**Hands on Day -1**

**INTCDE21ID008**

**C N S Varun**

**916214**

**Hands On-1**

**Unit Testing Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using NUnit.Framework;

using CalcLibrary;

namespace CalCtest

{

[TestFixture]

class UnitTest1

{

SimpleCalculator s;

[SetUp]

public void Setup()

{

s = new SimpleCalculator();

}

[TearDown]

public void TearDown()

{

s = null;

}

[TestCase]

public void add()

{

double sam=s.Addition(2, 10);

Assert.AreEqual(12, sam);

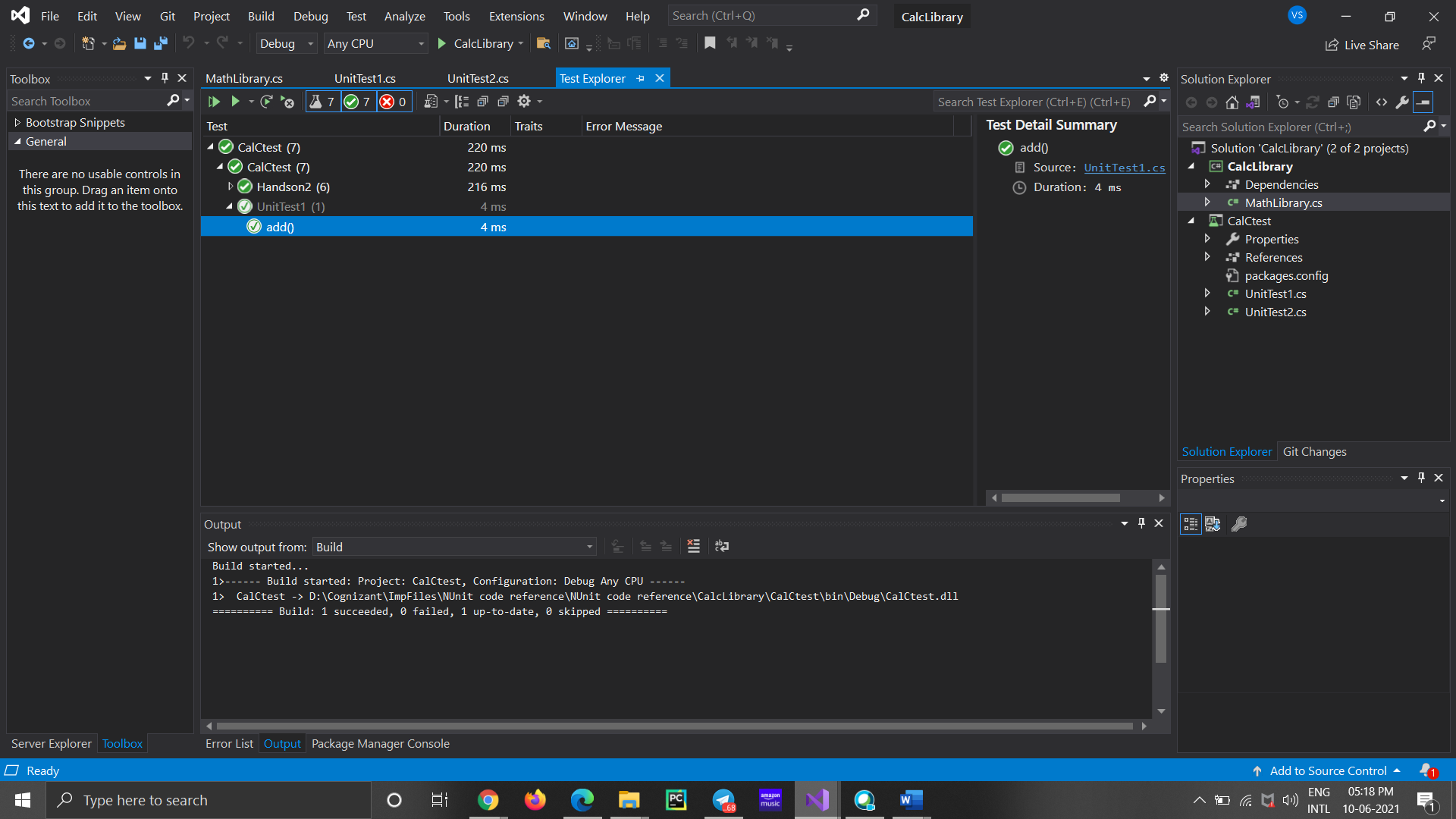
Assert.AreNotEqual(10, sam);

}

}

}

**Test Explorer**



**Handson-2**

**Unit Testing Code:**

//using Microsoft.VisualStudio.TestTools.UnitTesting;

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);

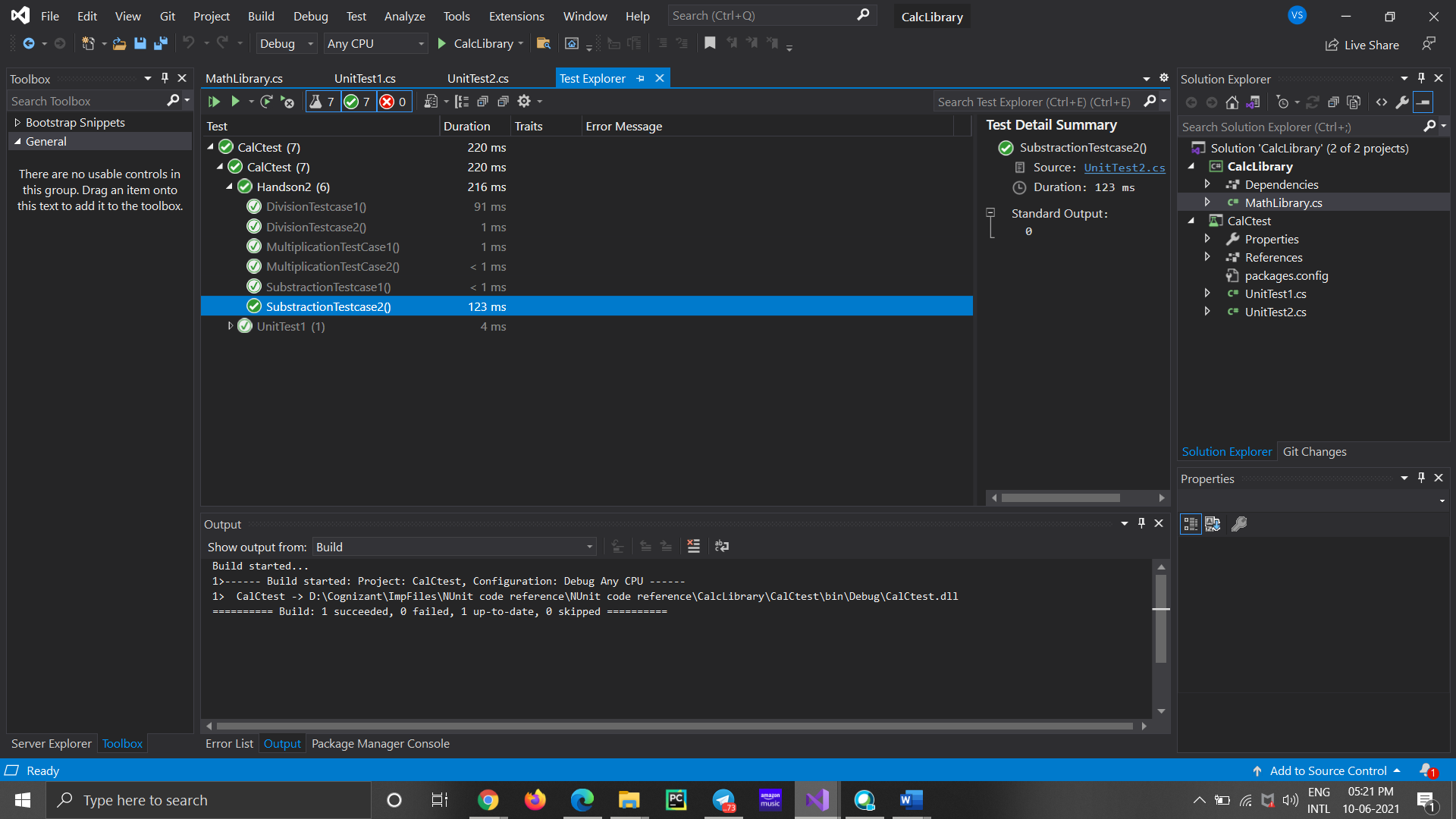
Assert.AreEqual(0, result);

}

}

}

**Test Explorer**



**Handson-3**

**Unit Testing Code:**

using NUnit.Framework;

using UtilLib;

using System;

namespace Handson

{

[TestFixture]

public class UnitTest1

{

UrlHostNameParser d;

[SetUp]

public void Setup()

{

d = new UrlHostNameParser();

}

[TearDown]

public void TearDown()

{

d = null;

}

[TestCase]

public void TestcaseCrct()

{

string a = d.ParseHostName("https://instagram.com");

Console.WriteLine(a);

Assert.That("instagram.com", Is.EqualTo(a));

}

[TestCase]

public void TestcaseFalse()

{

try

{

string a = d.ParseHostName("https12://instagram.com");

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

}

catch (Exception v)

{

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

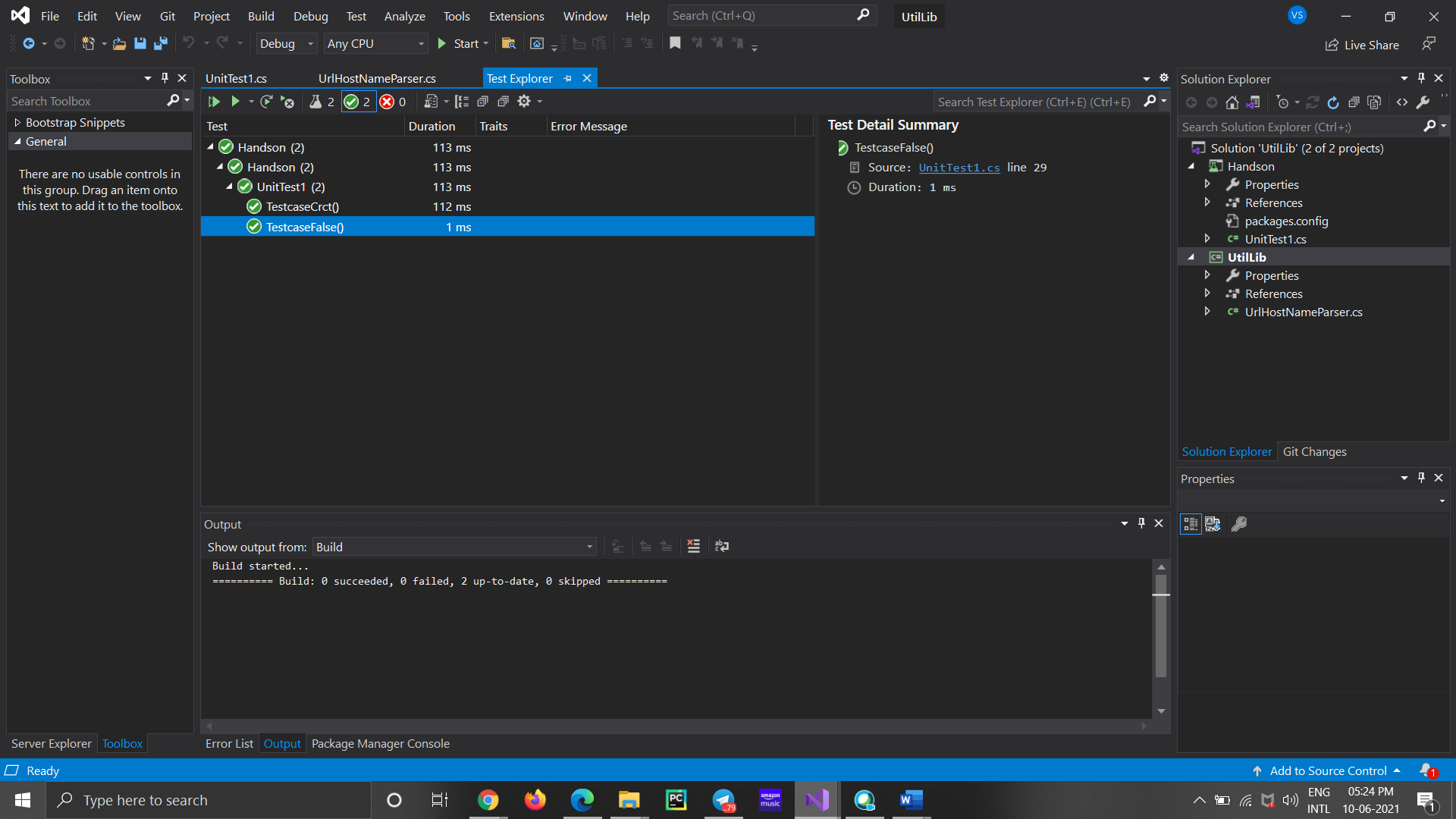
}

}

}

}

**Test Explorer:**



**Handson-4**

**Unit Testing Code**

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));

}

}

}

**Test Explorer**

