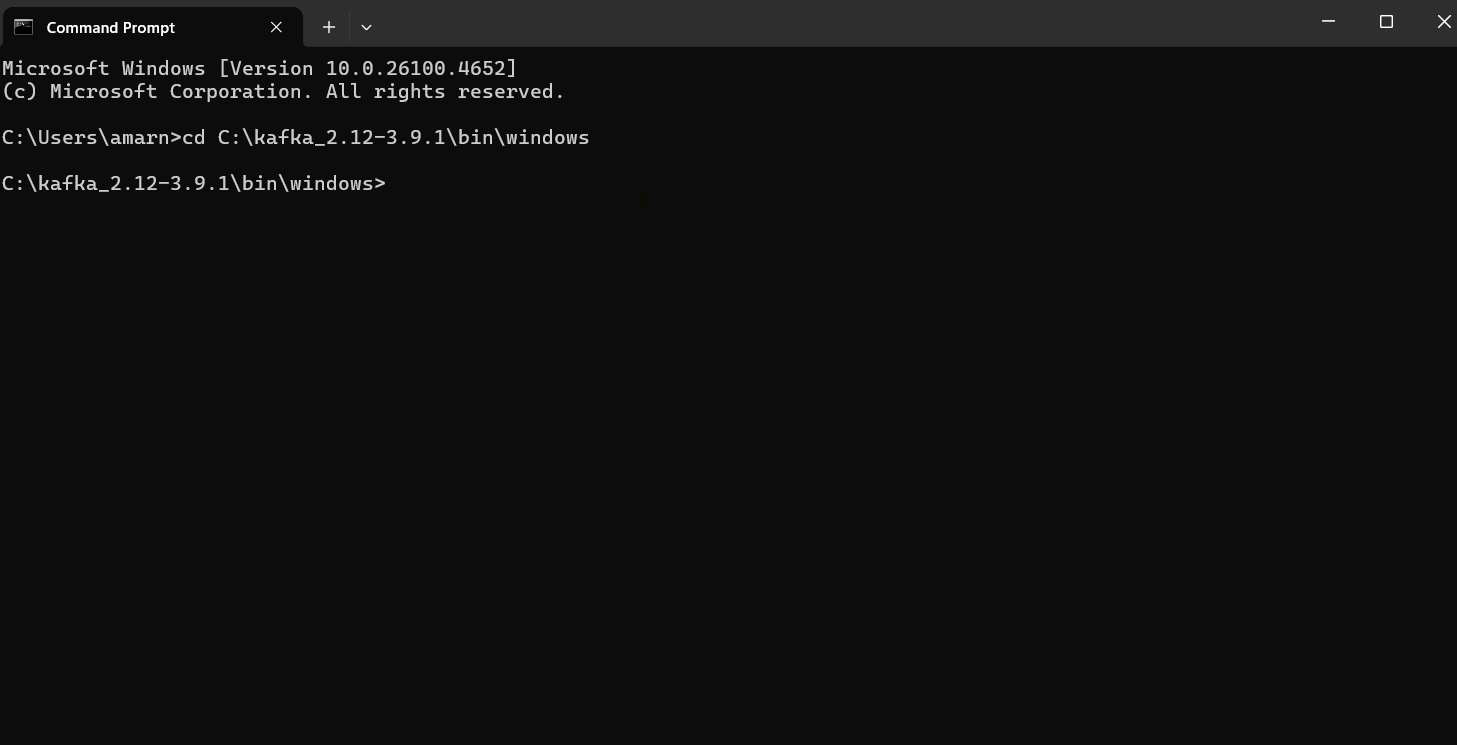
**Week 5 :**

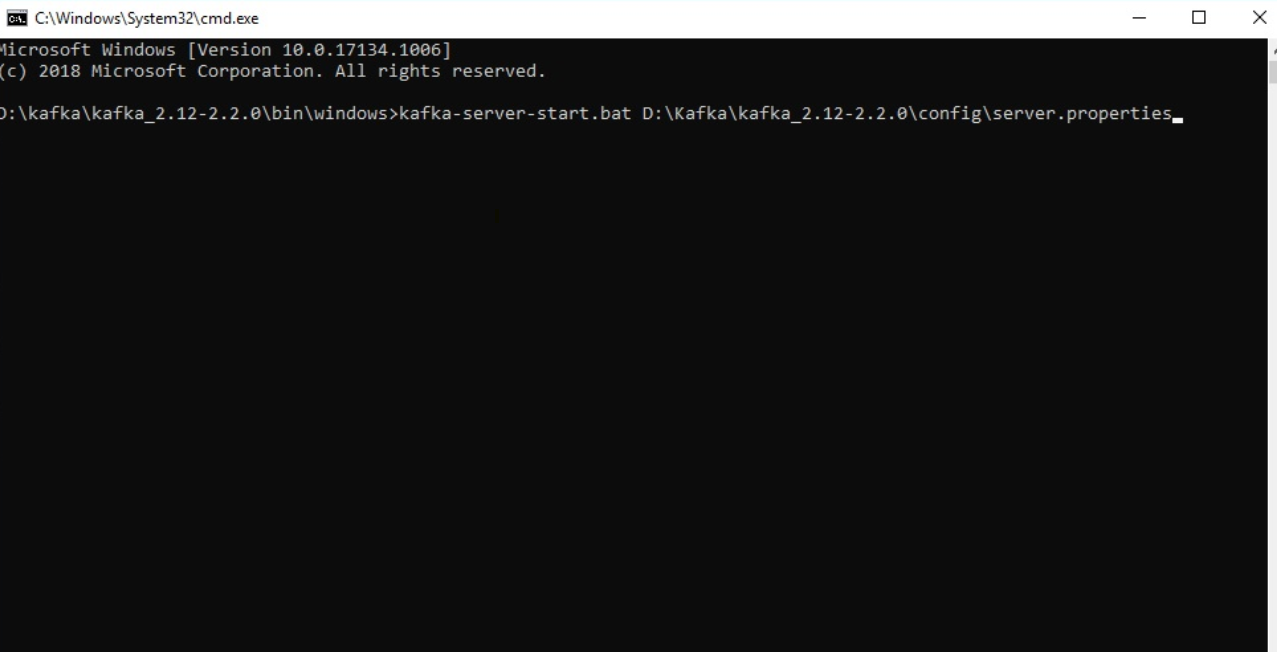
**Kafka Integration with C#:**

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

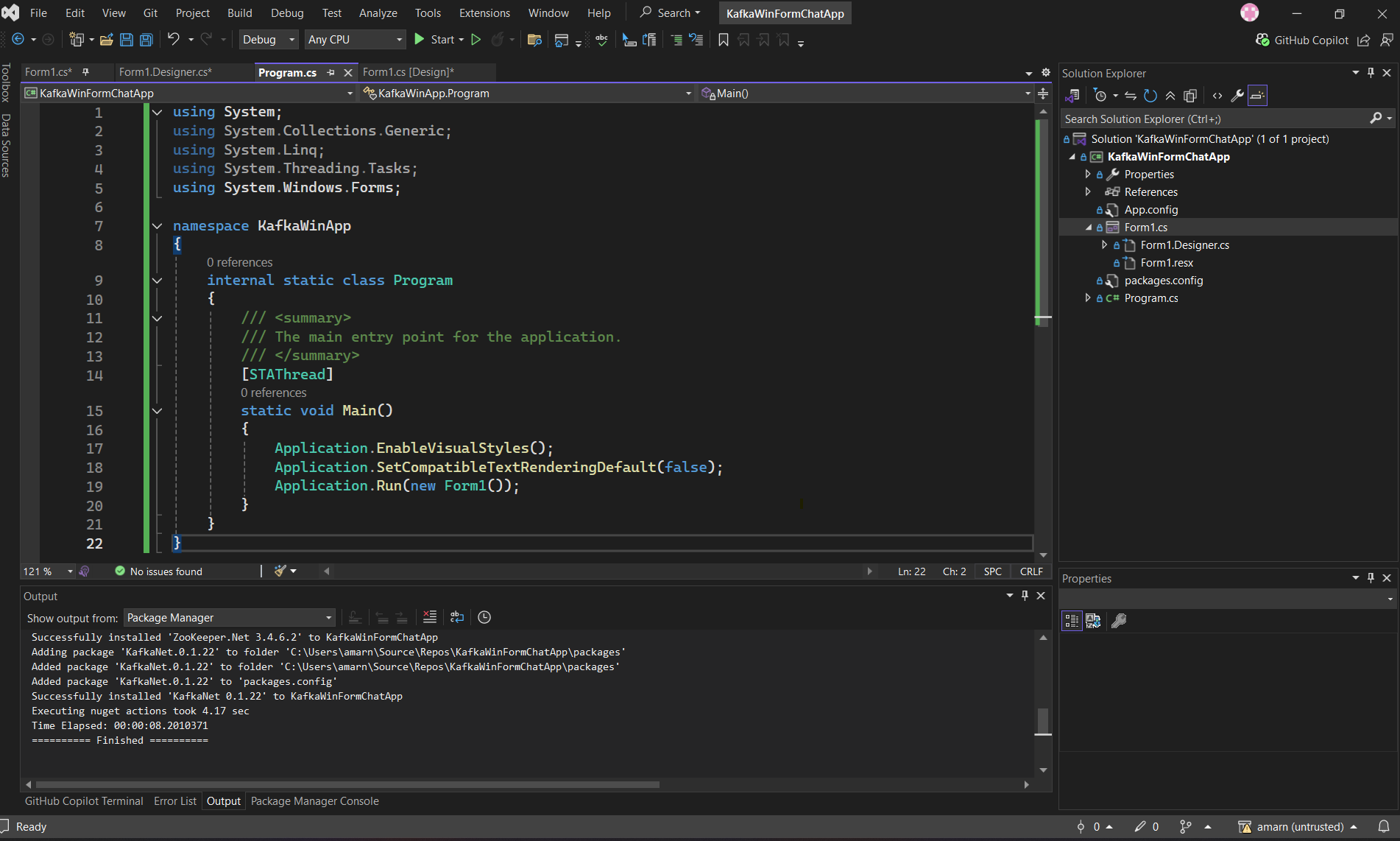
Installing kafka services and zookeeper services .



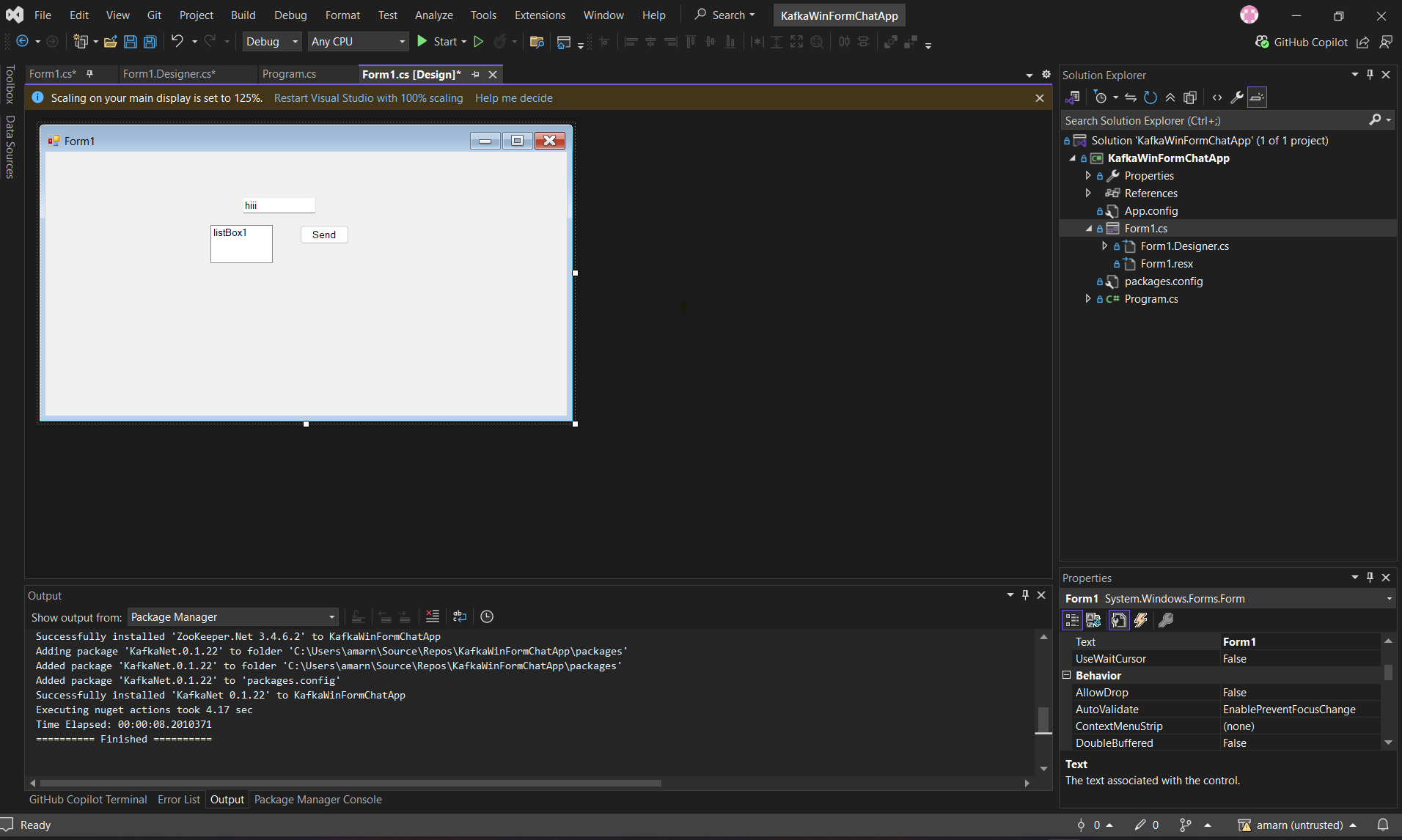
**Creating a message topic:** configuration

****

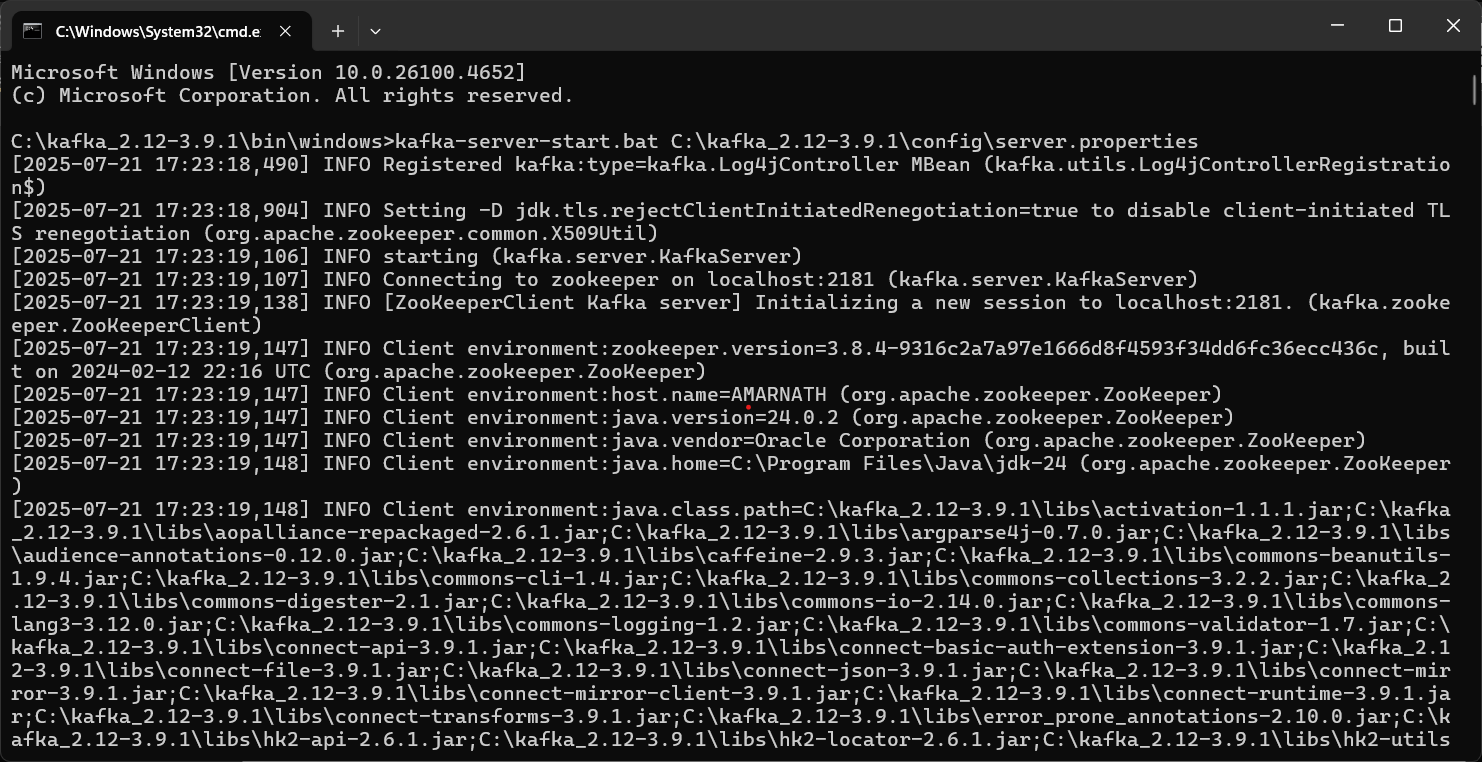
**Program.cs:**

****

**Form1.cs[design] :**

****

**Output 1 :**

****

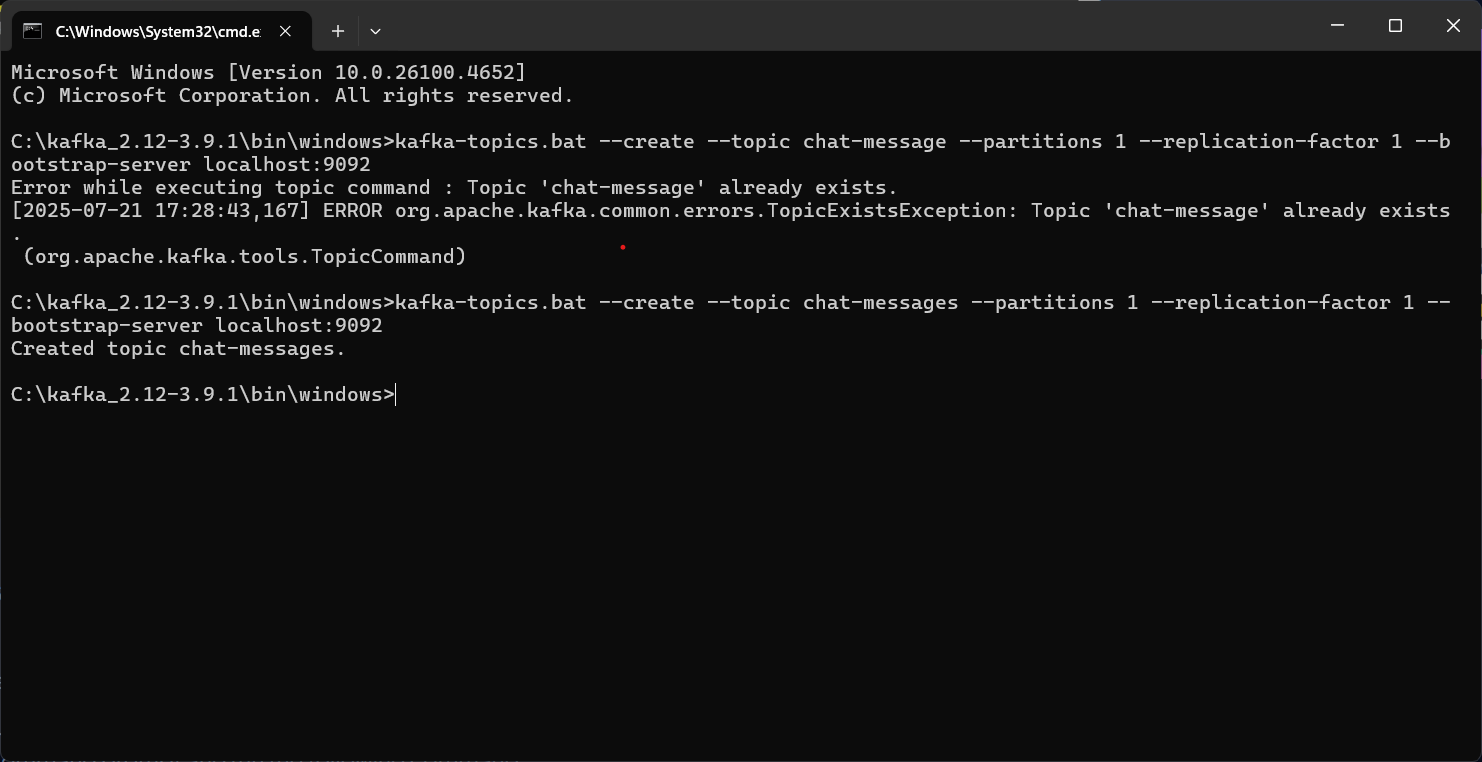
**Initializing zookeeper:**

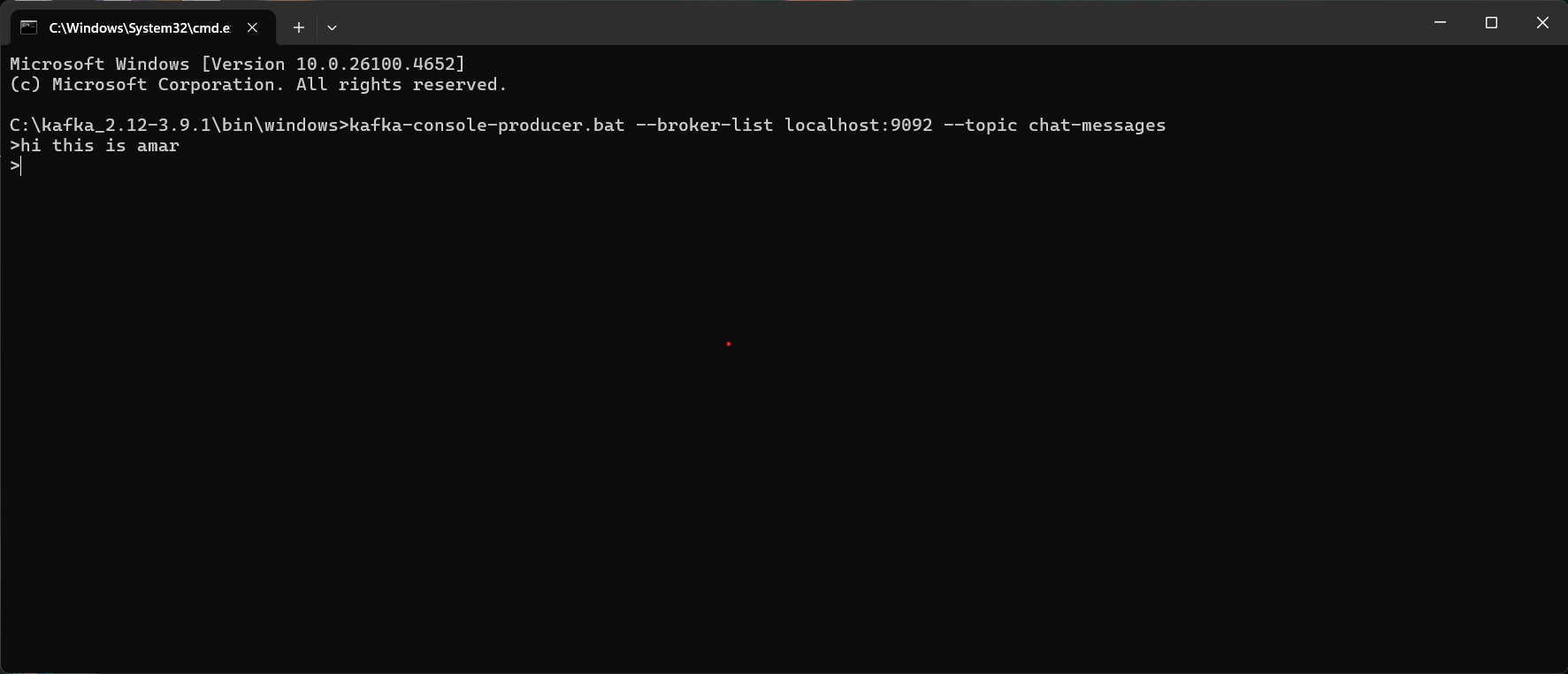
****

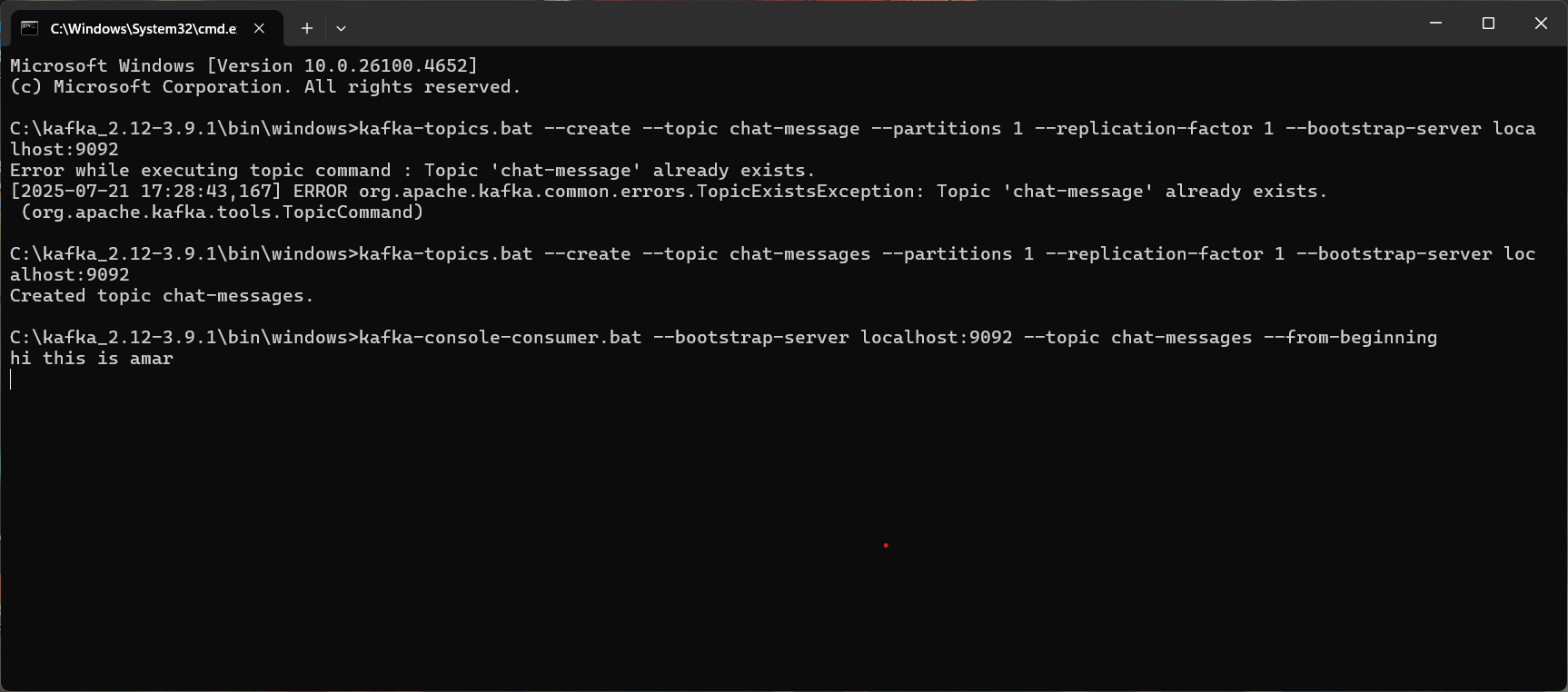
**Message output :**

**Chat message:**

creation chat message

****

****

****

**Microservices – JWT**

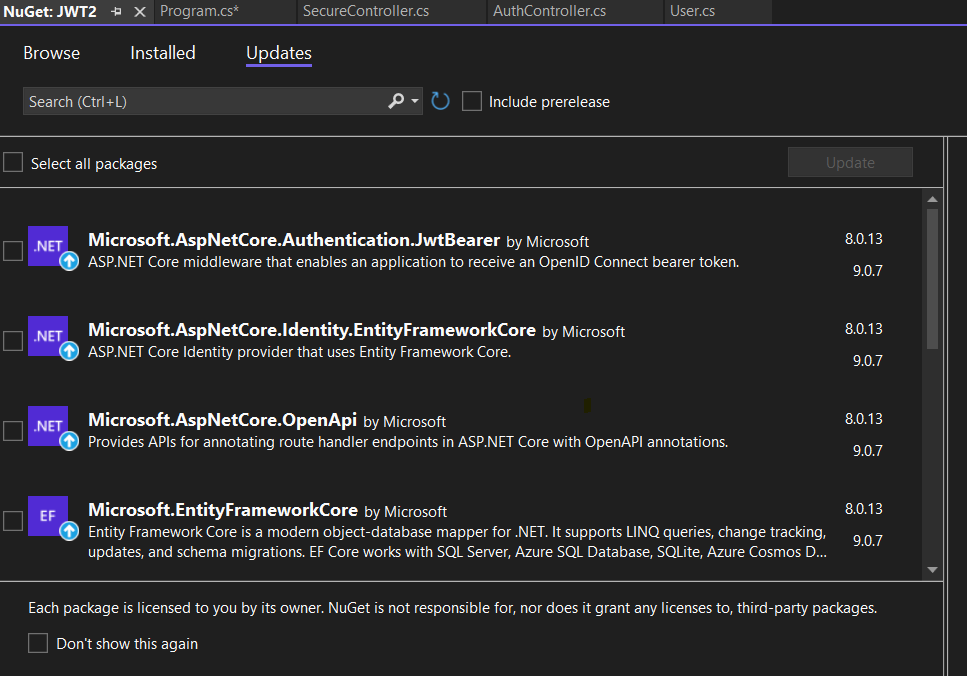
**Implementing JWT Authentication in ASP.NET Core Web API.**

**Steps:**

1. Creating new ASP.NET Core Web API project.

After creating the new web api .net project install

All the required packages .



1. **Adding a `User` model and a login endpoint.**

Add model class :

namespace JWT2.models

{

public class User

{

public string Username { get; set; }

public string Password { get; set; }

}

}namespace JWT2.models

{

public class User

{

public string Username { get; set; }

public string Password { get; set; }

}

}

**3.Generating a JWT token upon successful login.**

**appsettings.json:**

{

"Jwt": {

"Key": "ThisIsASecretKeyForJwtToken",

"Issuer": "MyAuthServer",

"Audience": "MyApiUsers",

"DurationInMinutes": 60 }

}

**Program.cs**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

namespace JWT2

{

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();

// JWT Authentication configuration

var key = Encoding.ASCII.GetBytes("YourSuperSecretKey123!");

builder.Services.AddAuthentication(options =>

{

options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

})

.AddJwtBearer(options =>

{

options.RequireHttpsMetadata = false;

options.SaveToken = true;

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuerSigningKey = true,

IssuerSigningKey = new SymmetricSecurityKey(key),

ValidateIssuer = false,

ValidateAudience = false

};

});

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

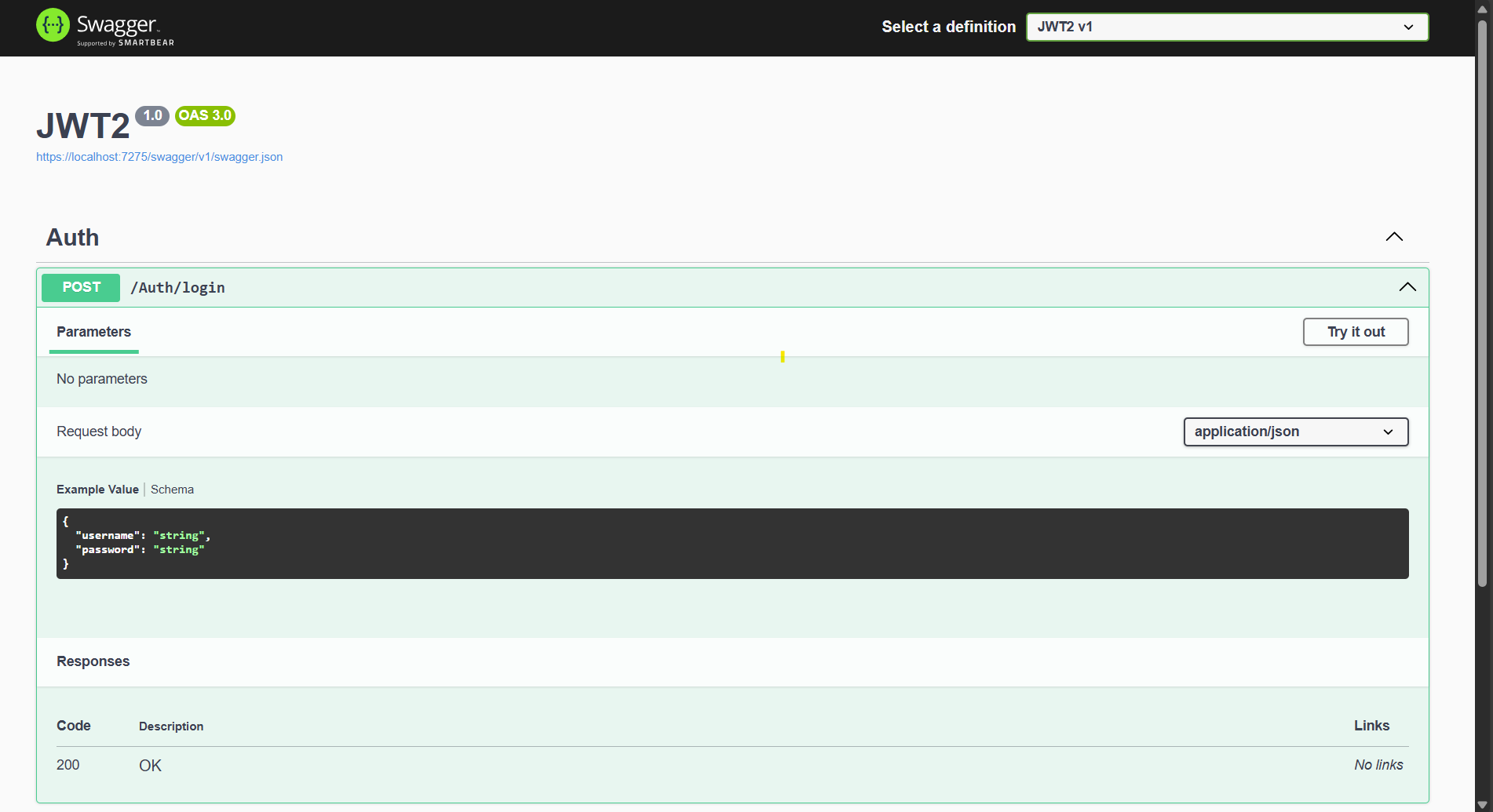
app.Run();

}

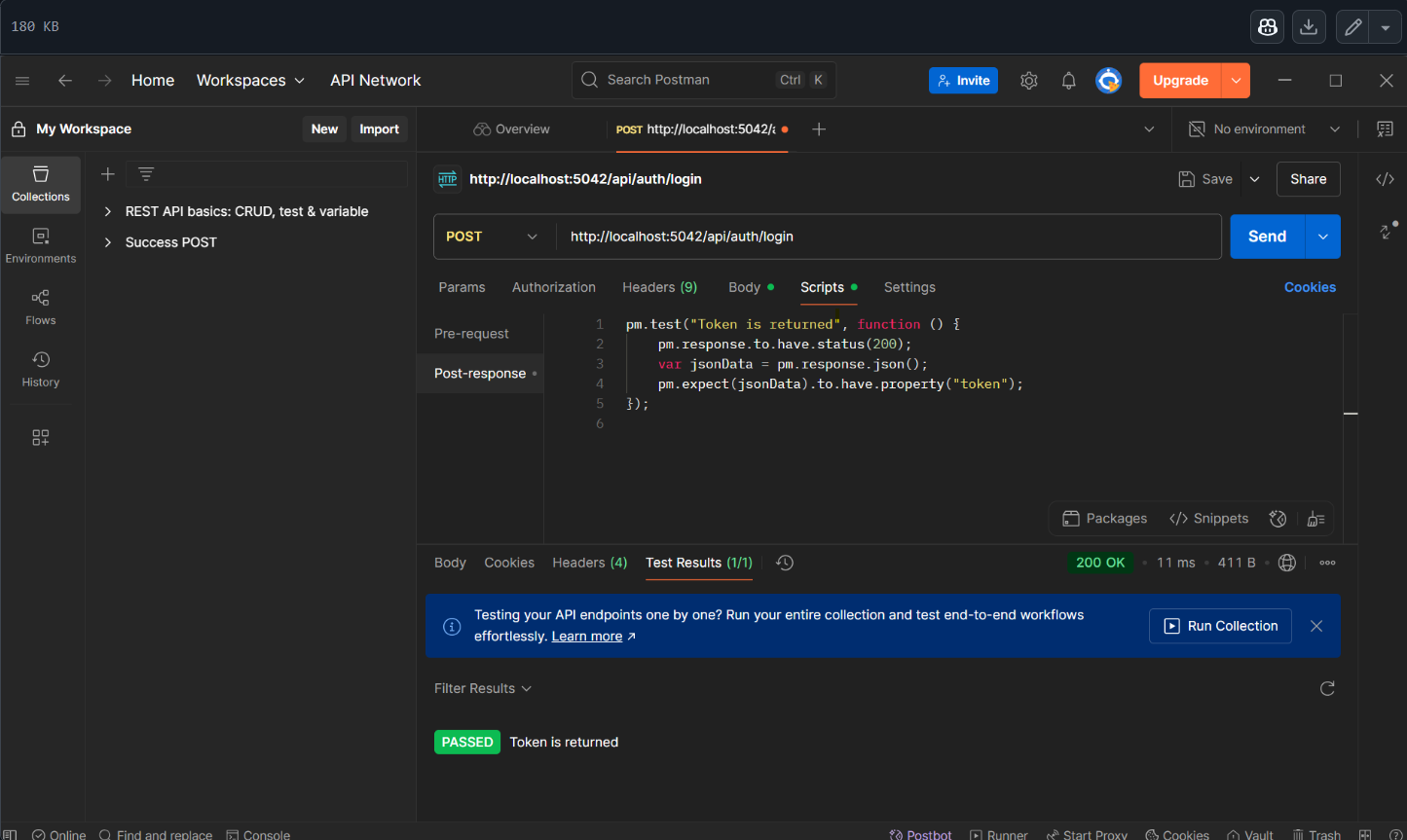
}

}

**4.Securing an endpoint using `[Authorize]`:**

****

**Testing using swagger and postman:**

****