**SuperSet ID: 6365365**

1. **NUnit-Handson**

Calculator.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CalcLibrary

{

public class Calculator

{

public int Add(int a, int b)

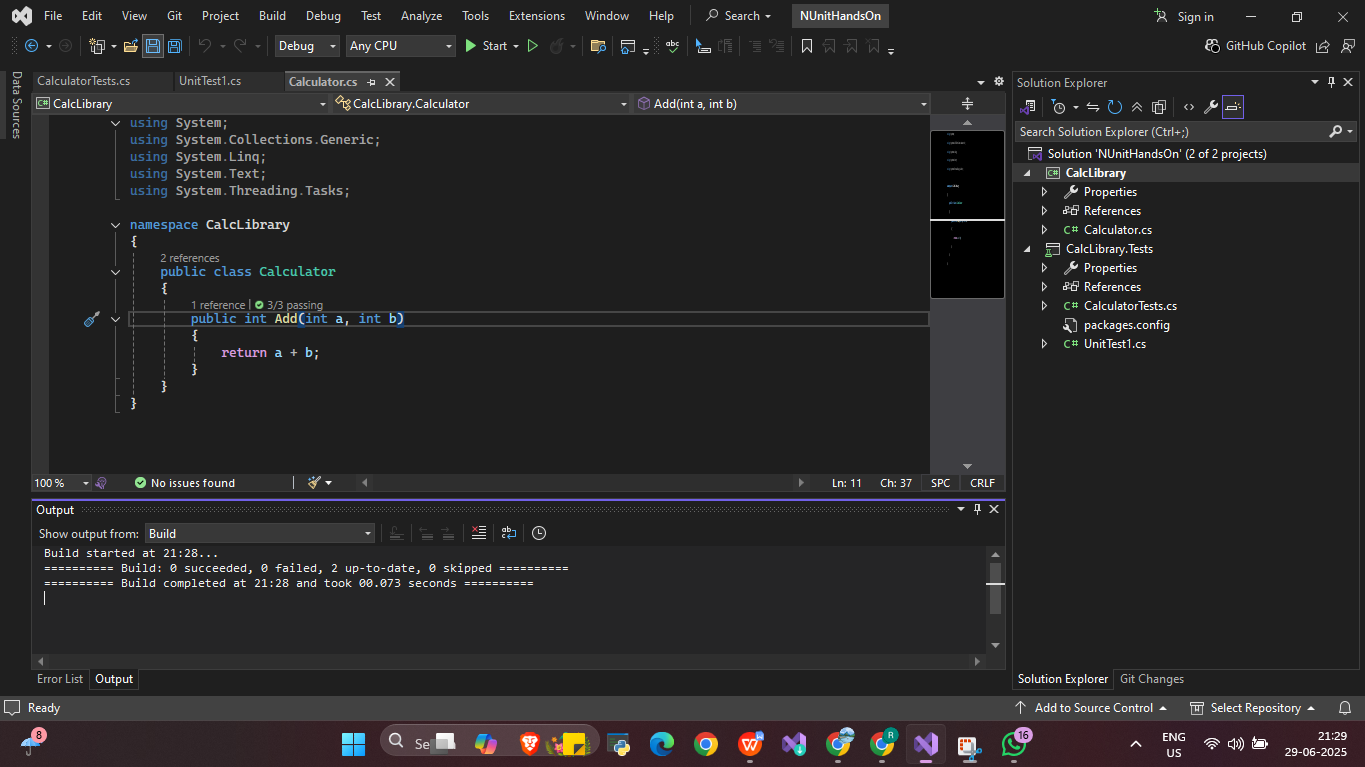
{

return a + b;

}

}

}



CalculatorTests.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using NUnit.Framework;

using CalcLibrary;

namespace CalcLibrary.Tests

{

[TestFixture]

public class CalculatorTests

{

private Calculator calc;

[SetUp]

public void Setup()

{

calc = new Calculator();

}

[TearDown]

public void Teardown()

{

calc = null;

}

[Test]

[TestCase(2, 3, 5)]

[TestCase(0, 0, 0)]

[TestCase(-1, 1, 0)]

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

{

var result = calc.Add(a, b);

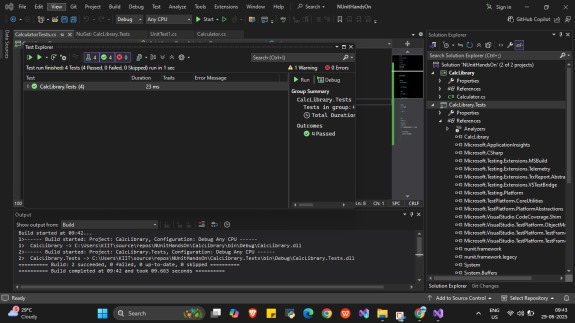
Assert.That(result, Is.EqualTo(expected));

}

}

}

Output



1. **Moq Handson**

**Write Testable Code with Moq**

MailSender.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Net;

using System.Net.Mail;

// MailSender.cs

namespace CustomerCommLib // ✅ Clear, not CustomerComm

{

public interface IMailSender

{

bool SendMail(string toAddress, string message);

}

public class MailSender : IMailSender

{

public bool SendMail(string toAddress, string message)

{

// Just simulate

return true;

}

}

public class CustomerComm // ✅ Class name is CustomerComm

{

private IMailSender \_mailSender;

public CustomerComm(IMailSender mailSender)

{

\_mailSender = mailSender;

}

public bool SendMailToCustomer()

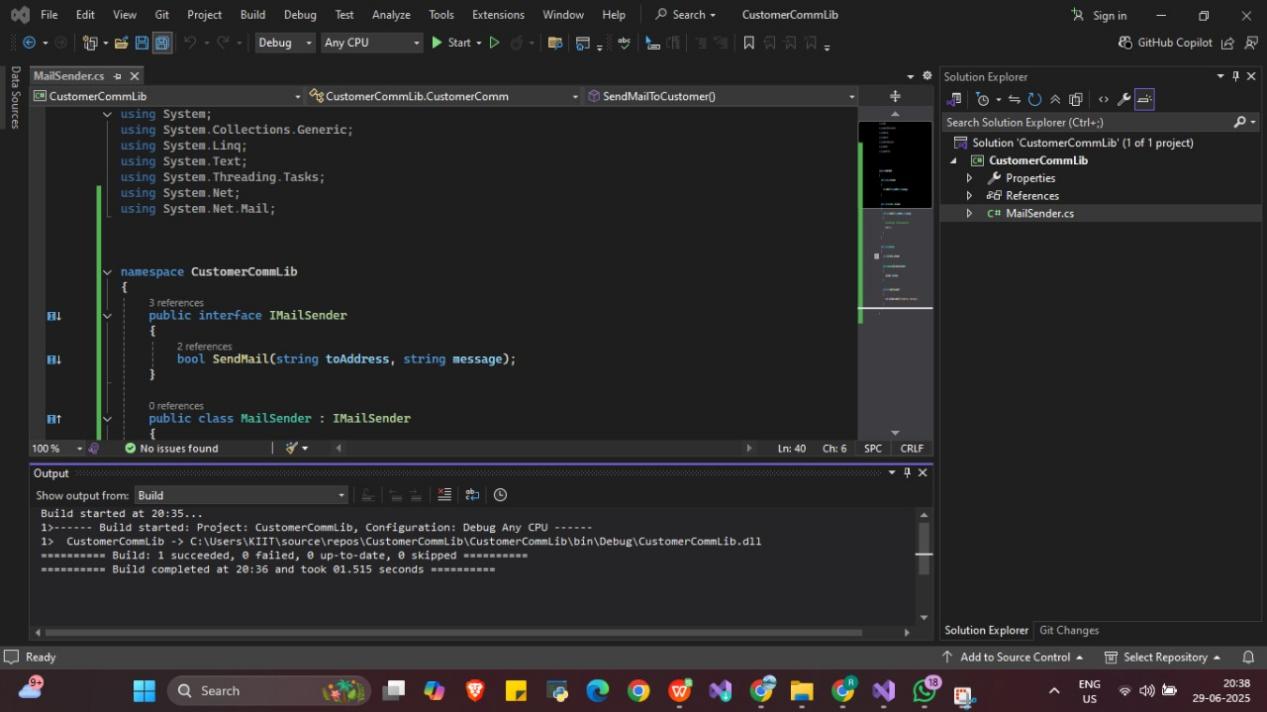
{

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

}

}

}



CustomerCommTests.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using NUnit.Framework;

using Moq;

using CustomerCommLib;

namespace CustomerComm.Tests

{

[TestFixture]

public class CustomerCommTests

{

private Mock<IMailSender> \_mailSenderMock;

private CustomerCommLib.CustomerComm \_customerComm; // ✅ Be explicit if needed

[OneTimeSetUp]

public void Setup()

{

\_mailSenderMock = new Mock<IMailSender>();

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

.Returns(true);

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

}

[Test]

public void SendMailToCustomer\_ReturnsTrue()

{

var result = \_customerComm.SendMailToCustomer();

Assert.That(result, Is.True);

}

}

}

Output

