1. NUnit – Handson - TestFixture & Test

**Creating dotnet project:**

dotnet new sln -n SmartCalcSolution

dotnet new classlib -n MathEngine

dotnet new nunit -n MathEngine.Tests

dotnet sln add MathEngine/MathEngine.csproj

dotnet sln add MathEngine.Tests/MathEngine.Tests.csproj

dotnet add MathEngine.Tests reference MathEngine

**Application Logic for Class1.cs:**

namespace MathEngine

{

    public class Calculator

    {

        public int Add(int a, int b)

        {

            return a + b;

        }

    }

}

**Testing code for UnitTest1.cs:**

using NUnit.Framework;

using MathEngine;

namespace MathEngine.Tests

{

[TestFixture]

public class CalculatorTests

{

private Calculator \_calculator;

[SetUp]

public void SetUp()

{

\_calculator = new Calculator();

}

[TearDown]

public void TearDown()

{

\_calculator = null;

}

[TestCase(2, 3, 5)]

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

[TestCase(0, 0, 0)]

public void Add\_GivenTwoNumbers\_ReturnsCorrectSum(int a, int b, int expected)

{

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

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

}

[Test, Ignore("This is a demo of the Ignore attribute")]

public void ThisTestIsIgnored()

{

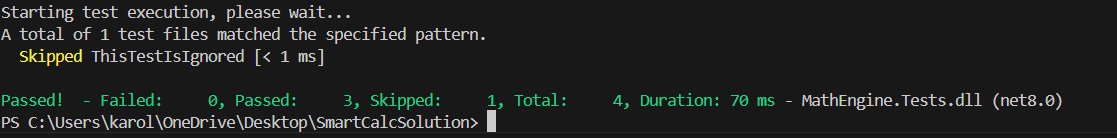
Assert.Fail("This part is ignored.");

}

}

}

**CODE OUTPUT:**



Moq – Handson : 1. Write Testable Code with Moq

**Task 1:**

**Create solution and projects:**

dotnet new sln -n CustomerCommSolution

cd CustomerCommSolution

dotnet new classlib -n CustomerCommLib

dotnet new classlib -n CustomerComm.Tests

dotnet sln add CustomerCommLib/CustomerCommLib.csproj

dotnet sln add CustomerComm.Tests/CustomerComm.Tests.csproj

dotnet add CustomerComm.Tests reference CustomerCommLib

**IMailSender code:**

namespace CustomerCommLib

{

public interface IMailSender

{

bool SendMail(string toAddress, string message);

}

}

**MailSender Code:**

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\_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;

}

}

}

**CustomerCommTests Code:**

namespace CustomerCommLib

{

public class CustomerComm

{

private IMailSender \_mailSender;

public CustomerComm(IMailSender mailSender)

{

\_mailSender = mailSender;

}

public bool SendMailToCustomer()

{

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

}

}

}

**Task 2:**

**Download Packages:**

cd CustomerComm.Tests

dotnet add package NUnit

dotnet add package NUnit3TestAdapter

dotnet add package Moq

dotnet add package Microsoft.NET.Test.Sdk

**Code for CustomerCommTests :**

using NUnit.Framework;

using Moq;

using CustomerCommLib;

namespace CustomerCommTests

{

[TestFixture]

public class CustomerCommTests

{

private Mock<IMailSender> \_mock;

[OneTimeSetUp]

public void Setup()

{

\_mock = new Mock<IMailSender>();

}

[Test]

public void SendMailToCustomer\_ReturnsTrue()

{

// Arrange

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

CustomerComm customerComm = new CustomerComm(\_mock.Object);

// Act

bool result = customerComm.SendMailToCustomer();

// Assert

Assert.That(result, Is.True);

}

}

}

**Build and Test:**

dotnet build

dotnet test

**CODE OUTPUT:**

