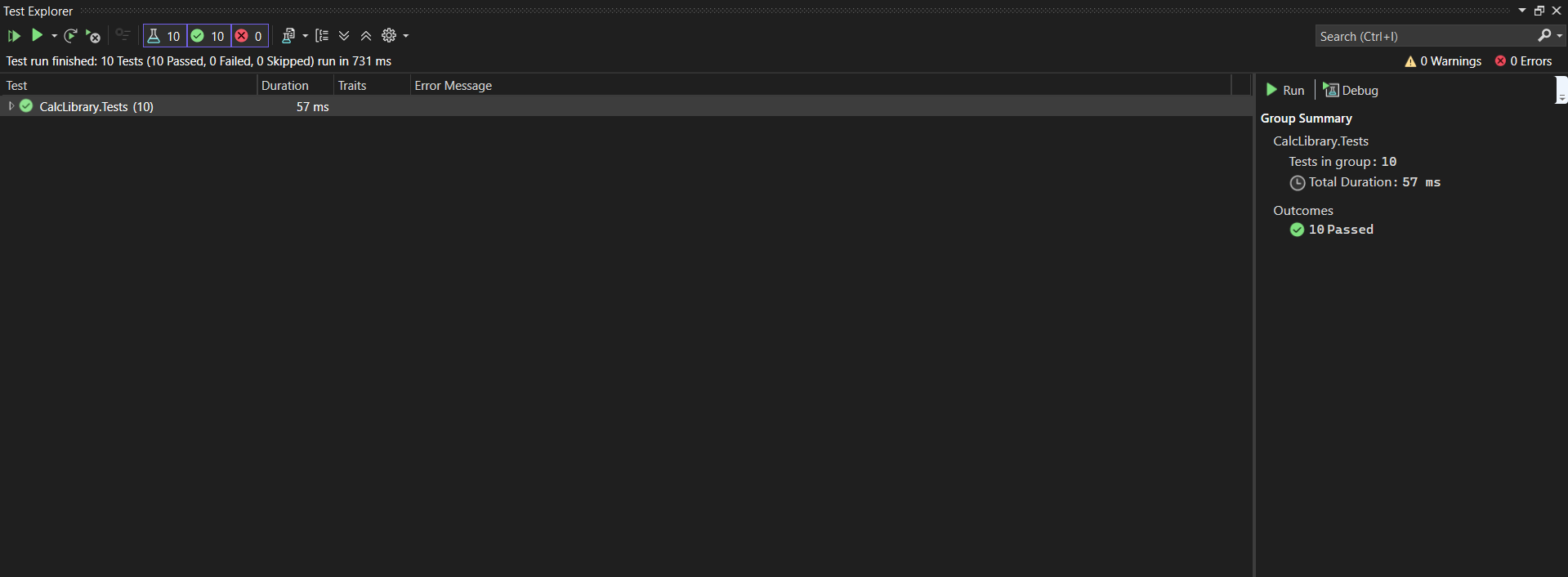
**TestFixture & Test**

Please download the application available [here](https://cognizantonline.sharepoint.com/:u:/r/sites/GTP-Solutions/Gencsharepath/Shared%20Documents/Internship2020/FSE/DotNet/02%20-%20NUnit,%20C%23%204.5,%20ASP.Net%20Core/Handson/CalcLibrary.zip?csf=1&e=aLxB66). This will be used to write Unit test cases  
  
Follow the steps listed below to write the NUnit test cases for the application.

* Create a Unit test project(.Net Framework) in the solution provided.
* Add the CalcLibrary project as reference
* Create a class “CalculatorTests” to write all the test cases for the methods in the solution
* Use the ‘TestFixture’, ‘SetUp’ and ‘TearDown’ attributes, to declare, initialize and cleanup activities respectively
* Create a Test method to check the addition functionality
* Use the ‘TestCase’ attribute to send the inputs and the expected result
* Use Assert.That to check the actual and expected result match

**Output:**

****

**Test Class**

**CalculatorTests.cs:**

using NUnit.Framework;

using CalcLibrary;

using System;

namespace CalcLibrary.Tests

{

[TestFixture]

public class CalculatorTests

{

private SimpleCalculator calc;

[SetUp]

public void Setup()

{

calc = new SimpleCalculator();

}

[TearDown]

public void Cleanup()

{

calc = null;

}

[Test]

[TestCase(2, 3, 5)]

[TestCase(-1, -1, -2)]

[TestCase(0, 0, 0)]

public void Addition\_ReturnsCorrectSum(double a, double b, double expected)

{

var result = calc.Addition(a, b);

Assert.That(result, Is.EqualTo(expected));

}

[Test]

[TestCase(5, 3, 2)]

[TestCase(0, 0, 0)]

public void Subtraction\_ReturnsCorrectResult(double a, double b, double expected)

{

var result = calc.Subtraction(a, b);

Assert.That(result, Is.EqualTo(expected));

}

[Test]

[TestCase(2, 3, 6)]

[TestCase(-2, 3, -6)]

public void Multiplication\_ReturnsCorrectProduct(double a, double b, double expected)

{

var result = calc.Multiplication(a, b);

Assert.That(result, Is.EqualTo(expected));

}

[Test]

[TestCase(10, 2, 5)]

public void Division\_ReturnsCorrectQuotient(double a, double b, double expected)

{

var result = calc.Division(a, b);

Assert.That(result, Is.EqualTo(expected));

}

[Test]

public void Division\_ByZero\_ThrowsArgumentException()

{

var ex = Assert.Throws<ArgumentException>(() => calc.Division(5, 0));

Assert.That(ex.Message, Is.EqualTo("Second Parameter Can't be Zero"));

}

}

}