**NUnit and MOQ**

**NUnit-HandsOn-Exercise-1**

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:

//File:CalcLibraryTests.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using NUnit.Framework;

using CalcLibrary;

namespace CalcLibraryTests

{

[TestFixture]

public class CalculatorTests

{

private SimpleCalculator \_calculator;

[SetUp]

public void Init()

{

\_calculator = new SimpleCalculator();

}

[TearDown]

public void Cleanup()

{

\_calculator = null;

}

[Test]

public void Addition\_TwoPositiveNumbers\_ReturnsCorrectSum()

{

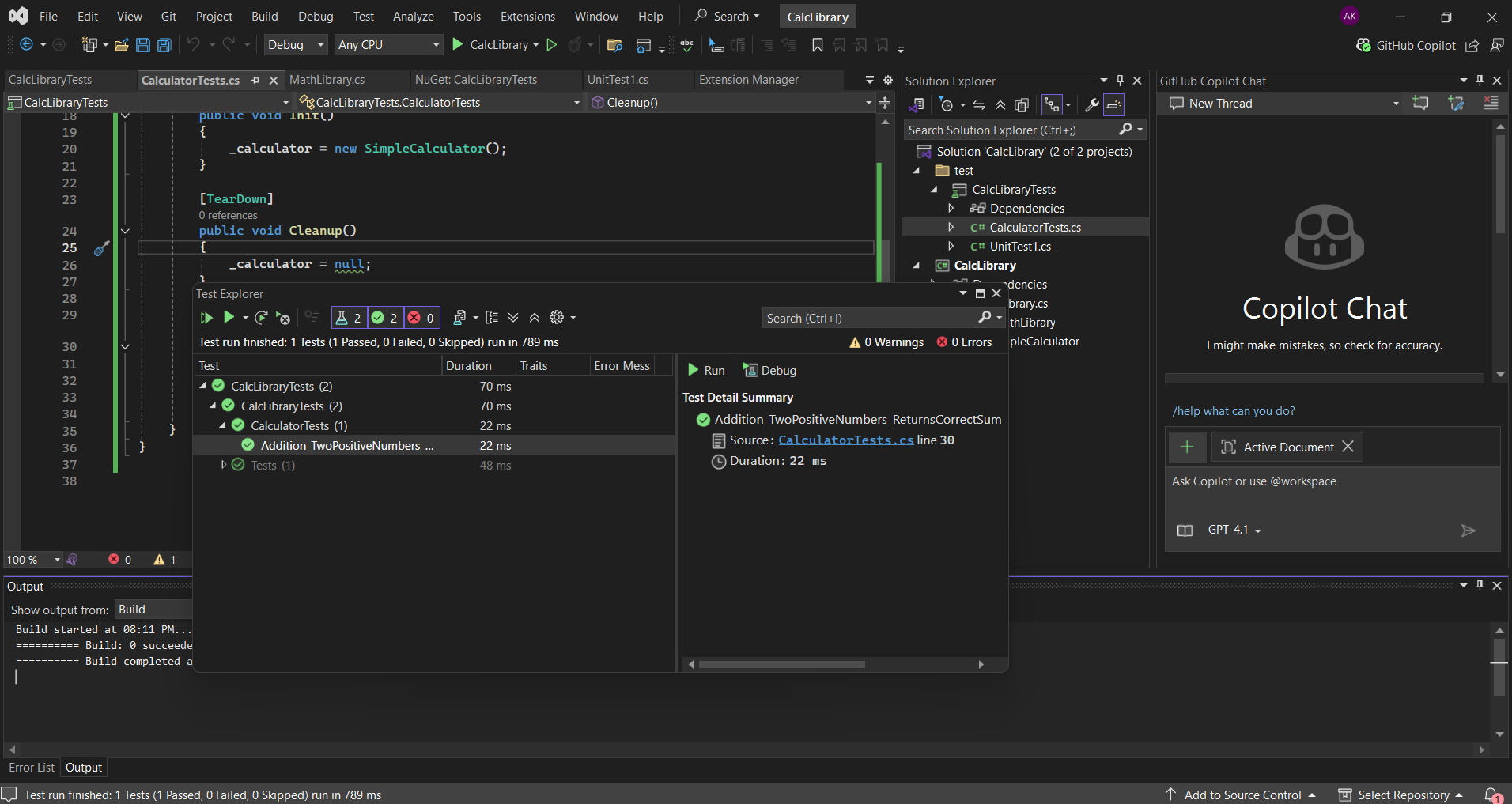
double result = \_calculator.Addition(10, 20);

Assert.That(result, Is.EqualTo(30));

}

}

}

Output:

1. **Moq-Handson**
   1. **Write Testable Code with Moq**

Task-1 You will create a class library that will be used for unit testing.

Code:

//File: CustomerCommLib/CustomerComm.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CustomerCommLib

{

public class CustomerComm

{

IMailSender \_mailSender;

public CustomerComm(IMailSender mailSender)

{

\_mailSender = mailSender;

}

public bool SendMailToCustomer()

{

return \_mailSender.SendMail("kranurag2004@gmail.com", "Hey! Its me!!");

}

}

}

//File:CustomerCommLib/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;

}

}

}

Task-2 Create unit test project which make use of NUnit framework and Moq.

//File:CustomerCommTests/CustomerCommTests.cs

using CustomerCommLib;

using Moq;

using NUnit.Framework;

namespace CustomerCommTests

{

[TestFixture]

public class CustomerCommTests

{

private Mock<IMailSender> mockMailSender = null!;

private CustomerComm customerComm = null!;

[SetUp]

public void Setup()

{

mockMailSender = new Mock<IMailSender>();

customerComm = new CustomerComm(mockMailSender.Object);

}

[Test]

public void SendMailToCustomer\_WhenMailSentSuccessfully\_ShouldReturnTrue()

{

// Arrange

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

.Returns(true);

// Act

bool result = customerComm.SendMailToCustomer();

// Assert

Assert.That(result, Is.True);

mockMailSender.Verify(x => x.SendMail(It.IsAny<string>(), It.IsAny<string>()),

Times.Once);

}

[Test]

public void SendMailToCustomer\_WhenMailSendingFails\_ShouldReturnFalse()

{

// Arrange

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

.Returns(false);

// Act

bool result = customerComm.SendMailToCustomer();

// Assert

Assert.That(result, Is.False);

mockMailSender.Verify(x => x.SendMail(It.IsAny<string>(), It.IsAny<string>()),

Times.Once);

}

[Test]

public void SendMailToCustomer\_ShouldCallSendMailWithCorrectParameters()

{

// Arrange

string expectedEmail = "kranurag2004@gmail.com";

string expectedMessage = "Hey! Its me!!";

mockMailSender.Setup(x => x.SendMail(expectedEmail, expectedMessage))

.Returns(true);

// Act

customerComm.SendMailToCustomer();

// Assert

mockMailSender.Verify(x => x.SendMail(expectedEmail, expectedMessage),

Times.Once);

}

}

}

Output:

