**Moq HandsOn Exercise-1**

## **Scenario**

You are tasked to write a unit test code for the below scenario.

The application in which you are teamed up with, deals with a mail server communication in which your application tries to send mail to its users upon every transaction. Your role is to write unit testing the module that contains send mail functionality. You wanted to perform testing the module without sending any email.

After investigating the problem scenario, you found a solution and that is creating **mock** objects of these external dependencies in the unit testing project so that you can achieve speedier test execution and loose coupling of code.

**UNIT TEST CODE:**

using Moq;

using NUnit.Framework;

using CustomerCommLib;

namespace CustomerCommTests

{

[TestFixture]

public class CustomerCommTests

{

private Mock<IMailSender> \_mockMailSender;

private CustomerComm \_customerComm;

[OneTimeSetUp]

public void Init()

{

// Arrange: Create mock and setup

\_mockMailSender = new Mock<IMailSender>();

// Setup mock to always return true for any input

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

// Inject mock into CustomerComm

\_customerComm = new CustomerComm(\_mockMailSender.Object);

}

[TestCase]

public void SendMailToCustomer\_ShouldReturnTrue\_WhenMocked()

{

// Act

var result = \_customerComm.SendMailToCustomer();

// Assert

Assert.That(result, Is.True);

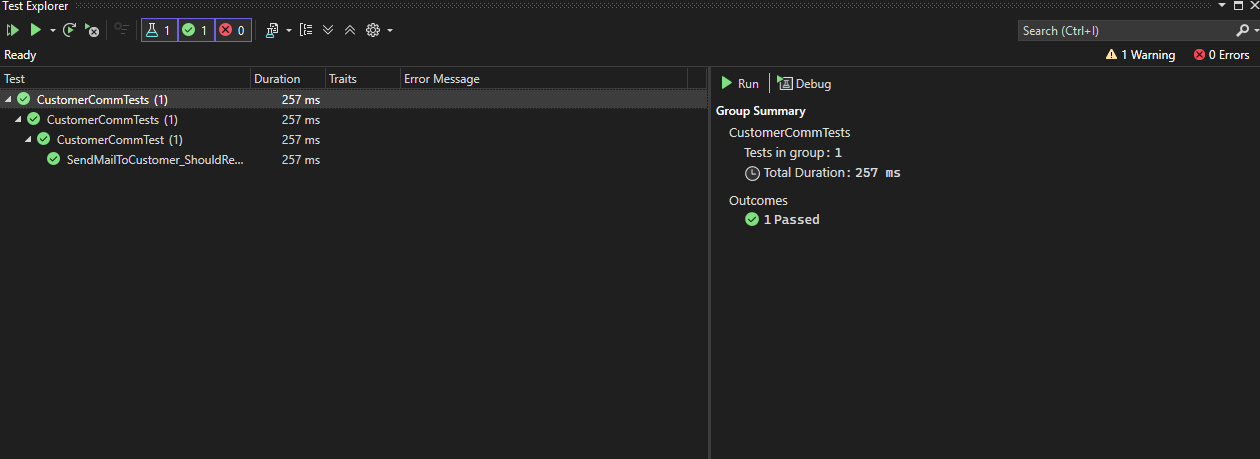
}

}

}

}

**UNIT TEST RUN:**

****