**Write Testable Code with Moq**

**CODE:**

**UserNotifier .cs**  
  
namespace UserNotifyLib

{

    public class UserNotifier

    {

        private readonly INotificationSender \_sender;

        public UserNotifier(INotificationSender sender)

        {

            \_sender = sender;

        }

        public bool NotifyUser()

        {

            return \_sender.SendNotification("user001@example.com", "Your notification message");

        }

    }

}

**EmailNotificationSender.cs**   
  
using System.Net;

using System.Net.Mail;

namespace UserNotifyLib

{

    public class EmailNotificationSender : INotificationSender

    {

        public bool SendNotification(string recipient, string content)

        {

            MailMessage message = new MailMessage();

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

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

            message.To.Add(recipient);

            message.Subject = "Notification";

            message.Body = content;

            client.Port = 587;

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

            client.EnableSsl = true;

            client.Send(message);

            return true;

        }

    }

}

**INotificationSender.cs**

namespace UserNotifyLib

{

    public interface INotificationSender

    {

        bool SendNotification(string recipient, string content);

    }

}

**UserNotifierTests.cs**  
  
using NUnit.Framework;

using Moq;

using UserNotifyLib;

namespace UserNotifyTests

{

    [TestFixture]

    public class UserNotifierTests

    {

        private Mock<INotificationSender> \_mockSender;

        [OneTimeSetUp]

        public void Init()

        {

            \_mockSender = new Mock<INotificationSender>();

        }

        [Test]

        public void NotifyUser\_ShouldReturnTrue\_WhenSendNotificationSucceeds()

        {

            \_mockSender.Setup(x => x.SendNotification(It.IsAny<string>(), It.IsAny<string>())).Returns(true);

            var notifier = new UserNotifier(\_mockSender.Object);

            bool result = notifier.NotifyUser();

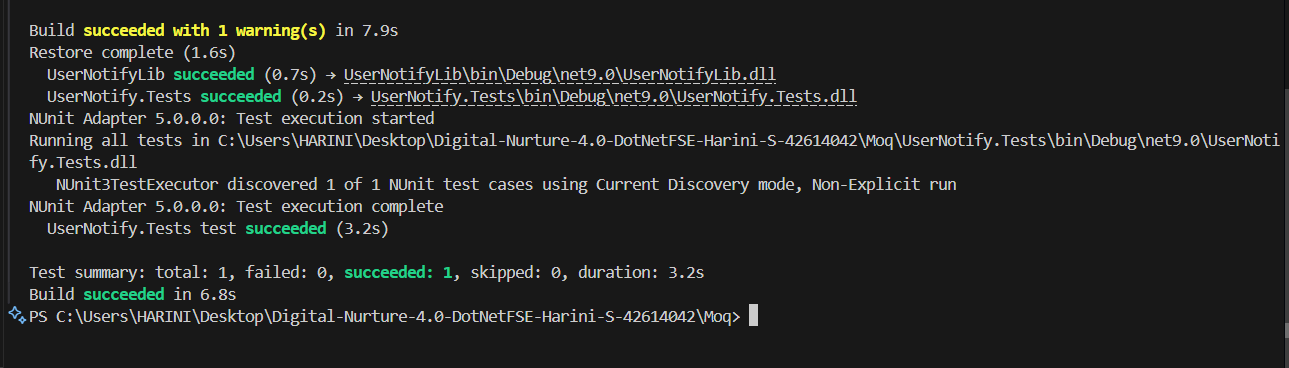
            Assert.That(result, Is.True);

        }

    }

}

**OUTPUT:**



**NUnit-Handson**

**Class1.cs**

namespace CalcLibrary

{

    public class Calculator

    {

        public int Add(int i, int j)

        {

            return i + j;

        }

    }

}

**UnitTest1.cs**

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 Cleanup()

        {

        }

        [Test]

        [TestCase(2, 3, 5)]

        [TestCase(0, 0, 0)]

        [TestCase(-2, -3, -5)]

        public void Add\_WhenCalled\_ReturnsCorrectSum(int i, int j, int expected)

        {

            int result = calc.Add(i, j);

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

        }

    }

}

**OUTPUT:**

