**Week 3**

Unit Testing: 1. NUnit-Handson .

EXERCISE 1

**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

**CalcLibrary project : Calculator.cs**

namespace CalcLibrary

{

public class Calculator

{

public int Add(int a, int b)

{

return a + b;

}

public int Subtract(int a, int b)

{

return a - b;

}

public int Multiply(int a, int b)

{

return a \* b;

}

public double Divide(int a, int b)

{

if (b == 0)

throw new DivideByZeroException("Division by zero is not allowed.");

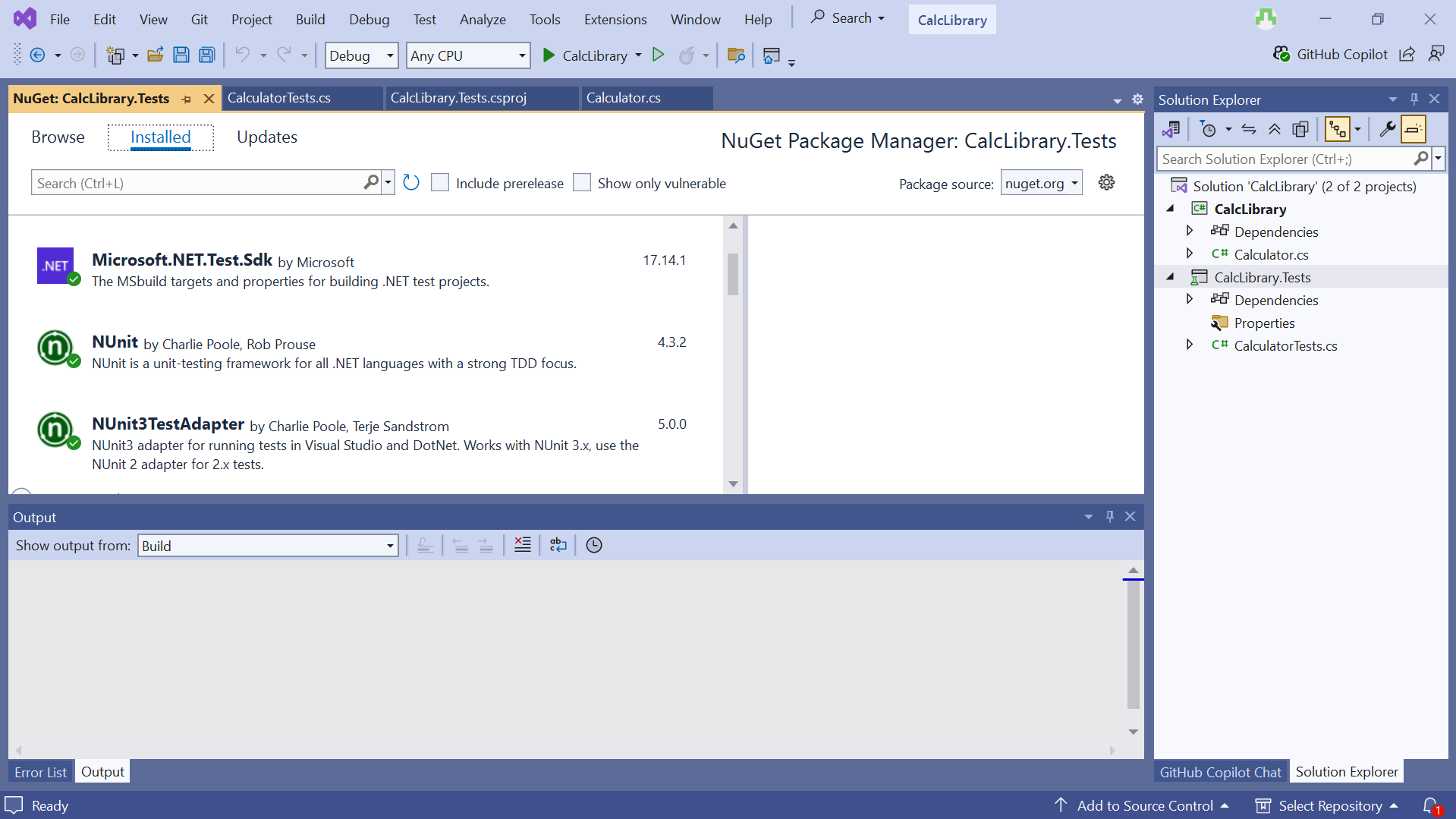
return (double)a / b;

}

}

}

Created a Unit testing project Inside the solution and added reference to our application . Created a Test class **CalculatorTest.cs** and install the packages **Nunit, NUnit3TestAdapter and Microsoft.NET.Test.Sdk**



Code Implementation:

using NUnit.Framework;

using CalcLibrary;

namespace CalcLibrary.Tests

{

[TestFixture]

public class CalculatorTests

{

private Calculator calculator;

[SetUp]

public void Init()

{

calculator = new Calculator();

}

[TearDown]

public void Cleanup()

{

// Optional cleanup

}

[Test]

[TestCase(2, 3, 5)]

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

[TestCase(100, 200, 300)]

public void Add\_WhenCalled\_ReturnsCorrectSum(int a, int b, int expected)

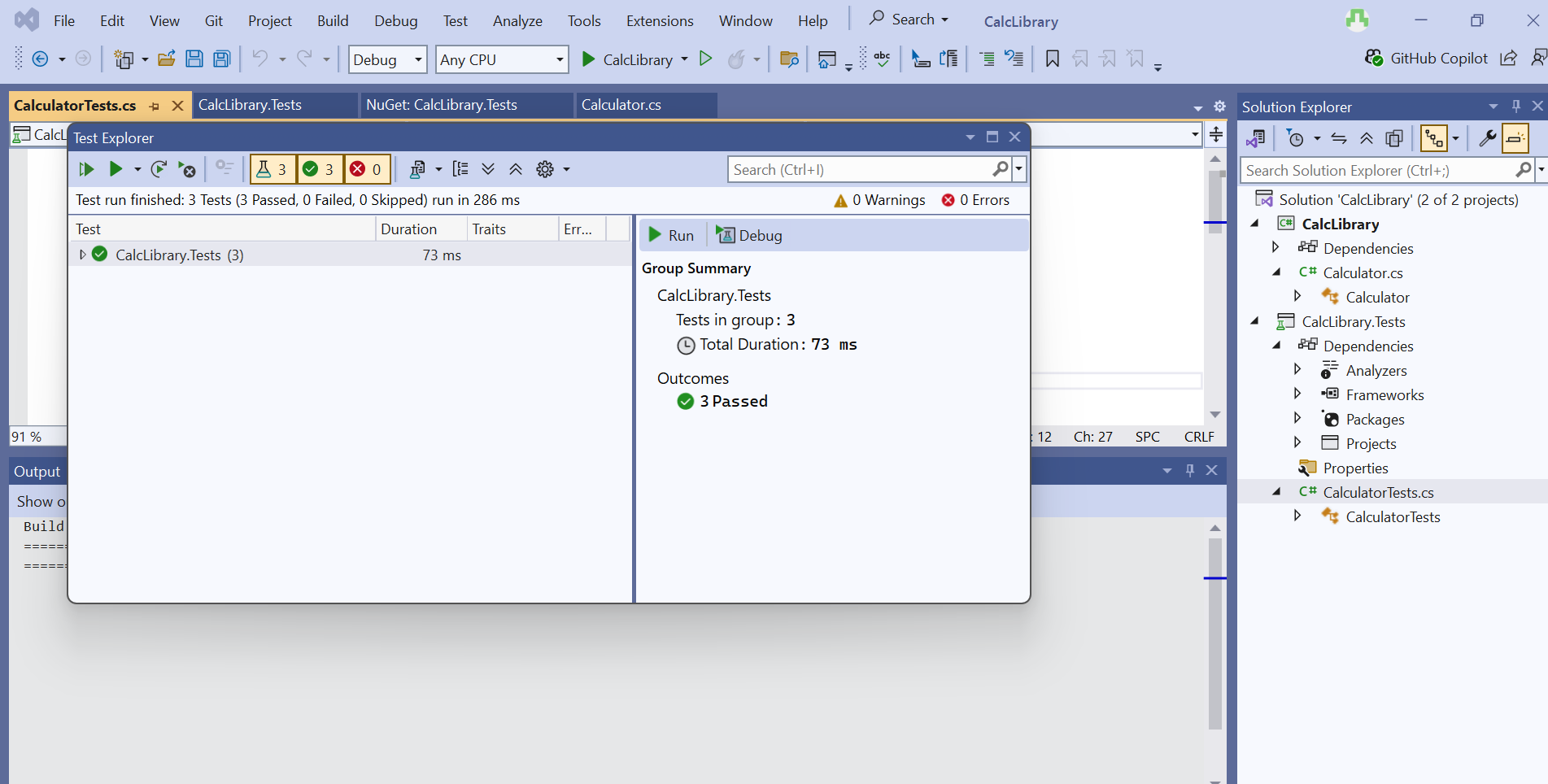
{

var result = calculator.Add(a, b);

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

}

} }



All the three test cases passed.