**Moq Testing, Task – 1**

**SOLUTION -> CustomerCommLib -> CustomerComm.cs**

namespace CustomerCommLib

{

public class CustomerComm

{

private readonly IMailSender \_mailSender;

public CustomerComm(IMailSender mailSender)

{

\_mailSender = mailSender;

}

public bool SendMailToCustomer()

{

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

}

}

}

**SOLUTION -> CustomerCommLib -> IMailSender.cs**

namespace CustomerCommLib

{

public interface IMailSender

{

bool SendMail(string toAddress, string message);

}

}

**SOLUTION -> CustomerCommLib -> MailSender.cs**

using System.Net;

using System.Net.Mail;

namespace CustomerCommLib

{

public class MailSender : IMailSender

{

public bool SendMail(string toAddress, string message)

{

var mail = new MailMessage("your\_email\_address@gmail.com", toAddress)

{

Subject = "Test Mail",

Body = message

};

var smtp = new SmtpClient("smtp.gmail.com", 587)

{

Credentials = new NetworkCredential("username", "password"),

EnableSsl = true

};

smtp.Send(mail);

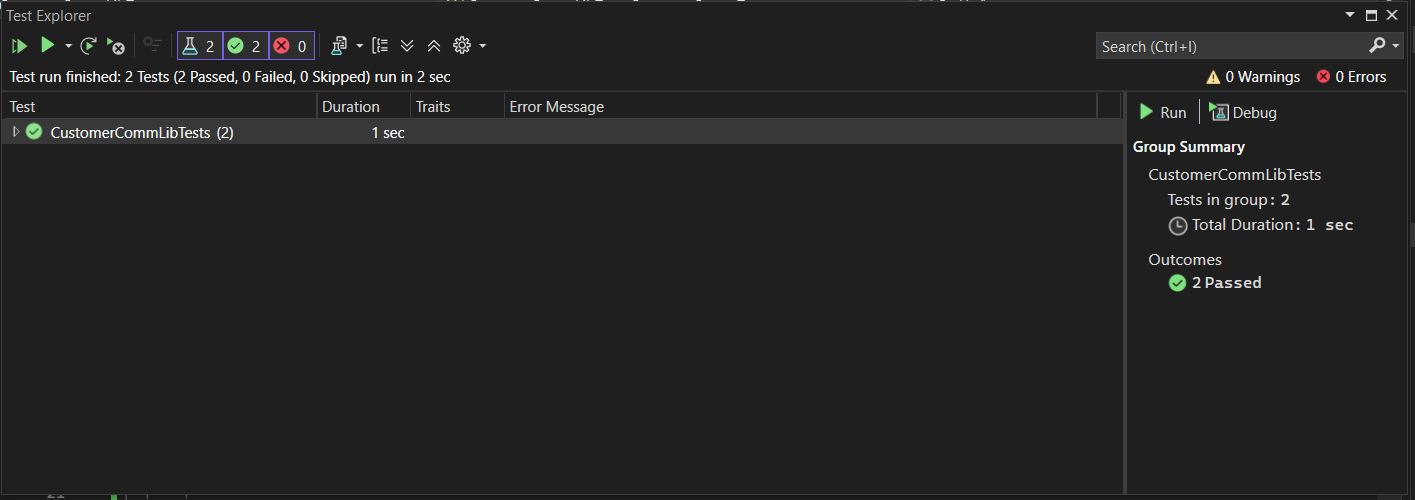
return true;

}

}

}

**OUTPUT :-**

****

**TASK – 2**

**SOLUTION -> CustomerComm.Tests -> CustomerCommTests.cs**

using NUnit.Framework;

using Moq;

using CustomerCommLib;

namespace CustomerComm.Tests

{

[TestFixture]

public class CustomerCommTests

{

private Mock<IMailSender> \_mailSenderMock;

private CustomerCommLib.CustomerComm \_customerComm;

[OneTimeSetUp]

public void Init()

{

\_mailSenderMock = new Mock<IMailSender>();

\_mailSenderMock

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

.Returns(true);

\_customerComm = new CustomerCommLib.CustomerComm(\_mailSenderMock.Object);

}

[TestCase]

public void SendMailToCustomer\_ReturnsTrue()

{

var result = \_customerComm.SendMailToCustomer();

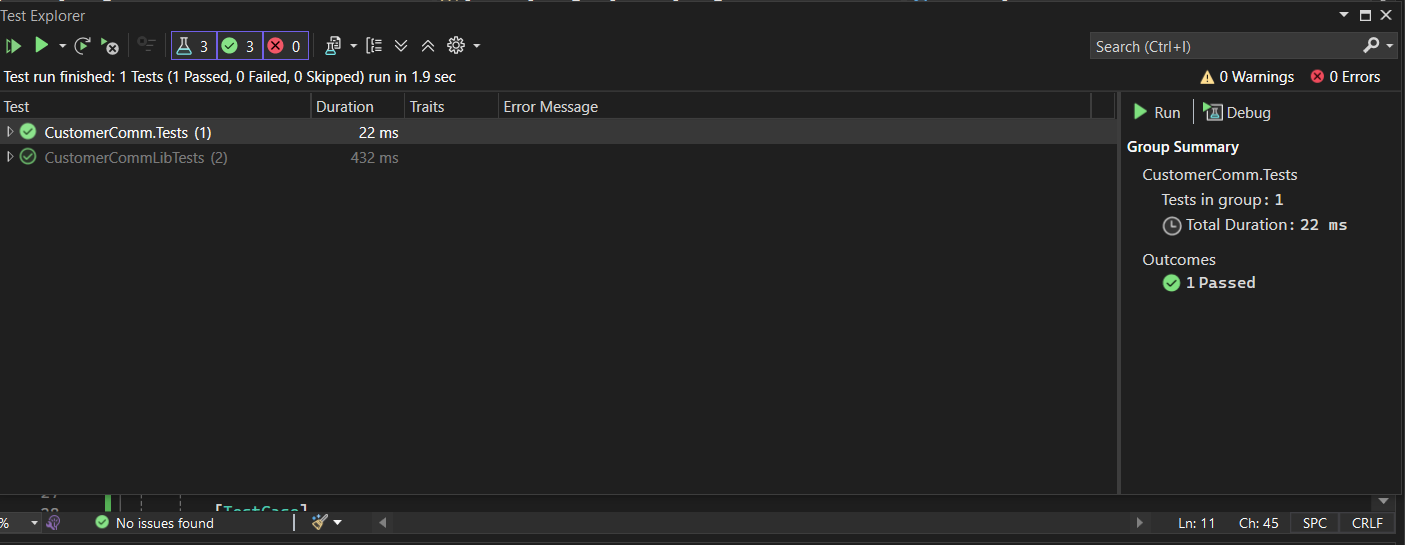
Assert.That(result, Is.True, "Expected true when mail is mocked");

}

}

}

**OUTPUT :-**

****