**NUnit and Moq**

Suman Sourabh

6363866

Question -1:

**NUnit HandsOn:**

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

CODE:

CalculatorTests.cs

using NUnit.Framework;

using CalcLibrary;

namespace CalcLibrary.Tests

{

[TestFixture]

public class CalculatorTests

{

private Calculator \_calculator;

[SetUp]

public void SetUp()

{

\_calculator = new Calculator();

}

[TearDown]

public void TearDown()

{

\_calculator = null;

}

[Test]

[TestCase(2, 3, 5)]

[TestCase(0, 0, 0)]

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

public void Add\_AddsTwoNumbers\_ReturnsExpectedResult(int a, int b, int expected)

{

var result = \_calculator.Add(a, b);

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

}

[Test]

[Ignore("Demo: Ignored test")]

public void IgnoredTest()

{

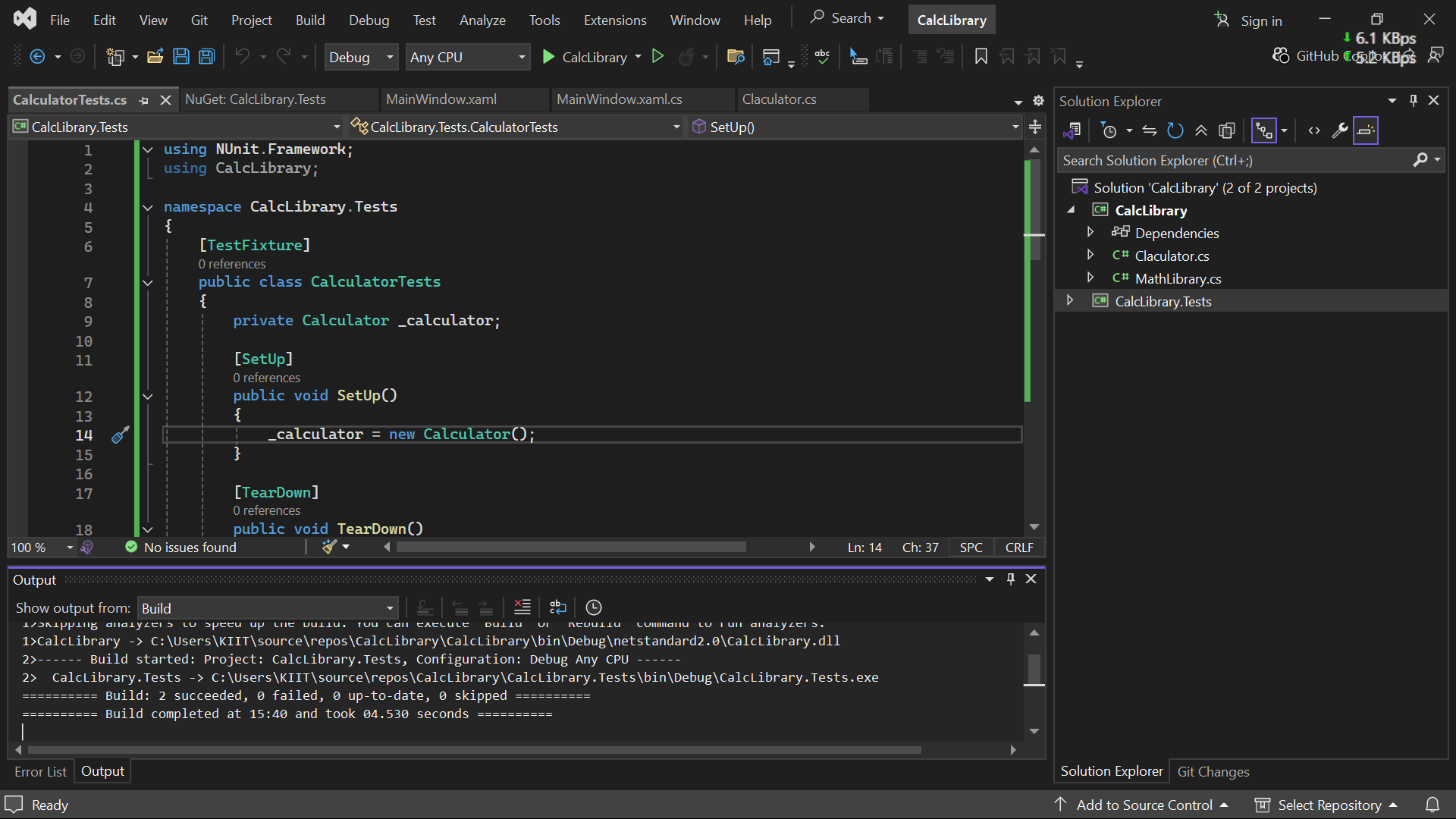
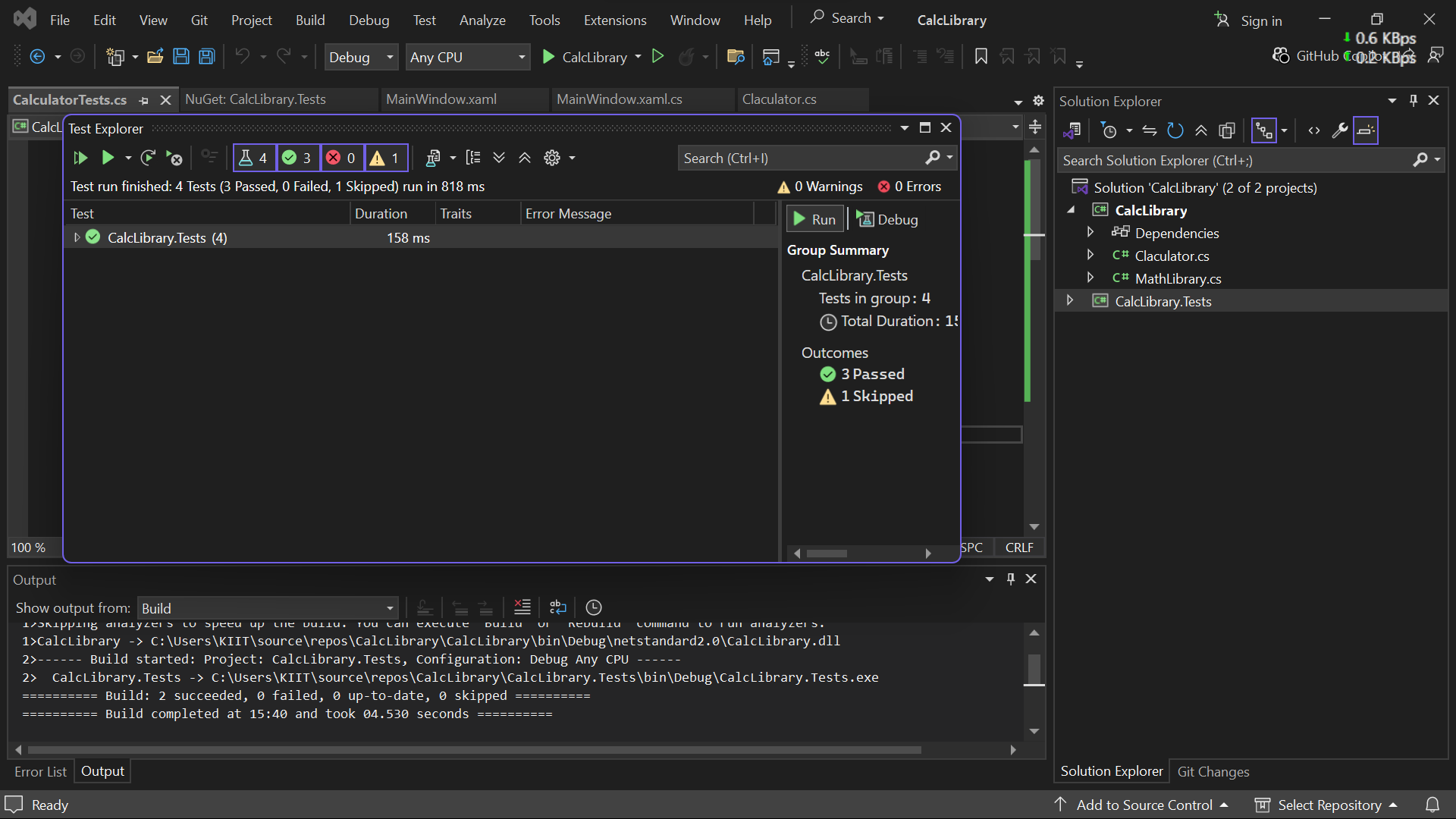
Assert.Fail("This should not run.");

}

}

}

OUTPUT:



Question-2:

1. **Write Testable Code with Moq**

**CODE:**

**CustomerCommTests.cs:**

using NUnit.Framework;

using Moq;

using CustomerCommLib;

namespace CustomerComm.Tests

{

[TestFixture]

public class CustomerCommTests

{

private Mock<IMailSender> \_mockMailSender;

private CustomerCommLib.CustomerComm \_customerComm;

[OneTimeSetUp]

public void Init()

{

\_mockMailSender = new Mock<IMailSender>();

\_mockMailSender

.Setup(m => m.SendMail(It.IsAny<string>(), It.IsAny<string>()))

.Returns(true);

\_customerComm = new CustomerCommLib.CustomerComm(\_mockMailSender.Object);

}

[Test]

public void SendMailToCustomer\_ReturnsTrue()

{

var result = \_customerComm.SendMailToCustomer();

Assert.IsTrue(result);

}

}

}

**CustomerComm.cs:**

namespace CustomerCommLib

{

public class CustomerComm

{

IMailSender \_mailSender;

public CustomerComm(IMailSender mailSender)

{

\_mailSender = mailSender;

}

public bool SendMailToCustomer()

{

\_mailSender.SendMail("cust123@abc.com", "Some Message");

return true;

}

}

}

**MailSender.cs:**

using System.Net;

using System.Net.Mail;

namespace CustomerCommLib

{

public interface IMailSender

{

bool SendMail(string toAddress, string message);

}

public class MailSender : IMailSender

{

public bool SendMail(string toAddress, string message)

{

MailMessage mail = new MailMessage();

SmtpClient SmtpServer = new SmtpClient("smtp.gmail.com");

mail.From = new MailAddress("your\_email\_address@gmail.com");

mail.To.Add(toAddress);

mail.Subject = "Test Mail";

mail.Body = message;

SmtpServer.Port = 587;

SmtpServer.Credentials = new NetworkCredential("username", "password");

SmtpServer.EnableSsl = true;

SmtpServer.Send(mail);

return true;

}

}

}

**OUTPUT:**

