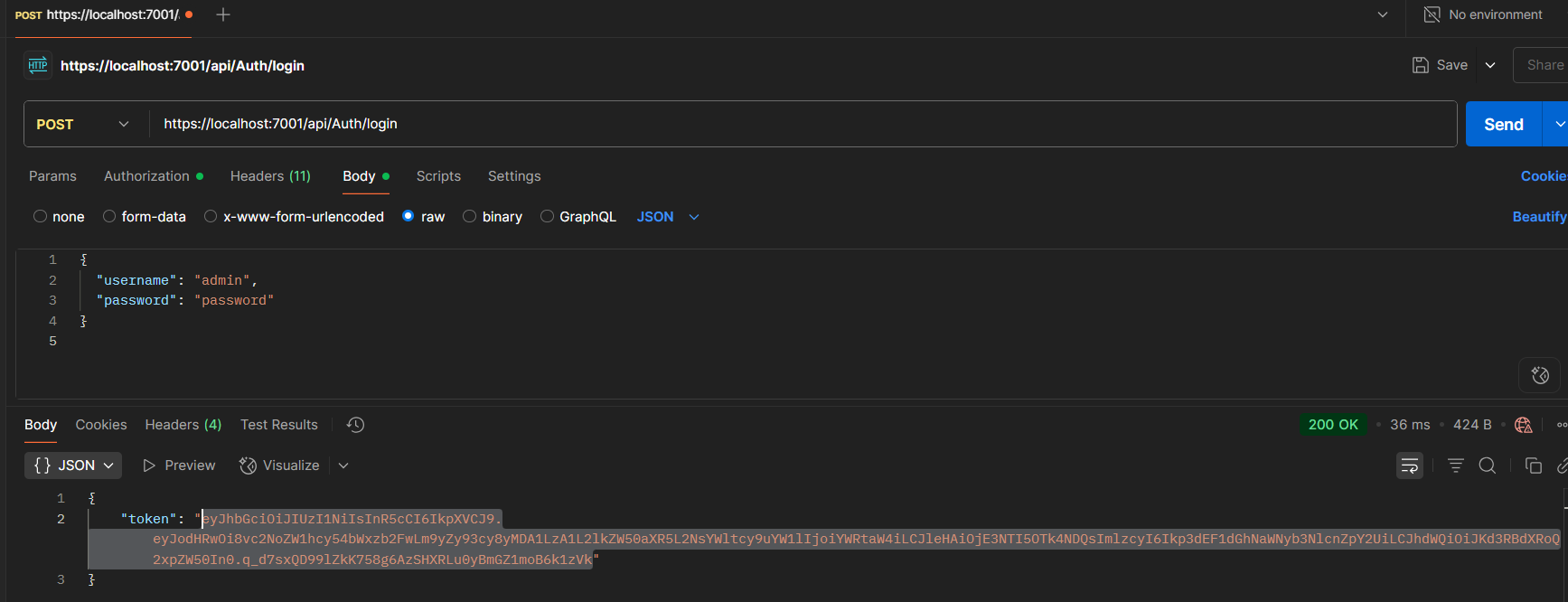
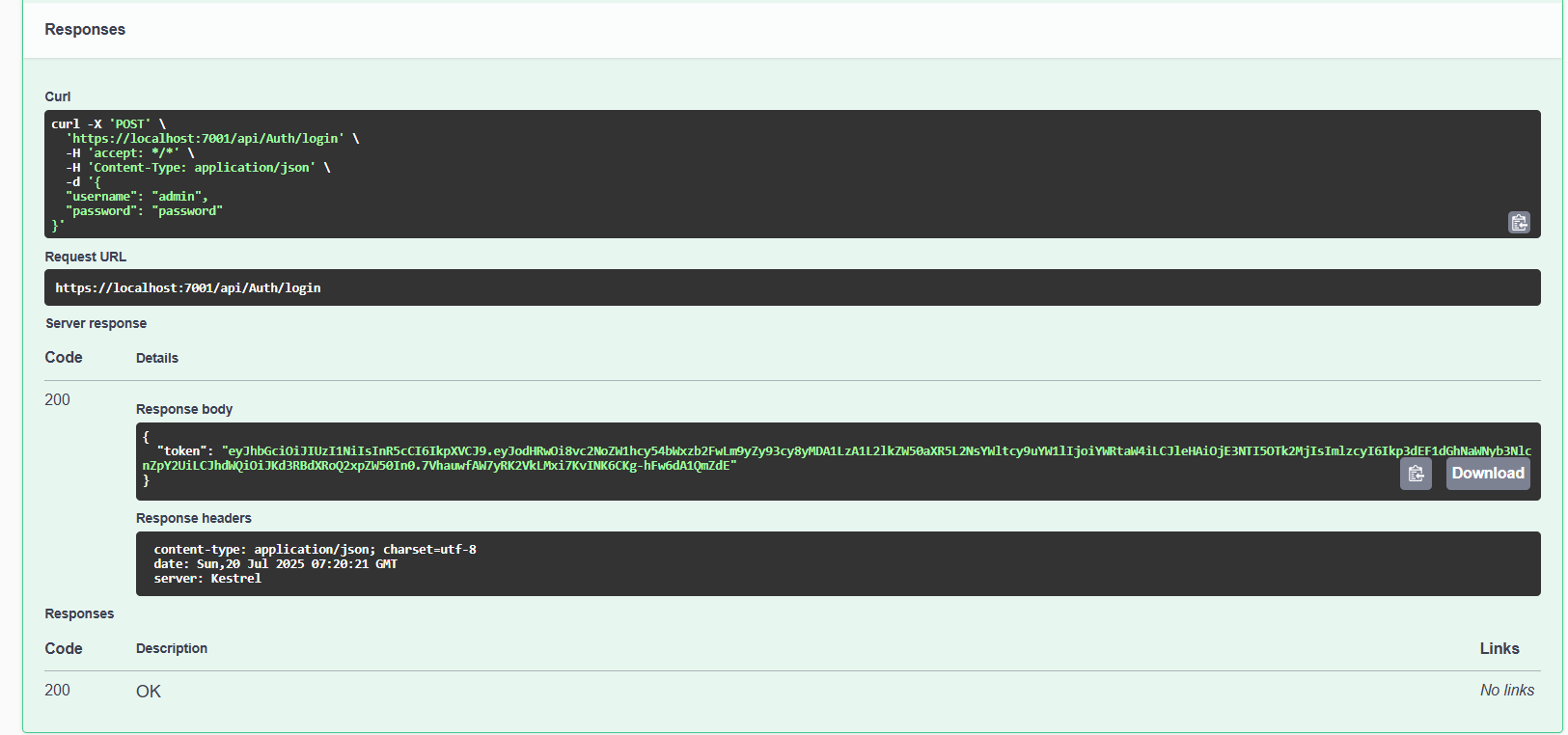
2. Add a `User` model and a login endpoint.

3. Generate a JWT token upon successful login.

4. Secure an endpoint using `[Authorize]`.

OUTPUT:





CODE:

Appsettings.json

{

"Jwt": {

"Key": "xA9uL8n5WsY3pM0dKvZrXe4TqBo1GjJc",

"Issuer": "JwtAuthMicroservice",

"Audience": "JwtAuthClient",

"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);

// 1. Controllers + Swagger

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new() { Title = "JwtAuthMicroservice", Version = "v1" });

var scheme = new Microsoft.OpenApi.Models.OpenApiSecurityScheme

{

Name = "JWT Authentication",

Description = "Enter JWT Bearer token \*\*\_only\_\*\*",

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

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

Scheme = JwtBearerDefaults.AuthenticationScheme,

BearerFormat = "JWT",

Reference = new Microsoft.OpenApi.Models.OpenApiReference

{

Id = JwtBearerDefaults.AuthenticationScheme,

Type = Microsoft.OpenApi.Models.ReferenceType.SecurityScheme

}

};

c.IncludeXmlComments(Path.Combine(AppContext.BaseDirectory, "JwtAuthMicroservice.xml"));

c.AddSecurityDefinition("Bearer", scheme);

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

{

{ scheme, Array.Empty<string>() }

});

});

// 2. JWT Authentication

var jwt = builder.Configuration.GetSection("Jwt");

builder.Services

.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddJwtBearer(options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = jwt["Issuer"],

ValidAudience = jwt["Audience"],

IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwt["Key"]!))

};

});

builder.Services.AddAuthorization();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

if (string.IsNullOrEmpty(jwt["Key"]))

{

throw new Exception("JWT Key not found in configuration. Check appsettings.json");

}

Console.WriteLine($"JWT Key: {builder.Configuration["Jwt:Key"]}");

app.UseHttpsRedirection();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

AuthController.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

using JwtAuthMicroservice.Models; // Adjust based on your project namespace

namespace JwtAuthMicroservice.Controllers // Match the folder structure

{

[ApiController]

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

public class AuthController : ControllerBase

{

[HttpPost("login")]

public IActionResult Login([FromBody] LoginModel model)

{

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

{

var token = GenerateJwtToken(model.Username);

return Ok(new { token });

}

return Unauthorized();

}

private string GenerateJwtToken(string username)

{

var claims = new[]

{

new Claim(ClaimTypes.Name, username)

};

var key = new SymmetricSecurityKey(

Encoding.UTF8.GetBytes("xA9uL8n5WsY3pM0dKvZrXe4TqBo1GjJc")); // match appsettings key

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

var token = new JwtSecurityToken(

issuer: "JwtAuthMicroservice",

audience: "JwtAuthClient",

claims: claims,

expires: DateTime.UtcNow.AddMinutes(60),

signingCredentials: creds);

return new JwtSecurityTokenHandler().WriteToken(token);

}

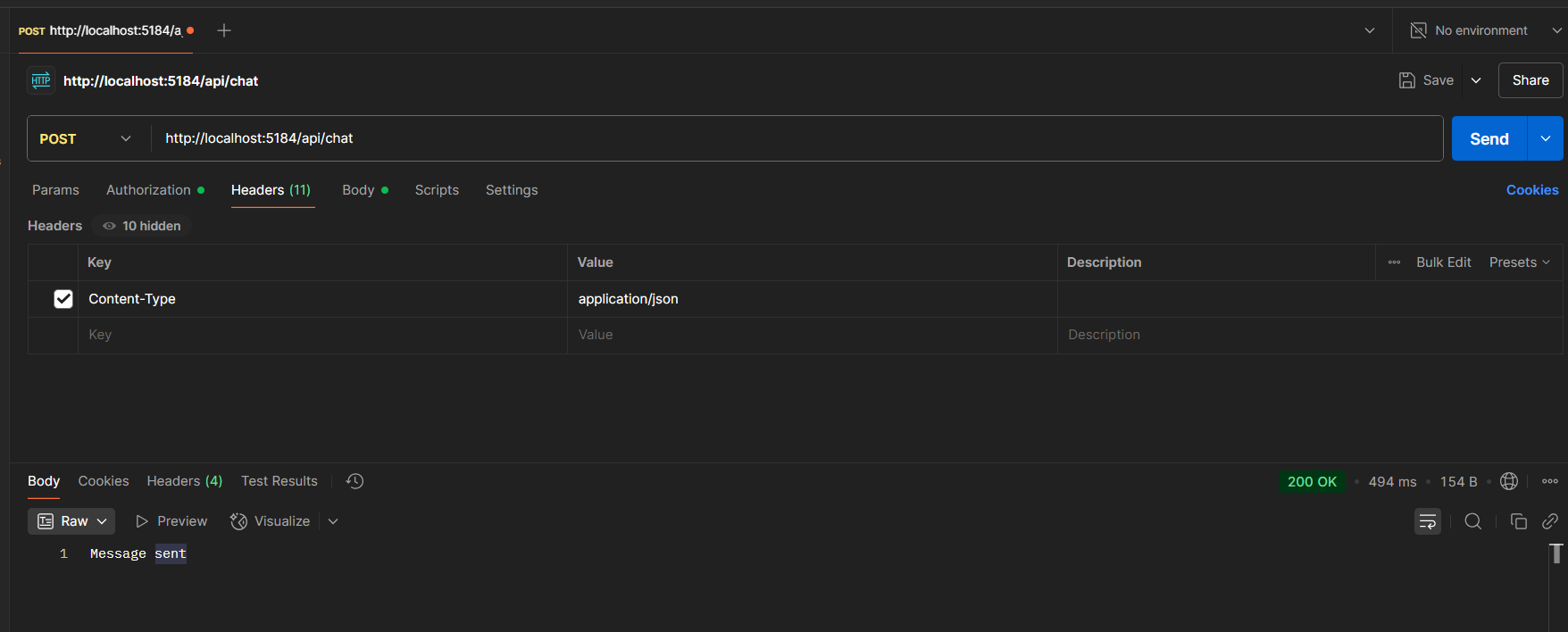
}

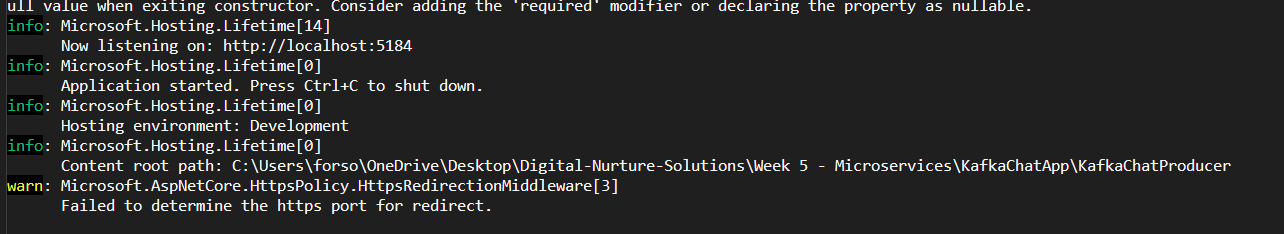
}

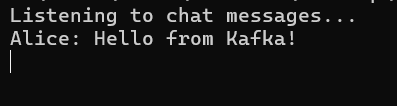
WebApi\_HandsOn

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

Output:







Code:

ChatController.cs

using Confluent.Kafka;

using KafkaChatProducer.Models;

using Microsoft.AspNetCore.Mvc;

using System.Text.Json;

namespace KafkaChatProducer.Controllers

{

[ApiController]

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

public class ChatController : ControllerBase

{

[HttpPost]

public IActionResult SendMessage([FromBody] ChatMessage message)

{

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

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

{

var jsonMessage = JsonSerializer.Serialize(message);

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

producer.Flush(TimeSpan.FromSeconds(5));

}

return Ok("Message sent");

}

}

}

ChatMessage.cs

namespace KafkaChatProducer.Models

{

public class ChatMessage

{

public string Sender { get; set; }

public string Message { get; set; }

}

}

Program.cs

namespace KafkaChatProducer

{

public class Program

{

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

}

}

}