**INTCDE21ID008**

**STAGE-3**

**916398 – SAAHIL TOMAR**

**Day 1 – NUnit**

**Hands-On 1:**

Follow the steps listed below to write the NUnit test cases for the application.

1. Create a Unit test project(.NET Framework) in the solution provided.

2) Add the CalcLibrary project as reference

3) Create a class “CalculatorTests” to write all the test cases for the methods in the solution

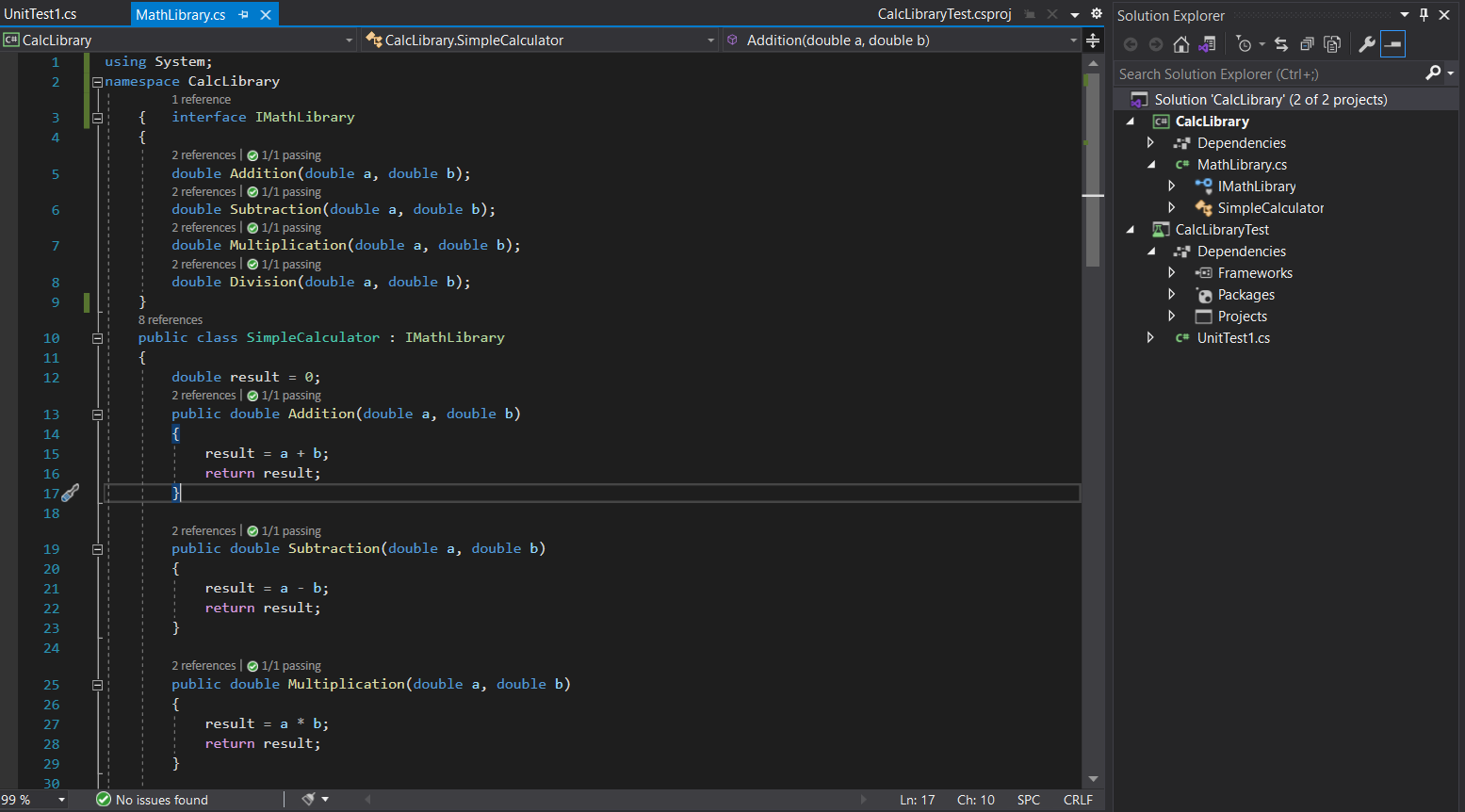
4) Use the ‘TestFixture’, ‘SetUp’ and ‘TearDown’ attributes, to declare, initialize and cleanup activities respectively

5) Create a Test method to check the addition functionality

6) Use the ‘TestCase’ attribute to send the inputs and the expected result

7) Use Assert.That to check the actual and expected result match

**PROJECT CODE:**



**IMPLEMENTATION:**

using NUnit.Framework;

using CalcLibrary;

using System;

namespace CalcLibraryTest

{

[TestFixture]

public class CalculatorTests

{

SimpleCalculator SimpleCalculator;

double Result;

[SetUp]

public void SetUp()

{

SimpleCalculator = new SimpleCalculator();

Result = 0;

}

[TearDown]

public void TearDown()

{

SimpleCalculator = null;

}

[Test]

public void Addition\_result()

{

double expectedResult = 4;

Result = SimpleCalculator.Addition(2, 2);

Assert.AreEqual(expectedResult, Result);

}

[Test]

public void Subtraction\_result()

{

double expectedResult = 5;

Result = SimpleCalculator.Subtraction(10, 5);

Assert.AreEqual(expectedResult, Result);

}

[Test]

public void Multiplication\_result()

{

double expectedResult = 16;

Result = SimpleCalculator.Multiplication(8, 2);

Assert.AreEqual(expectedResult, Result);

}

[Test]

public void Division\_result()

{

double expectedResult = 10;

double Result = SimpleCalculator.Division(20, 2);

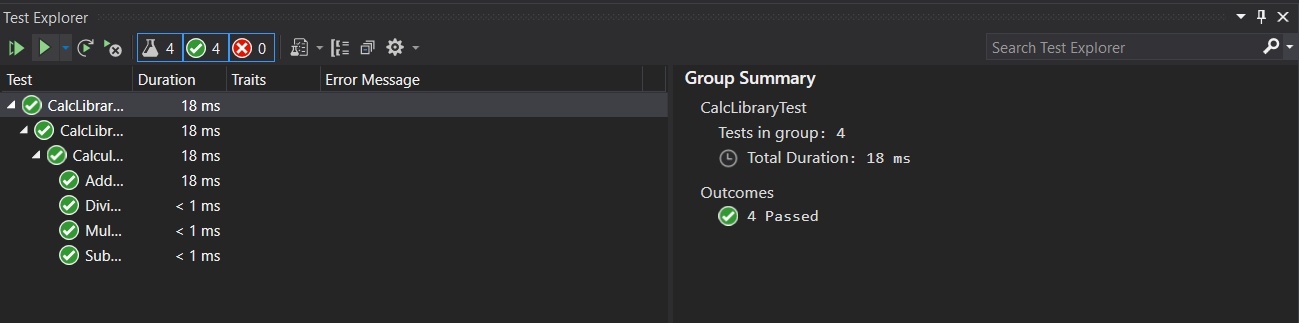
Assert.AreEqual(expectedResult, Result);

}

}

}

**OUTPUT:**



**Hands-On 2:**

Follow the steps listed below to write the NUnit test cases for the application.

1) Create test case to verify the subtraction feature of the calculator with various input types.

2) Create test cases with ‘TestCase’ attribute to send in input parameters and the expected result.

3) Add more than 1 ‘TestCase’ attributes to check various combinations for subtractions.

4) Use Assert.Equal to check the actual and expected results

5) Create a test case to verify the multiplication concepts of calculator

6) Create test cases with ‘TestCase’ attribute to send in input parameters and the expected result.

7) Add more than 1 ‘TestCase’ attributes to check various combinations for subtractions.

8) Use Assert.Equal to check the actual and expected results

9) Create a test case to verify the division logic of the calculator

10) Create test cases with ‘TestCase’ attribute to send in input parameters and the expected result.

11) Add more than 1 ‘TestCase’ attributes to check various combinations for subtractions.

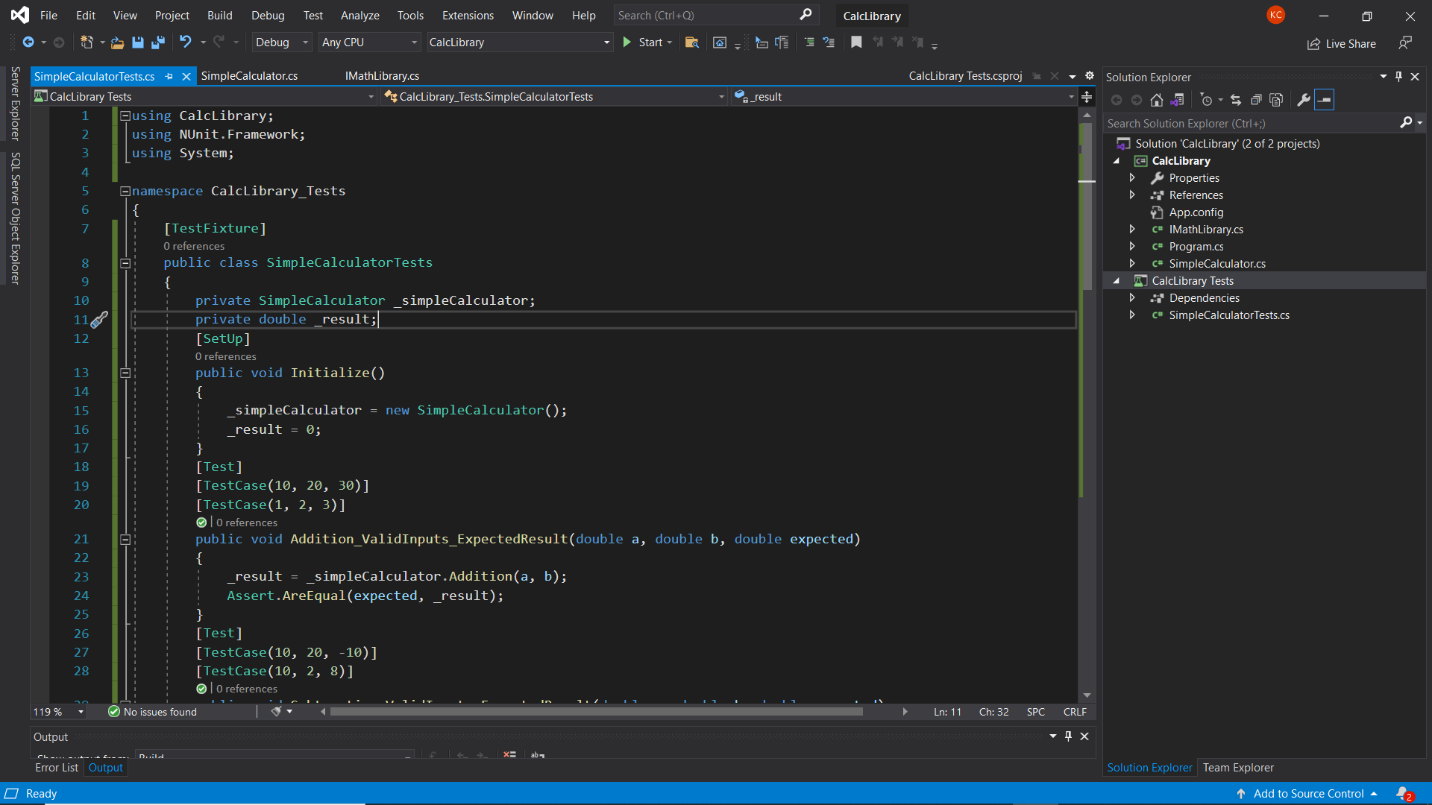
12) Use Assert.Equal to check the actual and expected results

13) In one of the inputs, provide the divisor value to be 0

14) Use Try Catch block to catch the ArgumentException

15) Use Assert.Fail to notify the user that the test case has failed. Give the message “Division by zero” in the Assert.Fail, which will be notified to the user. This message will be seen in the test explorer.

**PROJECT CODE:**



**IMPLEMENTATION:**

using System;

using CalcLibrary;

using NUnit.Framework;

namespace CalCtest

{

[TestFixture]

class Handson2

{

SimpleCalculator p;

private double result;

[SetUp]

public void SetUp()

{

p = new SimpleCalculator();

}

[TearDown]

public void TearDown()

{

p = null;

}

[TestCase]

public void SubstractionTestcase1()

{

double ares = p.Subtraction(30, 20);

double eres = 10;

Assert.That(ares, Is.EqualTo(eres));

}

[TestCase]

public void SubstractionTestcase2()

{

double ares = p.Subtraction(20, 30);

double eres = -10;

Assert.That(ares, Is.EqualTo(eres));

}

[TestCase]

public void MultiplicationTestCase1()

{

double ares = p.Multiplication(5, 4);

double eres = 20;

Assert.That(ares, Is.EqualTo(eres));

}

[TestCase]

public void MultiplicationTestCase2()

{

double ares = p.Multiplication(4, 4);

double eres = 16;

Assert.That(ares, Is.EqualTo(eres));

}

[TestCase]

public void DivisionTestcase1()

{

double ares = p.Division(10, 10);

Assert.That(1, Is.EqualTo(ares));

}

[TestCase]

public void DivisionTestcase2()

{

try

{

double f = p.Division(10, 0);

// Assert.Fail("xyz",);

}

catch (Exception v)

{

Assert.AreEqual("Second Parameter Can't be Zero", v.Message);

}

}

[TearDown]

public void CleanUp()

{

p.AllClear();

result = p.GetResult;

Console.WriteLine(result);

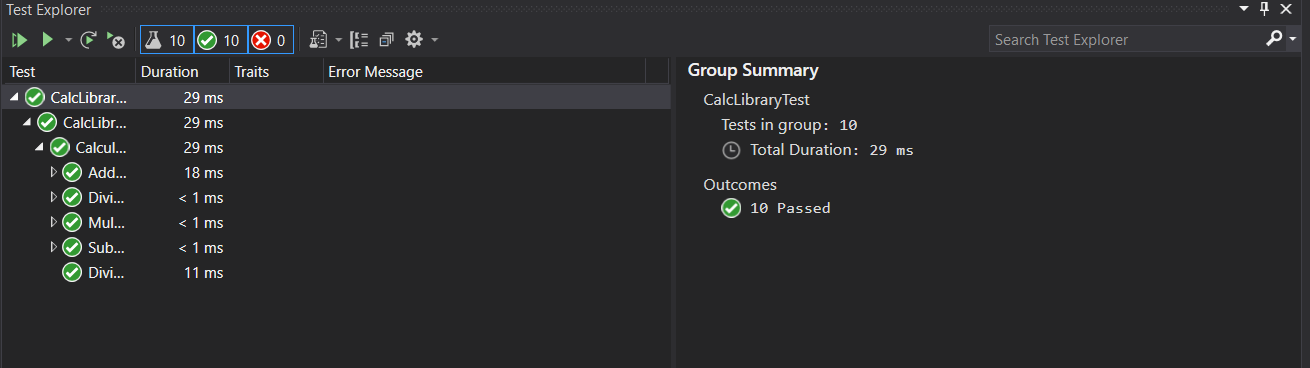
Assert.AreEqual(0, result);

}

}

}

**OUTPUT:**



**Hands-On 3:**

Follow the steps listed below to write the NUnit test cases for the application.

1) Create a Class Library project in the same solution which is provided and name it as suggested.

2) Rename the class file name (<SUT>Tests.cs).

3) Add the assembly reference of the UtilLib project to the test project.

4) Additionally, add the reference of both NUnit and NUnit3TestAdapter in the test project using NuGet Package Manager (NPM).

5) Write the suggested test methods.

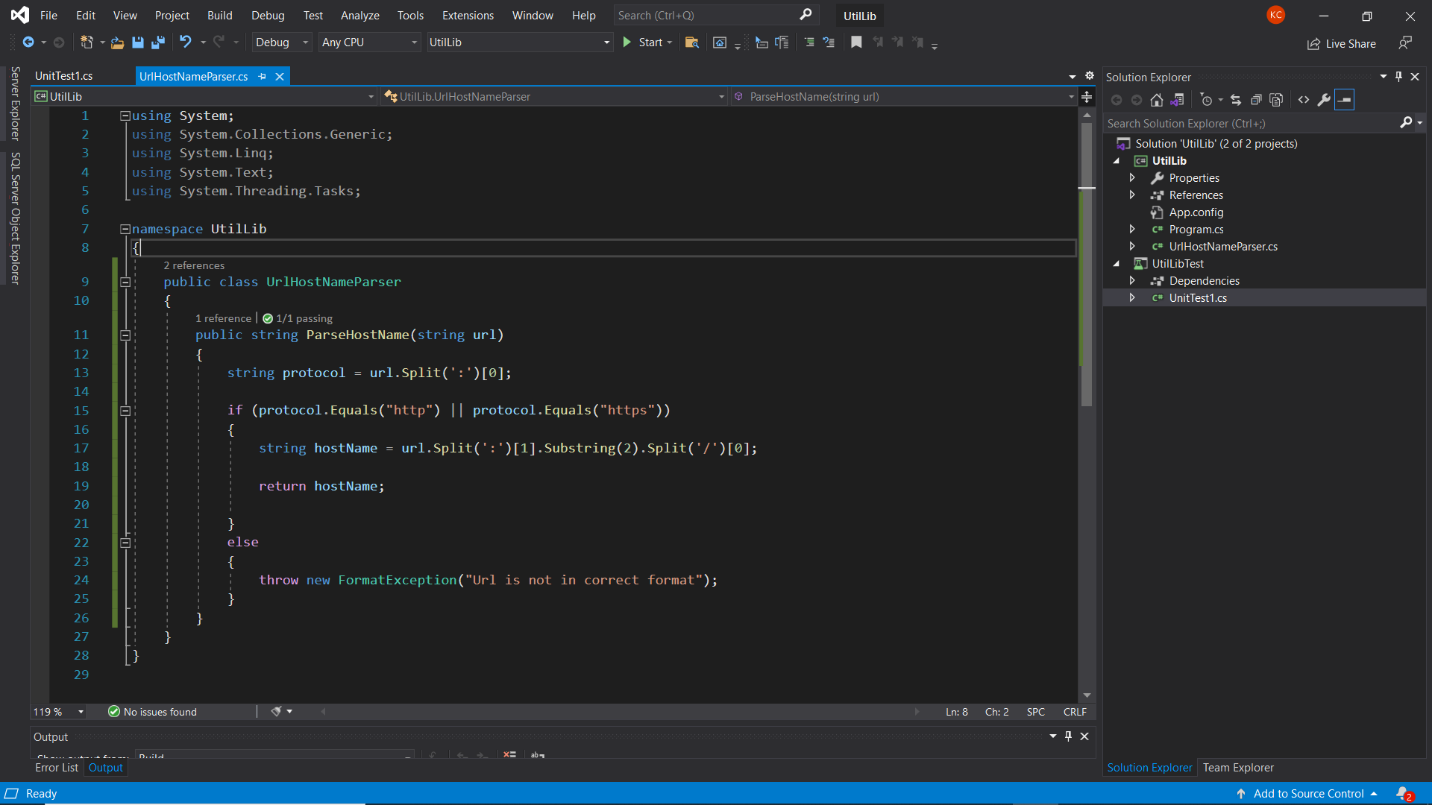
6) Run your tests.

7) Break the test by modifying the source project functionality.

8) Rerun the test.

9) Observe the test result.

**PROJECT CODE:**



**IMPLEMENTATION:**

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using NUnit.Framework;

using nunithandson3;

using Assert = NUnit.Framework.Assert;

namespace UnitTestProject3

{

[TestFixture]

public class UnitTest1

{

UrlHostNameParser u;

[SetUp]

public void SetUp()

{

u = new UrlHostNameParser();

}

[TearDown]

public void Dispose()

{

u = null;

}

[TestCase]

public void Testcase1()

{

string act = u.ParseHostName("https://Facebook.com");

string exp = "Facebook.com";

Assert.That(act, Is.EqualTo(exp));

}

[TestCase]

public void Testcase2()

{

string act = u.ParseHostName("http://twitter.com");

string exp = "twitter.com";

Assert.That(act, Is.EqualTo(exp));

}

[TestCase]

public void Testcase3()

{

var ex = Assert.Throws<FormatException>(() => u.ParseHostName("https12://gmail.com"));

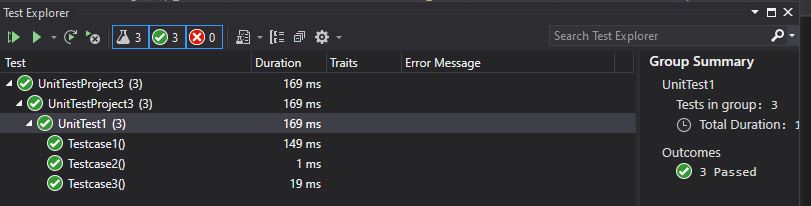
Assert.That(ex.Message, Is.EqualTo("Url is not in correct format"));

}

}

}

**OUTPUT:**



**Hands-On 4:**

Follow the steps listed below to write the NUnit test cases for the application.

1) Create a Class Library project in the same solution which is provided and name it as suggested.

2) Rename the class file name (<SUT>Tests.cs).

3) Add the assembly reference of the UtilLib project to the test project.

4) Additionally add the reference of both NUnit and NUnit3TestAdapter in the test project using NuGet Package Manager (NPM).

5) Write the suggested test methods.

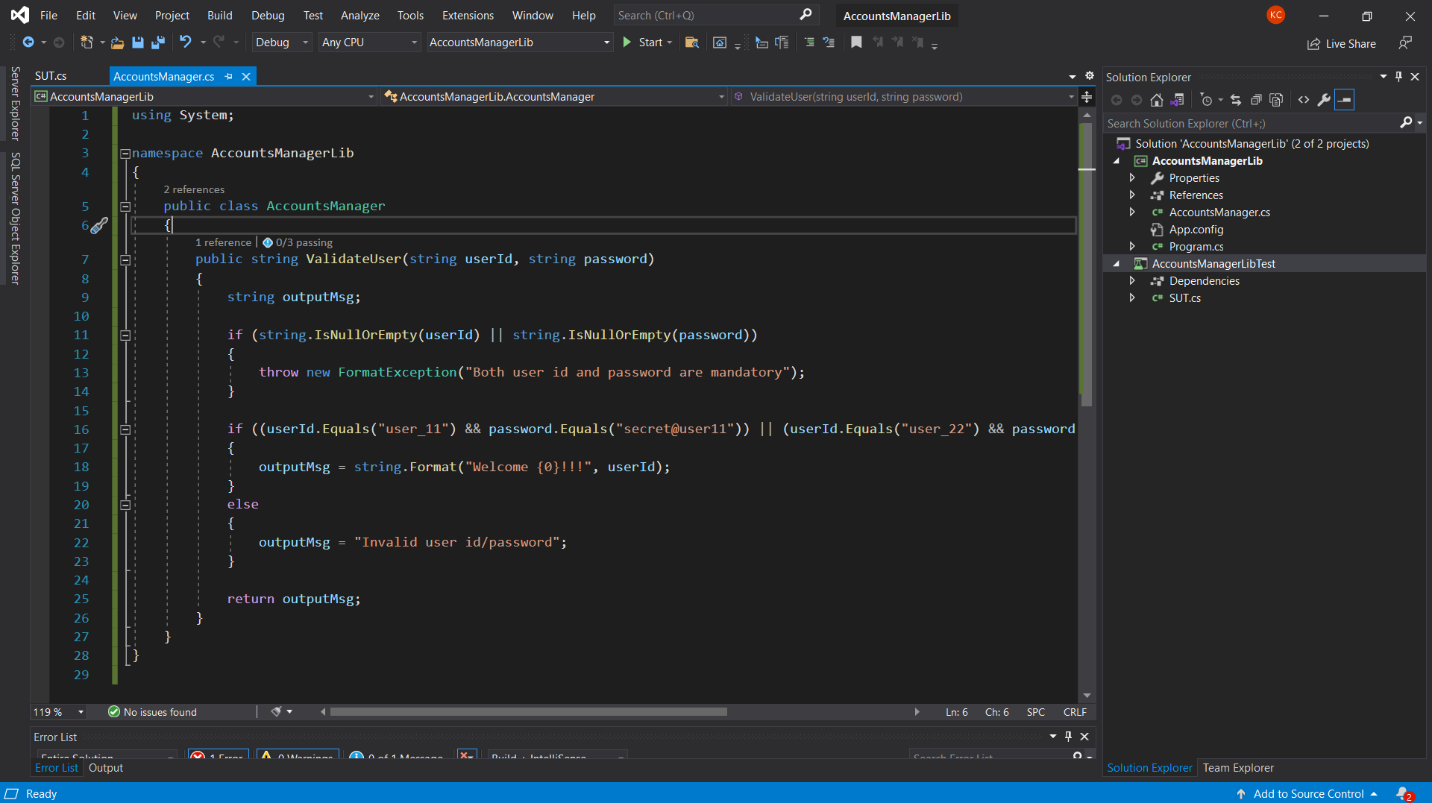
6) Run your tests.

7) Break the test by modifying the source project functionality.

8) Rerun the test.

9) Observe the test result.

**PROJECT CODE:**



**IMPLEMENTATION:**

using AccountsManagerLib;

using NUnit.Framework;

using System;

namespace Handson\_3

{

[TestFixture]

public class UnitTest1

{

AccountsManager u;

[SetUp]

public void SetUp()

{

u = new AccountsManager();

}

[TearDown]

public void Dispose()

{

u = null;

}

[TestCase]

public void LoginTestcase1()

{

string exp = "Invalid user id/password";

string act = u.ValidateUser("user\_22", "secret@user12s");

Assert.That(act, Is.EqualTo(exp));

}

[TestCase]

public void LoginTestcase2()

{

try

{

String f = u.ValidateUser("user2", "");

// Assert.Fail("xyz",);

}

catch (Exception v)

{

Assert.AreEqual("Both user id and password are mandatory", v.Message);

}

}

[TestCase]

public void LoginTestcase3()

{

string act = u.ValidateUser("user\_11", "secret@user11");

string exp = string.Format("Welcome user\_11!!!");

Assert.That(act, Is.EqualTo(exp));

}

[TestCase]

public void LoginTestcase4()

{

string exp = "Invalid user id/password";

string act = u.ValidateUser("user\_23", "secret@user12s");

Assert.That(act, Is.EqualTo(exp));

}

}

}

**OUTPUT:**

