1. **NUnit- Handson**

**Calculator.cs**

namespace CalcLibrary

{

public class Calculator

{

public int Add(int a, int b) => a + b;

}

}

**calculatorTest.cs**

using NUnit.Framework;

using CalcLibrary;

namespace CalcLibrary.Tests

{

[TestFixture]

public class CalculatorTests

{

private Calculator \_calculator;

[SetUp]

public void Setup()

{

\_calculator = new Calculator();

}

[Test]

[TestCase(1, 2, 3)]

[TestCase(-1, 1, 0)]

[TestCase(0, 0, 0)]

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

{

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

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

}

[TearDown]

public void Teardown()

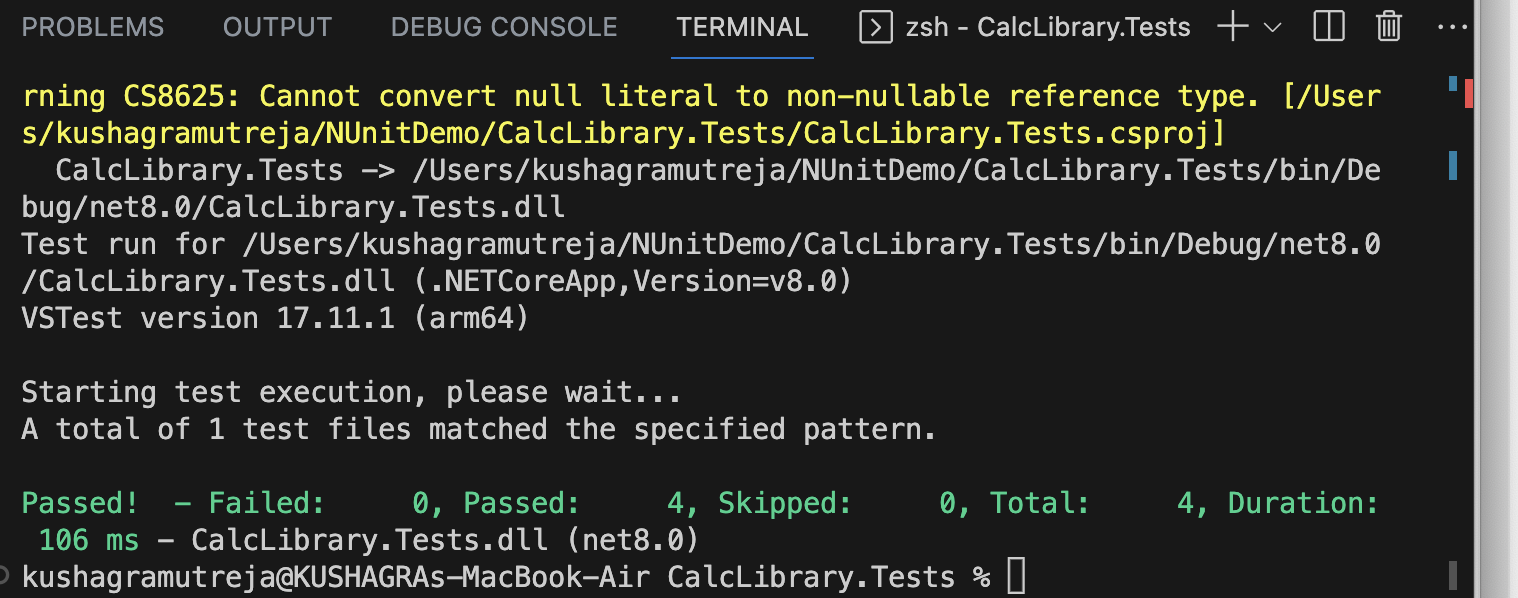
{

\_calculator = null;

}

}

}

****

**2 MoQ Handson**

namespace CustomerCommLib

{

public class CustomerComm

{

private IMailSender \_mailSender;

public CustomerComm(IMailSender mailSender)

{

\_mailSender = mailSender;

}

public bool SendMailToCustomer()

{

// Real customer data in real app

string toAddress = "cust123@abc.com";

string message = "Some Message";

return \_mailSender.SendMail(toAddress, message);

}

}

}

namespace CustomerCommLib

{

public interface IMailSender

{

bool SendMail(string toAddress, string message);

}

}

using System.Net;

using System.Net.Mail;

namespace CustomerCommLib

{

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@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;

}

}

}

using NUnit.Framework;

using Moq;

using CustomerCommLib;

namespace CustomerCommLib.Tests

{

[TestFixture]

public class CustomerCommTests

{

private Mock<IMailSender> \_mockMailSender;

private CustomerComm \_customerComm;

[SetUp]

public void Setup()

{

\_mockMailSender = new Mock<IMailSender>();

\_mockMailSender.Setup(m => m.SendMail(It.IsAny<string>(), It.IsAny<string>())).Returns(true);

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

}

[Test]

public void SendMailToCustomer\_WhenCalled\_ReturnsTrue()

{

var result = \_customerComm.SendMailToCustomer();

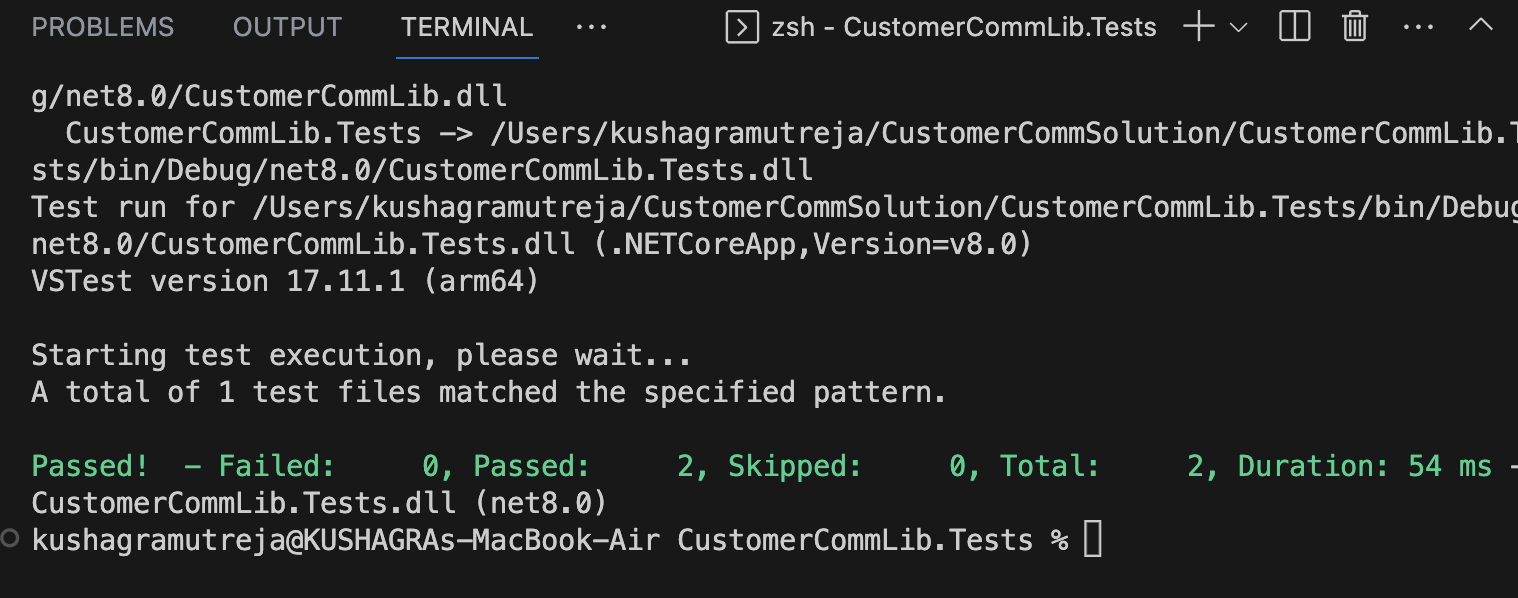
Assert.That(result, Is.True);

\_mockMailSender.Verify(m => m.SendMail("cust123@abc.com", "Some Message"), Times.Once);

}

}

}

****