**Superset id:- 6363303**

**Exercise 1: Moq-Handson**

**CustomerComm.cs code:-**

**namespace CustomerLibrary**

**{**

**public class CustomerComm**

**{**

**private readonly IMailSender \_mailSender;**

**public CustomerComm(IMailSender mailSender)**

**{**

**\_mailSender=mailSender;**

**}**

**public bool SendMailToCustomer()**

**{**

**return \_mailSender.SendMail("cust@Gmail.com","Hello Message");**

**}**

**}**

**}**

**IMailSender.cs Code:-**

**namespace CustomerLibrary**

**{**

**public interface IMailSender**

**{**

**bool SendMail(string toAddress, string message);**

**}**

**}**

**MailSender.cs Code:-**

**using System.Net;**

**using System.Net.Mail;**

**namespace CustomerLibrary**

**{**

**public class MailSender:IMailSender**

**{**

**public bool SendMail(string toAddress,string message)**

**{**

**try**

**{**

**var mail=new MailMessage();**

**var smtp=new SmtpClient("smtp.gmail.com");**

**mail.From = new MailAddress("ayushagrawal.5288@gmail.com");**

**mail.To.Add(toAddress);**

**mail.Subject="Test Mail";**

**mail.Body=message;**

**smtp.Port=587;**

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

**smtp.EnableSsl=true;**

**smtp.Send(mail);**

**return true;**

**}**

**catch**

**{**

**return false;**

**}**

**}**

**}**

**}**

**CustomerCommTests.cs Code:-**

**using NUnit.Framework;**

**using Moq;**

**using CustomerLibrary;**

**namespace CustomerLibrary.Tests**

**{**

**public class CustomerCommTests**

**{**

**[Test]**

**public void MailStatus()**

**{**

**var mockMailSender=new Mock<IMailSender>(); mockMailSender.Setup(sender=>sender.SendMail(It.IsAny<string>(),It.IsAny<string>())).Returns(true);**

**var customerComm=new CustomerComm(mockMailSender.Object);**

**var result=customerComm.SendMailToCustomer();**

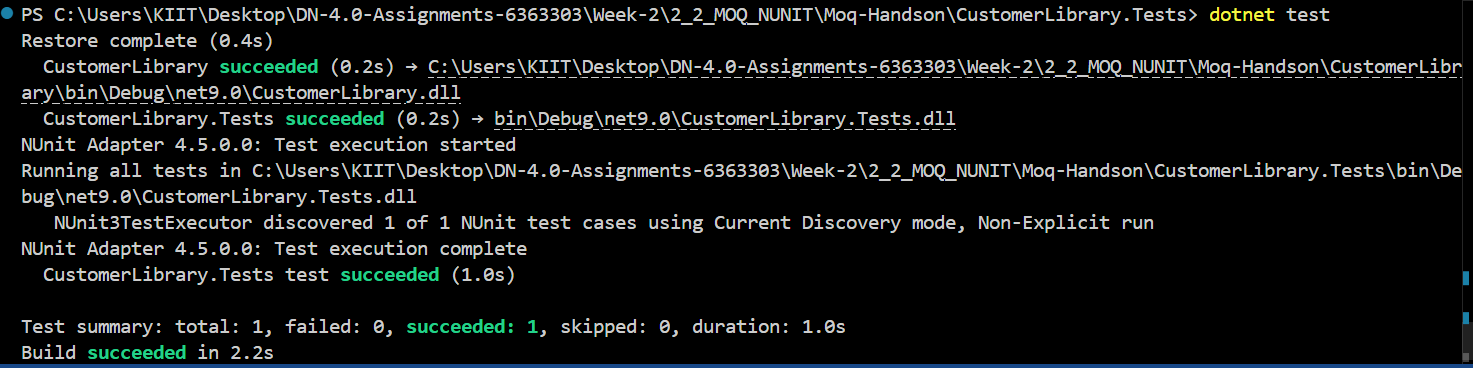
**Assert.IsTrue(result);**

**}**

**}**

**}**

**Output:-**

****

**What is Mocking:-**

Mocking means creating a fake version of something, like a service or class, that we use during testing.

For example, if your code sends emails, you don’t want to send real emails every time you run a test. So, we create a "mock" that pretends to send the email and just says “sent” without doing anything.

**Why Use Mocks in Unit Testing:-**

* Real systems (like databases, email servers, files) are **slow** or can **fail**.
* Mocks help us **test only our code**, not outside systems.
* Mocks make tests run **faster and more reliably**.
* They help us check **if the code is behaving correctly**, like whether a method was called or not.

**What is Dependency Injection (DI):-**

Sometimes our classes need other classes to work — like a CustomerService needs an EmailService.

In Dependency Injection, instead of creating the EmailService inside CustomerService, we **pass it from outside** (like giving a tool to a worker instead of the worker buying it themselves).

**Types of DI:-**

* **Constructor Injection: Dependencies are passed via the constructor.**
* **Method Injection: Dependencies are passed to methods as parameters.**
* **Property Injection: Dependencies are set via public properties.**

### **How DI Helps Testing:-**

* **Allows passing mock objects during tests.**
* **Makes the class loosely coupled and more modular.**
* **Makes code easier to maintain and test.**

### **What is Moq:-**

**Moq is a popular mocking library for .NET. It helps create fake objects for interfaces and verify their behavior.**

### **Steps to Create Testable Code:-**

1. **Create an interface for external services (like IMailSender)**
2. **Inject the interface using DI**
3. **Use Moq in your test project to create a mock**
4. **Write tests using NUnit or xUnit**

**Ayush Kumar (6363303)**