**NUnit Hands on**

**TestFixture & Test**

using NUnit.Framework;

using CalcLibrary;

using System;

namespace CalcLibrary.Tests

{

[TestFixture]

public class CalculatorTests

{

private SimpleCalculator calculator;

[SetUp]

public void SetUp()

{

calculator = new SimpleCalculator();

}

[TearDown]

public void TearDown()

{

calculator = null;

}

[Test]

[TestCase(2, 3, 5)]

[TestCase(-1, -2, -3)]

[TestCase(0, 0, 0)]

[TestCase(10.5, 4.5, 15.0)]

public void Addition\_WhenCalled\_ReturnsExpectedResult(double a, double b, double expected)

{

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

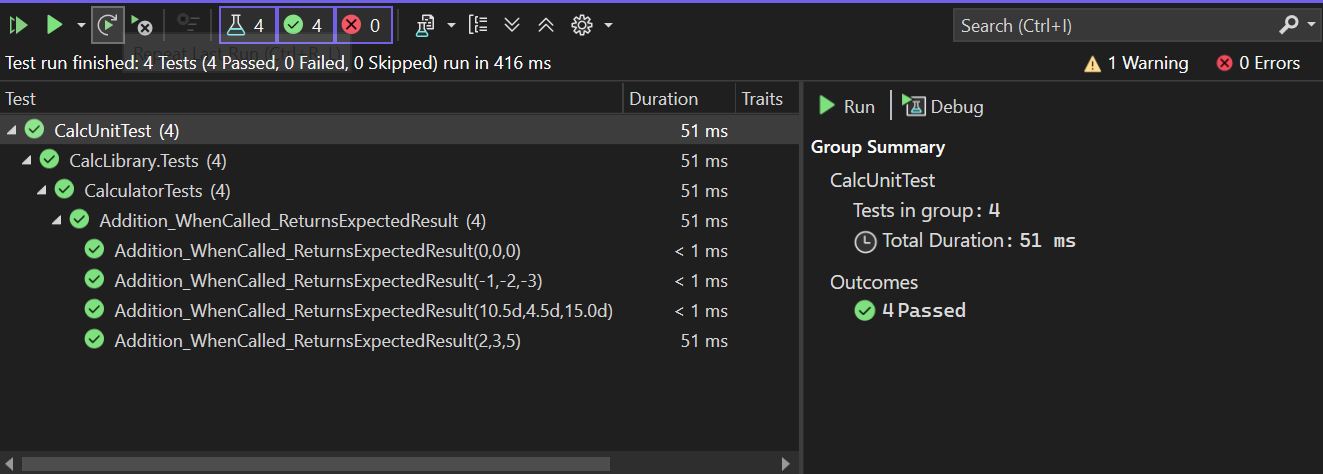
Assert.That(result, Is.EqualTo(expected));

}

}

}

**OUTPUT**



**Moq Handson**

**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;

}

}

}

**CustomeComm.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()

{

//Actual logic goes here

//define message and mail address

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

return true;

}

}

}

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

.Returns(true);

customerComm = new CustomerCommLib.CustomerComm(mockMailSender.Object);

}

[Test]

public void SendMailToCustomer\_WhenCalled\_ReturnsTrue()

{

bool result = customerComm.SendMailToCustomer();

Assert.That(result, Is.True);

}

}

}

**OUTPUT**

