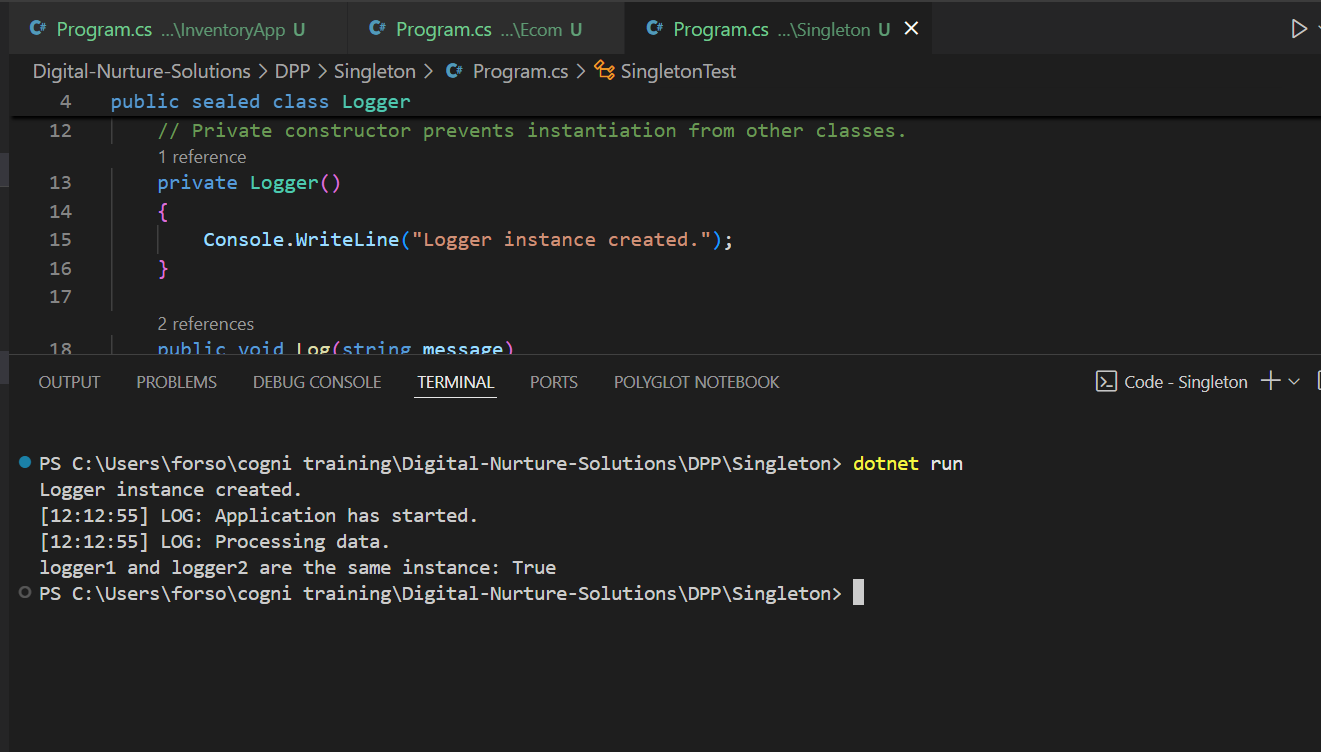
**Exercise 1: Implementing the Singleton Pattern**

**Scenario:**

You need to ensure that a logging utility class in your application has only one instance throughout the application lifecycle to ensure consistent logging.

**OUTPUT:**

****

**CODE:**

using System;

public sealed class Logger

{

    private static readonly Lazy<Logger> lazyInstance = new Lazy<Logger>(() => new Logger());

    public static Logger Instance => lazyInstance.Value;

    private Logger()

    {

        Console.WriteLine("Logger instance created.");

    }

    public void Log(string message)

    {

        Console.WriteLine($"[{DateTime.Now:HH:mm:ss}] LOG: {message}");

    }

}

public class SingletonTest

{

    public static void Main(string[] args)

    {

        Logger logger1 = Logger.Instance;

        logger1.Log("Application has started.");

        Logger logger2 = Logger.Instance;

        logger2.Log("Processing data.");

        Console.WriteLine($"logger1 and logger2 are the same instance: {object.ReferenceEquals(logger1, logger2)}");

    }

}