# Unit Testing with NUnit and Moq - Email Scenario

## Overview

This guide demonstrates how to create a unit-testable class library using dependency injection and mock external services like email using Moq and NUnit in C#.  
The application mimics sending emails using a class `MailSender` which communicates with an SMTP server. Instead of sending real emails during testing, we mock the `IMailSender` interface.

## Project Structure

The solution contains three projects:  
1. CustomerCommLib - Class Library for core logic  
2. CustomerComm.Tests - NUnit test project using Moq  
3. CustomerCommApp - Console application with Main() entry point

## CustomerCommLib - Code

IMailSender and 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;  
 }  
 }  
}

CustomerComm.cs

namespace CustomerCommLib  
{  
 public class CustomerComm  
 {  
 IMailSender \_mailSender;  
  
 public CustomerComm(IMailSender mailSender)  
 {  
 \_mailSender = mailSender;  
 }  
  
 public bool SendMailToCustomer()  
 {  
 \_mailSender.SendMail("cust123@abc.com", "Some Message");  
 return true;  
 }  
 }  
}

## CustomerComm.Tests - Unit Test Code

CustomerCommTests.cs

using NUnit.Framework;  
using Moq;  
using CustomerCommLib;  
  
namespace CustomerComm.Tests  
{  
 [TestFixture]  
 public class CustomerCommTests  
 {  
 private Mock<IMailSender> \_mockMailSender;  
 private CustomerComm \_customerComm;  
  
 [OneTimeSetUp]  
 public void Init()  
 {  
 \_mockMailSender = new Mock<IMailSender>();  
 \_customerComm = new CustomerComm(\_mockMailSender.Object);  
 }  
  
 [Test]  
 public void SendMailToCustomer\_ShouldReturnTrue()  
 {  
 \_mockMailSender.Setup(x => x.SendMail(It.IsAny<string>(), It.IsAny<string>())).Returns(true);  
  
 var result = \_customerComm.SendMailToCustomer();  
  
 Assert.IsTrue(result);  
 }  
 }  
}

## CustomerCommApp - Program.cs

using System;  
using CustomerCommLib;  
  
namespace CustomerCommApp  
{  
 class Program  
 {  
 static void Main(string[] args)  
 {  
 IMailSender mailSender = new MailSender(); // Will try to send real email  
 CustomerComm comm = new CustomerComm(mailSender);  
 bool result = comm.SendMailToCustomer();  
 Console.WriteLine("Mail send status: " + result);  
 }  
 }  
}

## NuGet Package Installation

For CustomerComm.Tests project:  
1. Right-click on the project > Manage NuGet Packages.  
2. Install the following:  
 - NUnit  
 - NUnit3TestAdapter  
 - Moq  
  
Then rebuild the solution.

