**Week 2 - NUnit and Moq**

**1. NUnit-Handson**

**TestFixture & Test**

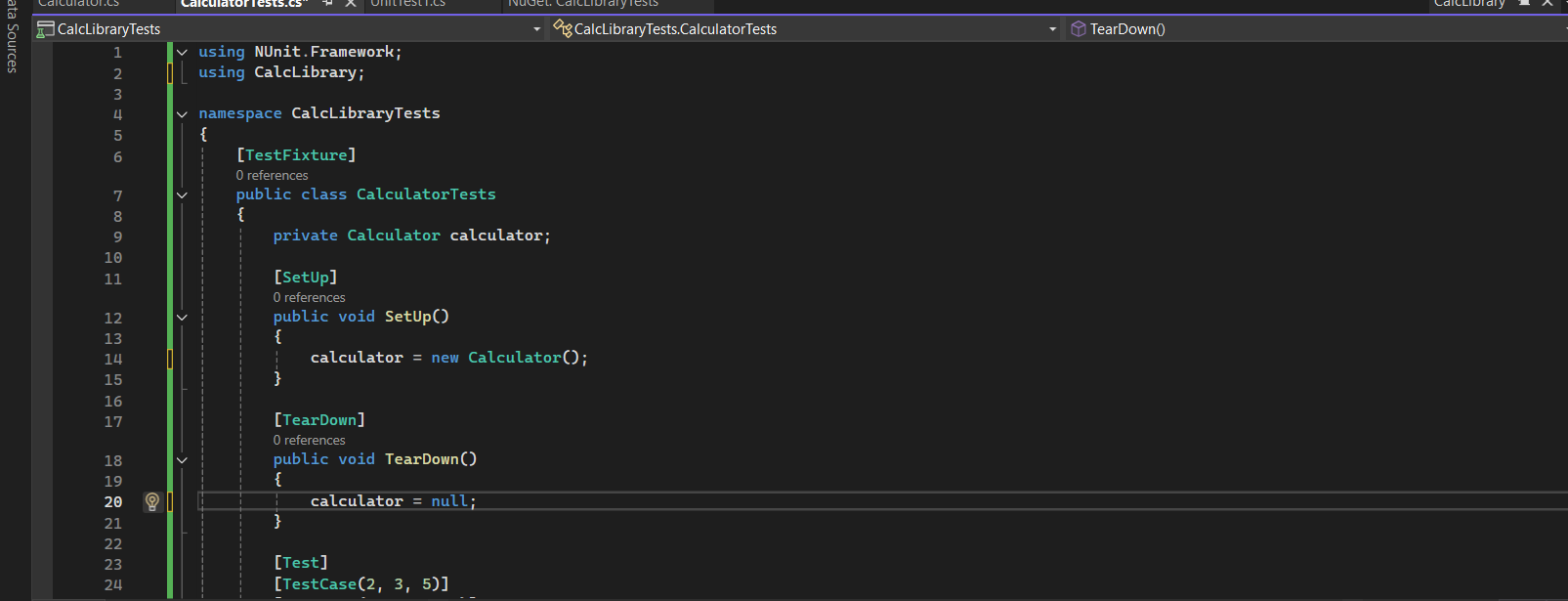
Please download the application available [here](https://cognizantonline.sharepoint.com/:u:/r/sites/GTP-Solutions/Gencsharepath/Shared%20Documents/Internship2020/FSE/DotNet/02%20-%20NUnit,%20C%23%204.5,%20ASP.Net%20Core/Handson/CalcLibrary.zip?csf=1&e=aLxB66). This will be used to write Unit test cases  
  
Follow the steps listed below to write the NUnit test cases for the application.

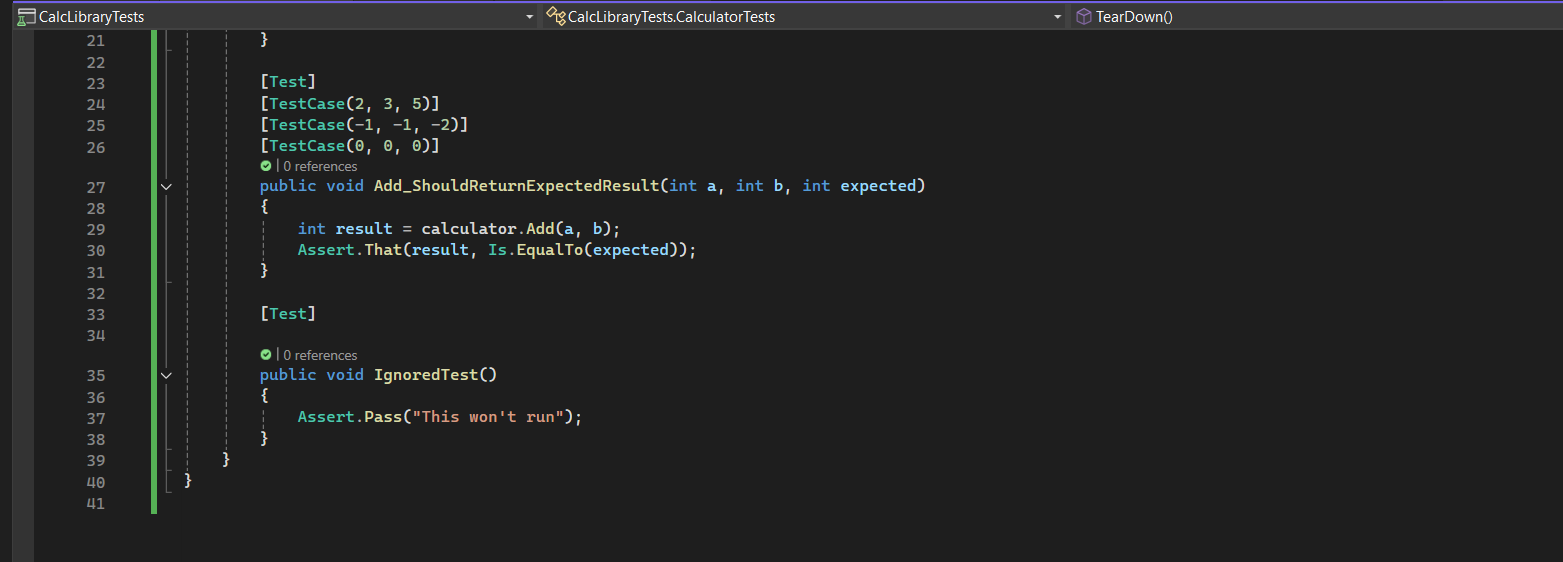
* Create a Unit test project(.Net Framework) in the solution provided.
* Add the CalcLibrary project as reference
* Create a class “CalculatorTests” to write all the test cases for the methods in the solution
* Use the ‘TestFixture’, ‘SetUp’ and ‘TearDown’ attributes, to declare, initialize and cleanup activities respectively
* Create a Test method to check the addition functionality
* Use the ‘TestCase’ attribute to send the inputs and the expected result
* Use Assert.That to check the actual and expected result match

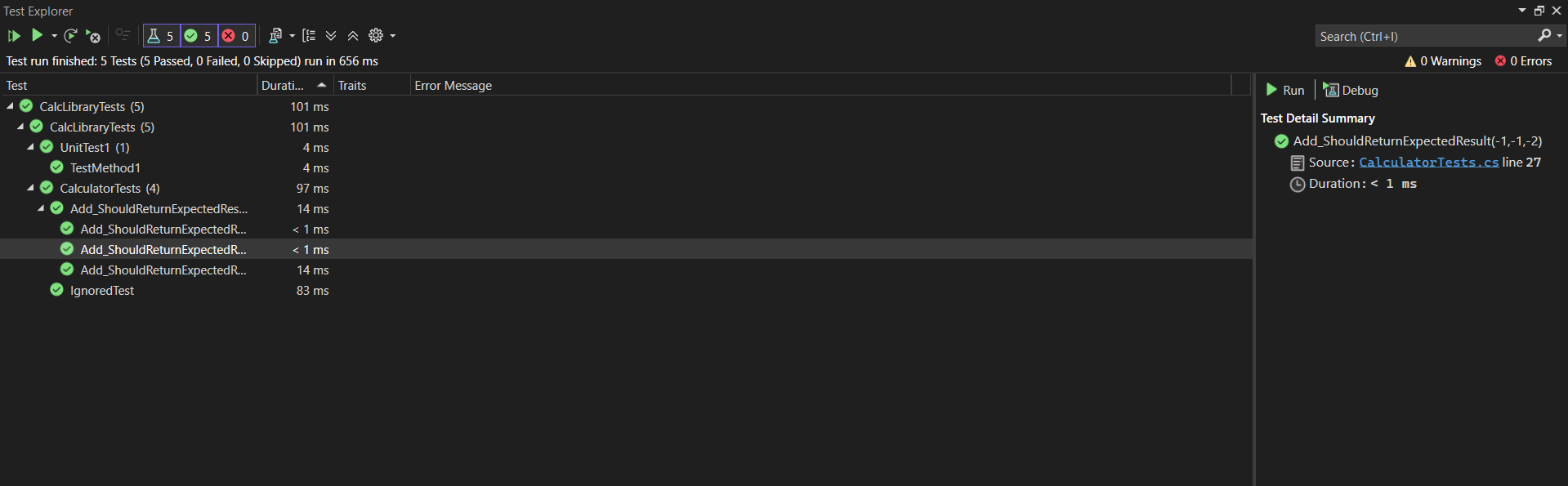
**Code Implementation:**

Created a Unit test Project and added a reference to the CalcLibrary.

Created a new class called CalculatorTests.cs in my project.

**CalculatorTests.cs**



**Output:**  


**1. Moq-Handson**

**1. Write Testable Code with Moq**

**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.

**Task 1:**

* Created a **Class Library** named CustomerCommLib.
* Defined an **interface IMailSender** with a SendMail() method.
* Implemented the interface in a class called **MailSender** using SmtpClient to send real emails.
* Created a class **CustomerComm** that **injects IMailSender** through its constructor.
* The method SendMailToCustomer() calls the injected SendMail() method.
* This structure makes the code **testable** by allowing **mocking** of the SendMail() method during unit testing.

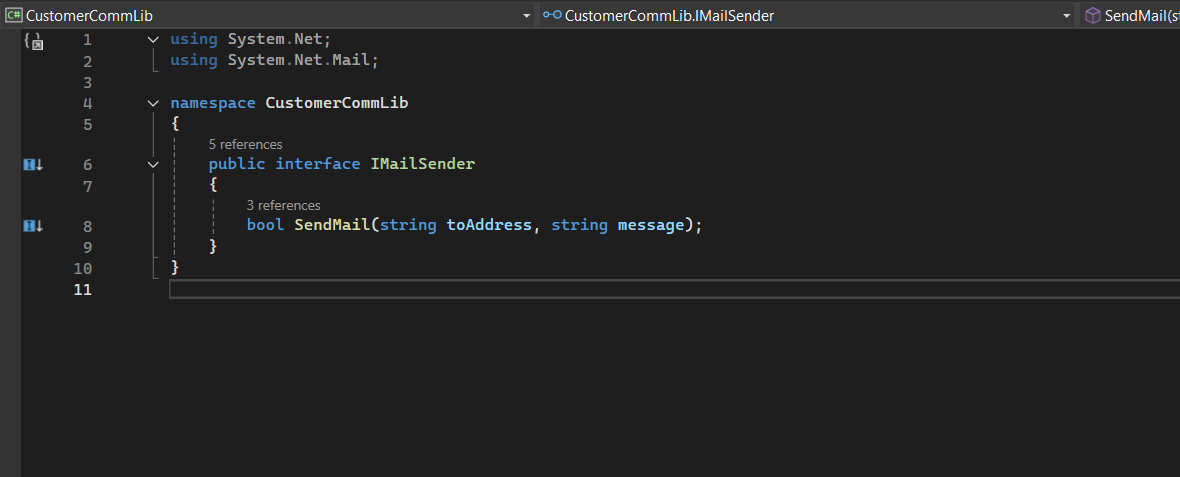
**Task 2 :**

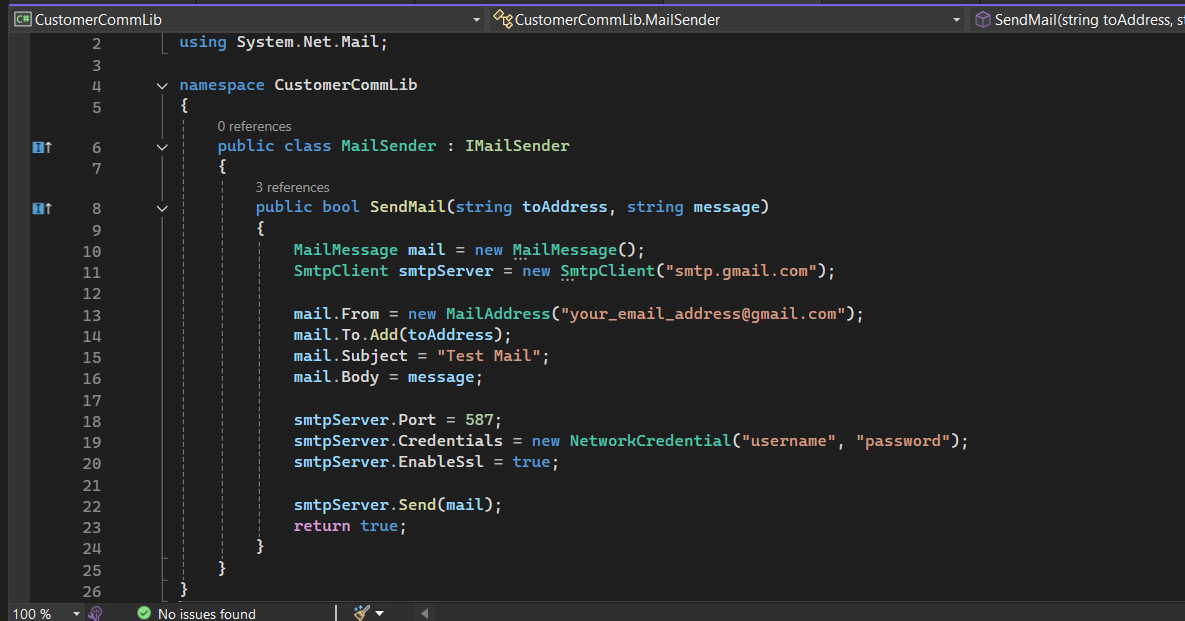
In this task, you will create unit test project which make use of NUnit framework and Moq.

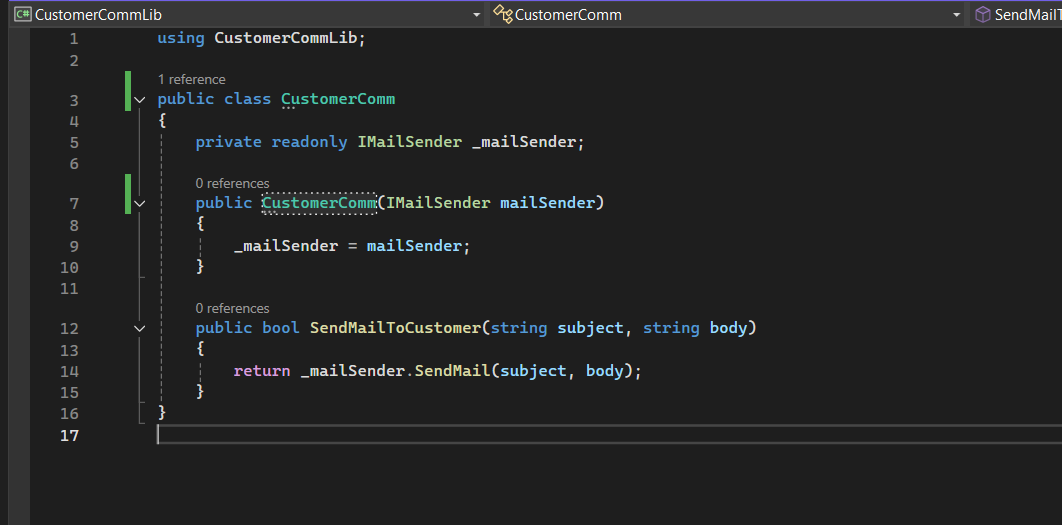
* Create a new class library project called **CustomerComm.Tests** and add the following external dependencies to it using **NuGet Package Manager.**
  + NUnit
  + NUnit Test Adapter
  + Moq
* Add the references of assemblies as appropriate including **CustomerCommLib.**
* Write unit test code and **mock** the **MailSender (IMailSender)** class.
* Use **TestFixture**, **OneTimeSetUp** and **TestCase** attribute classes on top of test class, init method and test method respectively.
* **Configure** the mock object in such away that **SendMail()** method will accept any two string arguments and always return true when **SendMailToCustomer()** gets invoked.
* Finally **assert** the return value to “true”.

**Code Implementation:**

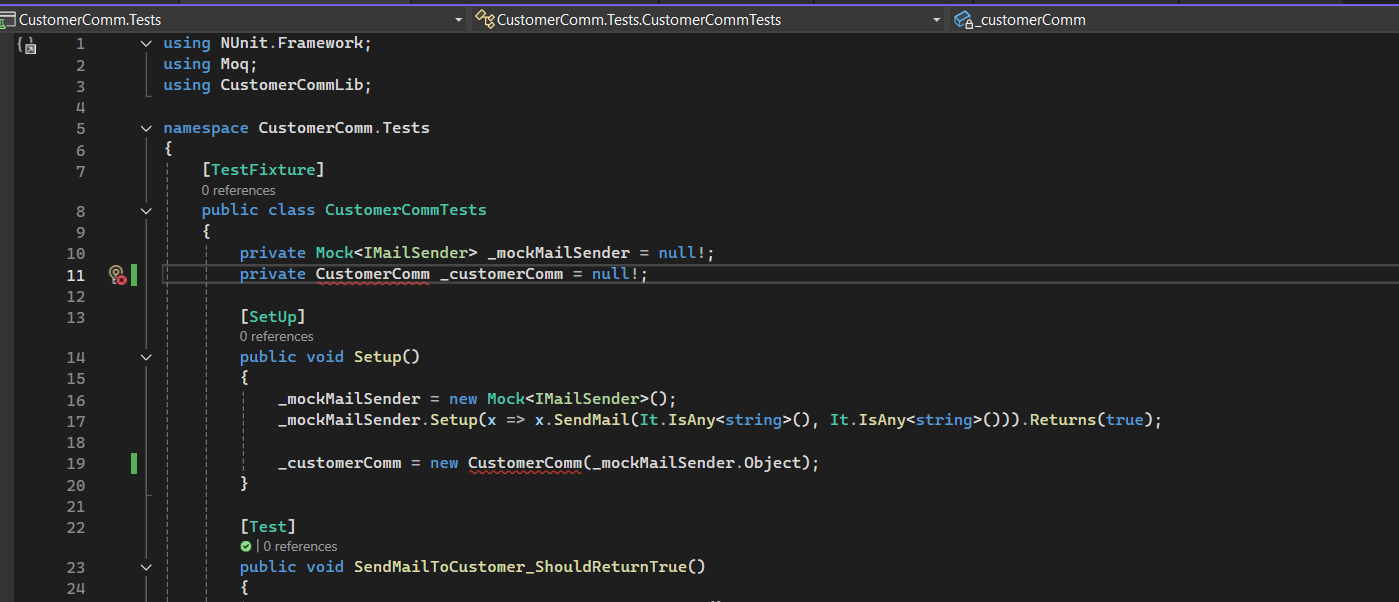
1. **Write Code in CustomerCommLib**

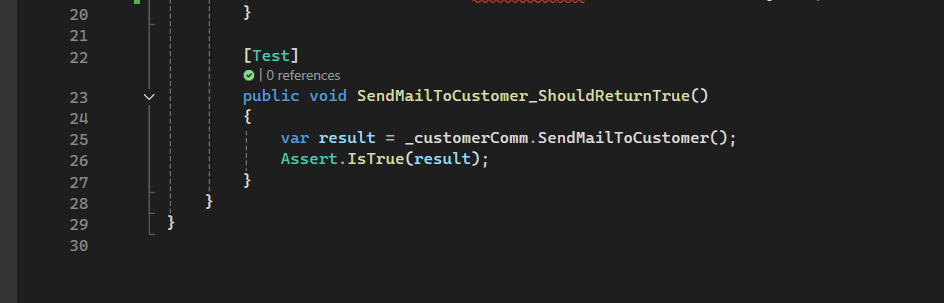
**IMailSender.cs**  
  


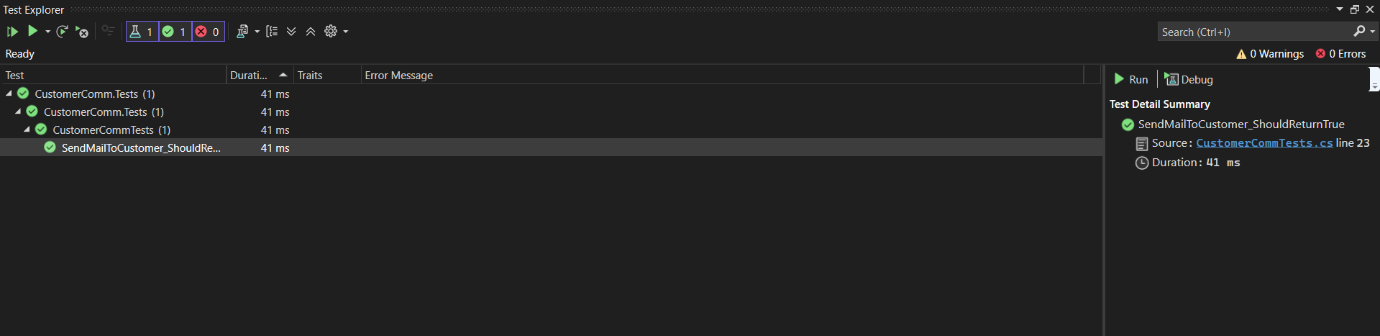
**MailSender.cs**  
  


**CustomerComm.cs**  
  


1. **Write Test in CustomerComm.Tests**

**CustomerCommTests.cs  
  
**



**Output:  
**