**Skill:-NUnit and Moq**

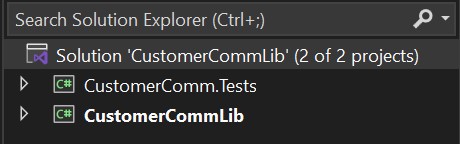
**Filename:- 1. Moq-Handson**

**Name:-1. Write Testable Code with Moq**

## **Task1**

Create a **Class Library (Language C#)** project using Visual Studio IDE, and name it as **CustomerCommLib.**

**OUTPUT:-**

****

**Code For MailSender.cs**

// 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;

}

}

}

**According to question** Include the following namespaces with ‘using’ directive.

* **System.Net**
* **System.Net.Mail**

***It is been included in code above…….***

**Code for CustomerComm.cs**

namespace CustomerCommLib

{

public class CustomerComm

{

private readonly IMailSender \_mailSender;

private readonly ICustomerRepository \_repository;

public CustomerComm(IMailSender mailSender, ICustomerRepository repository)

{

\_mailSender = mailSender;

\_repository = repository;

}

public bool SendMailToCustomer(int customerId)

{

string email = \_repository.GetCustomerEmail(customerId);

return \_mailSender.SendMail(email, "Some Message");

}

}

}

**Code for CustomerRepository.cs**

namespace CustomerCommLib

{

public class CustomerRepository : ICustomerRepository

{

public string GetCustomerEmail(int customerId)

{

// Simulate database logic

return "cust123@abc.com";

}

}

}

**Code for ICustomerRepository.cs**

namespace CustomerCommLib

{

public interface ICustomerRepository

{

string GetCustomerEmail(int customerId);

}

}

**Code for FileProcessor.cs**

namespace CustomerCommLib

{

public class FileProcessor

{

private readonly IFileReader \_fileReader;

public FileProcessor(IFileReader fileReader)

{

\_fileReader = fileReader;

}

public string GetFileContent(string path)

{

return \_fileReader.ReadFile(path);

}

}

}

**Code for FileReader.cs**

using System.IO;

namespace CustomerCommLib

{

public class FileReader : IFileReader

{

public string ReadFile(string path)

{

return File.ReadAllText(path);

}

}

}

Code for IFileReader.cs

namespace CustomerCommLib

{

public interface IFileReader

{

string ReadFile(string path);

}

}

## **Task2**

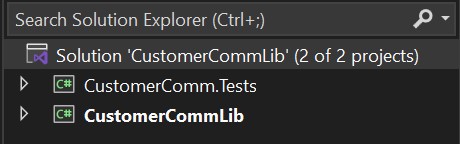
Create a new class library project called **CustomerComm.Tests** and add the following external dependencies to it using **NuGet Package Manager.**

* + NUnit
  + NUnit Test Adapter
  + Moq

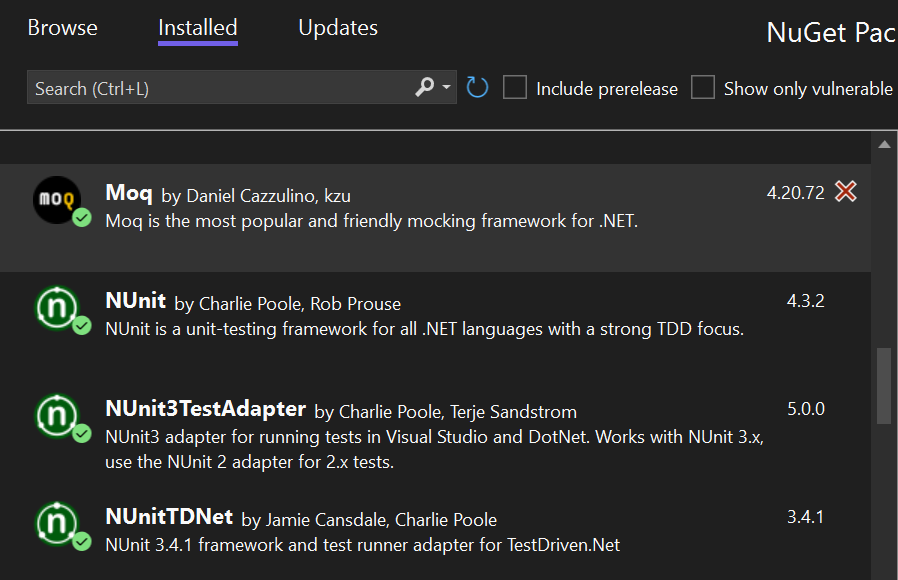
**OUTPUT:-**

***Screenshots of added project Customer Comm.Tests and added packages.***

**1.**

****

**2.**



**In CustomerComm.Tests project**

**Code for CustomerCommTests.cs**

using NUnit.Framework;

using Moq;

using CustomerCommLib;

namespace CustomerComm.Tests

{

[TestFixture]

public class CustomerCommTests

{

private Mock<IMailSender> \_mockMailSender;

private Mock<ICustomerRepository> \_mockRepository;

private CustomerCommLib.CustomerComm \_customerComm;

[OneTimeSetUp]

public void Init()

{

\_mockMailSender = new Mock<IMailSender>();

\_mockRepository = new Mock<ICustomerRepository>();

\_mockMailSender

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

.Returns(true);

\_mockRepository

.Setup(x => x.GetCustomerEmail(It.IsAny<int>()))

.Returns("test@abc.com");

\_customerComm = new CustomerCommLib.CustomerComm(\_mockMailSender.Object, \_mockRepository.Object);

}

[Test]

public void SendMailToCustomer\_ShouldReturnTrue\_WhenMocked()

{

var result = \_customerComm.SendMailToCustomer(123);

NUnit.Framework.Assert.IsTrue(result);

}

}

}

**Code for FileProcessorTests.cs**

using NUnit.Framework;

using Moq;

using CustomerCommLib;

namespace CustomerComm.Tests

{

[TestFixture]

public class FileProcessorTests

{

private Mock<IFileReader> \_mockFileReader;

private FileProcessor \_fileProcessor;

[SetUp]

public void Init()

{

\_mockFileReader = new Mock<IFileReader>();

\_mockFileReader

.Setup(x => x.ReadFile(It.IsAny<string>()))

.Returns("This is mock file content");

\_fileProcessor = new FileProcessor(\_mockFileReader.Object);

}

[Test]

public void GetFileContent\_ShouldReturnMockedContent()

{

var result = \_fileProcessor.GetFileContent("fake\_path.txt");

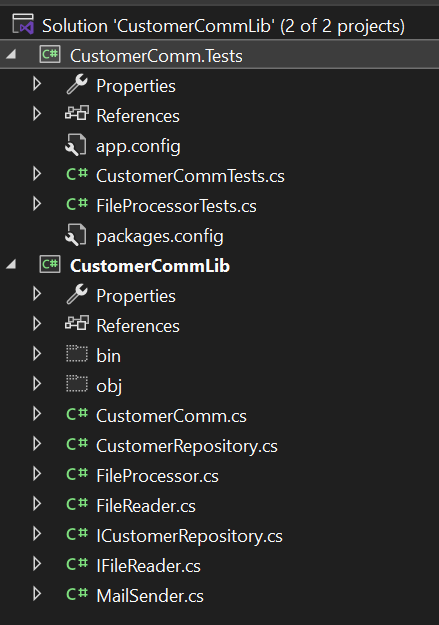
Assert.AreEqual("This is mock file content", result);

}

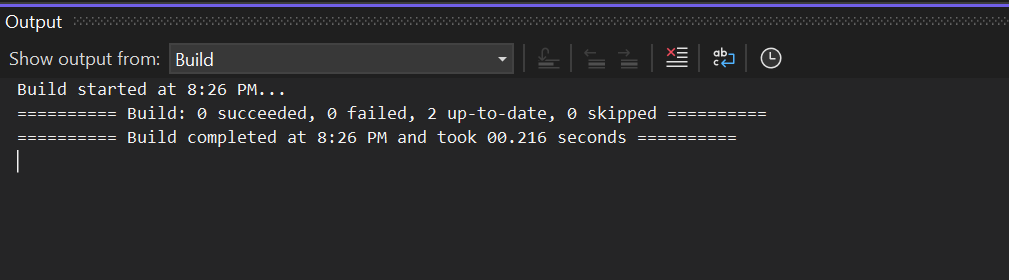
}

}

**Full Structure of Created File.**



**Finally Biulding of Project**

**OUTPUT:-**

**Passing Test Case**

**OUTPUT:-**