**C# N-UNIT TEST FRAMEWORK**

Unit testing is a software testing method by which individual units of source code, such as functions, methods, and class are tested to determine whether they are fit for use.

In unit test framework, each smallest testable part of an application is considered as a unit. In our case, the smallest part is each function in every file of C# project.

For every function, there should a positive and negative case. Positive case refers to the case of which the test case should be written ensuring that users can perform appropriate action when using valid data. Negative case refers to the case of which the test case should be expecting the function to do appropriate action if the user enters the invalid data. Along with these two test cases, we can check for different scenarios with respect to the expect output.

C# and .NET Unit Test Landscape: Testing Frameworks and Tools

There are several testing frameworks and tools available for unit testing in C# and .NET, but the most popular ones are:

xUnit: A modern, extensible testing framework that focuses on simplicity and ease of use. It is often considered the de facto choice for unit testing in .NET Core.

NUnit: A widely used, well-established testing framework with a rich feature set and extensive plugin ecosystem. It has a long history and many legacy .NET projects use it.

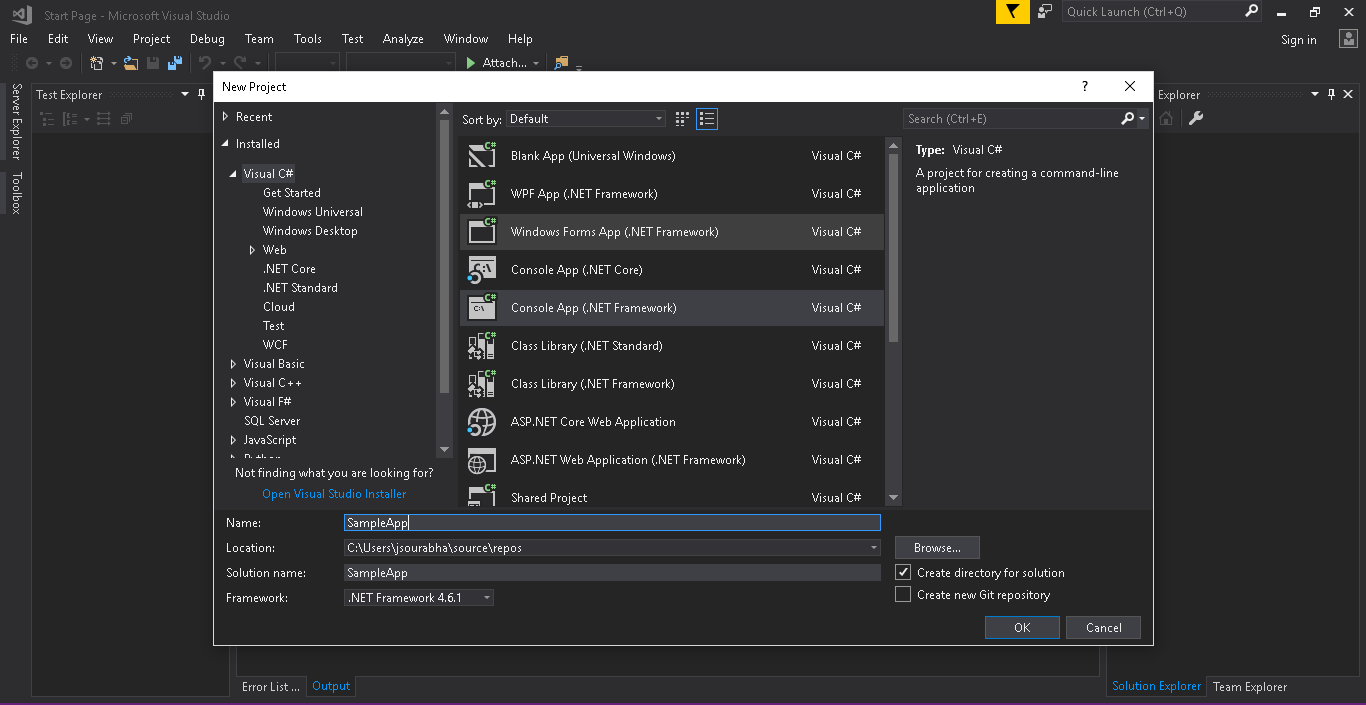
MSTest: The default testing framework provided by the Microsoft Visual Studio suite, offering tight integration with Visual Studio, and backed by Microsoft support.

Moq: A powerful mocking library specifically designed for .NET, allowing developers to create mock objects for isolated testing of units that interact with external dependencies.

Steps for Writing NUnit Testing using C#

Step1:

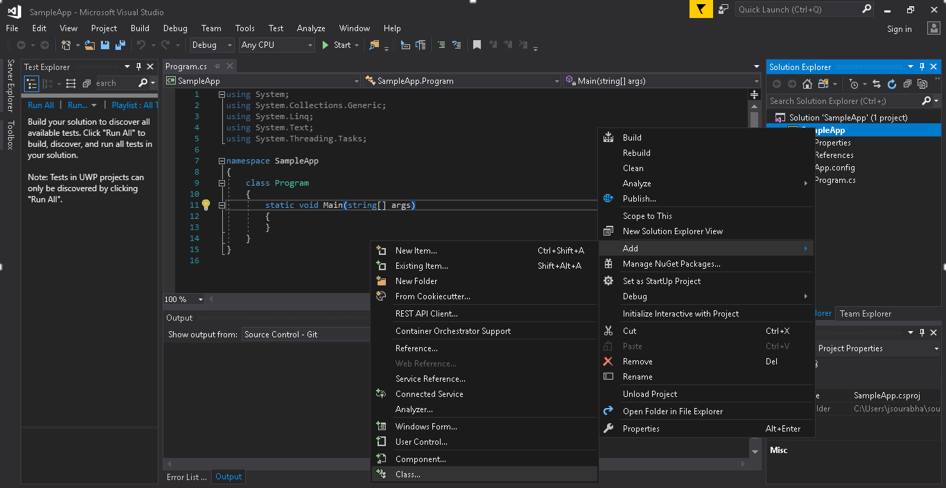
Open visual studio 2017 and create a new file using C#, using Ctrl + shift + N or from file tab



***Figure 1: New project***

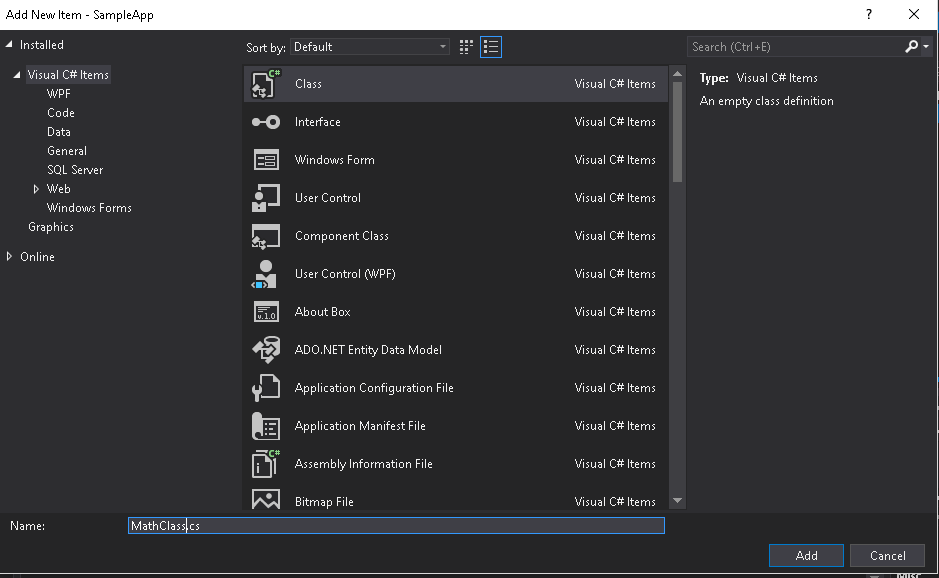
Step 2:

Once after create smapleApp project, right and add class



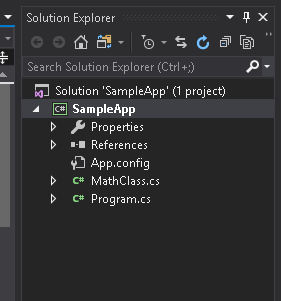
***Figure 1: Add new class***

Click on visual C# Items, and select C# class from the list. Give a class name (ex: MathClass.cs) then click on Add button.



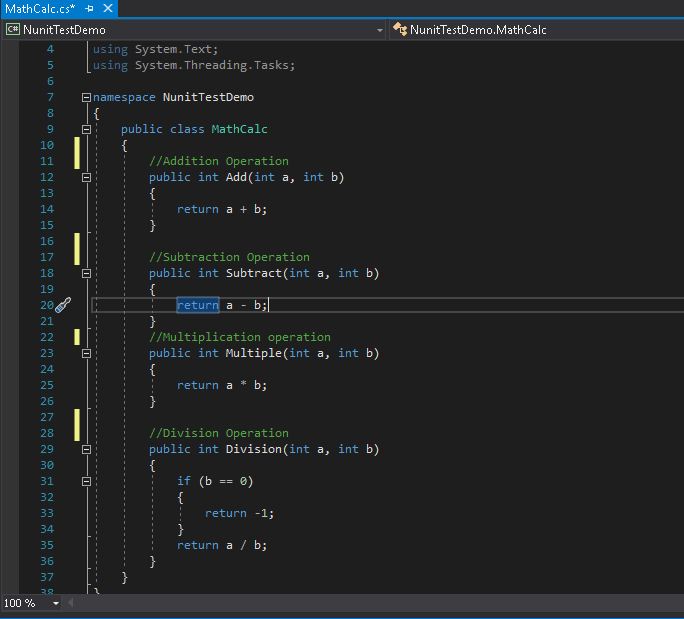
***Figure 3: class creation with class Name***

Verify, newly created class exist in the project folder



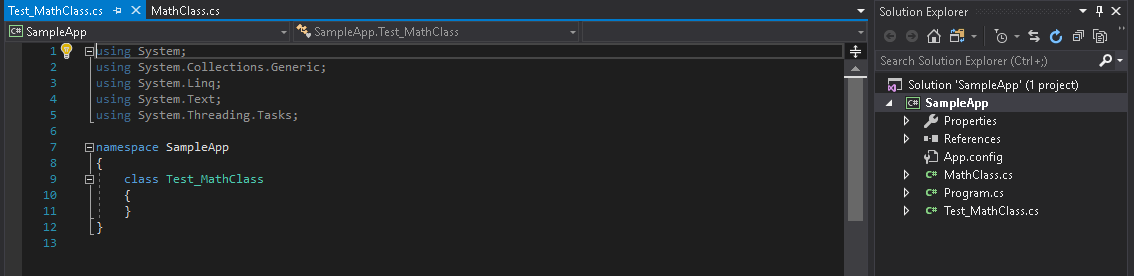
***Figure 4: Newly created class***

Step2: Open newly created file MathClass.cs and write basic mathematical operation functions



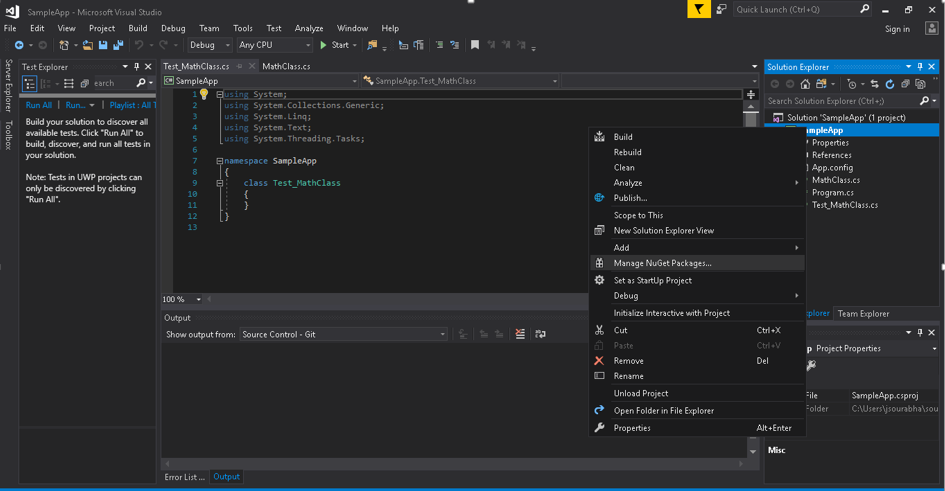
***Figure 5: mathematical operation methods***

Step3: Create a testfile for the MathClass.cs class in same project.



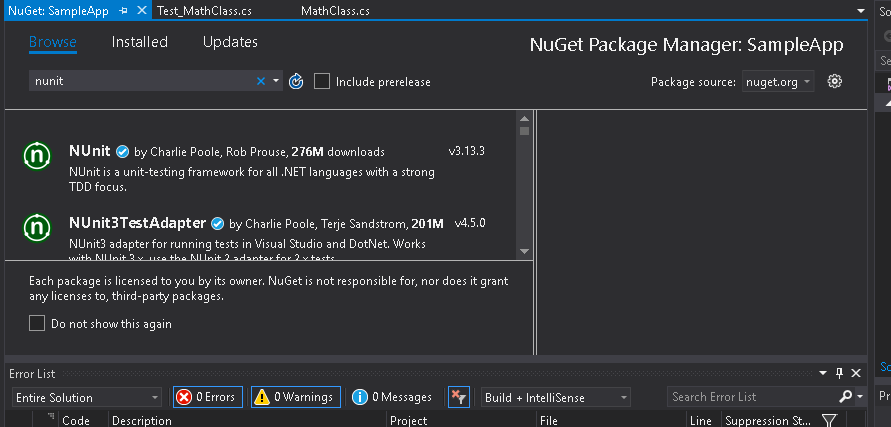
***Figure 6: new test class***

Step4: Add nunit framework to the project by right clicking on the project name and select “Manage NuGet Package”



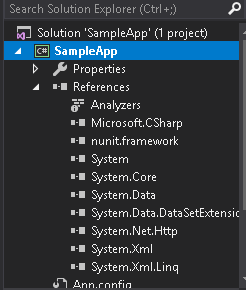
***Figure 7: Install packages***

Search nunit and install Nunit and Nunit3TestAdapter packages



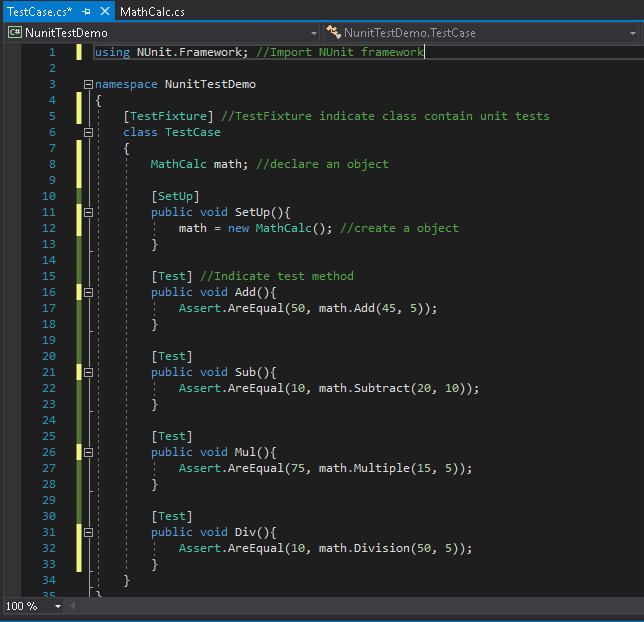
***Figure 8: Install nunit and test3 adapter***

Verify the installed framework from References folder in existing project. (nunit.framework)



***Figure 9: Installed references***

Step5: Open Test\_MathClass.cs test file and write test cases as shown in below figure

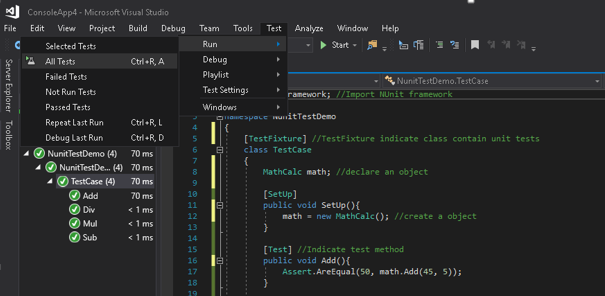


***Figure 10: Unit test class***

* The **setUp** method is the one which executes first of all in the class. So, the variables which are needed by all of the test cases should be placed here
* The [TestFixture] attribute denotes a class that contains unit tests. The [Test] attribute indicates a method is a test method.

Step6: Run the test case by selecting Test > Run > All Tests or Ctrl+R,A

Absorbed the pass or fail status



***Figure 11: Run test cases***

**HTML TEST REPORT GENERATION**

**Steps for Generating HTML Report:**

1. After you have finished your code about test, go build=>rebuild the solution. (No need to use test explorer)
2. When rebuild succeeds, find the .dll file of your unit test project and copy the path of this file. (In my test, it’s UnitTestProject3.dll)
3. Open the ‘developer command prompt for VS’. And then type “cd path” command to locate the file which contains the dll file. (The path is what you copy in step2)

“C:\Program Files (x86)\Microsoft Visual Studio\2017\Professional>cd C:\cocoro\_project\PC-Agent\SHPCAgent\_28072020\UpdatedCode\SHPCAgent\_28072020\SHPCAgent.Tests\bin\Debug”

1. After that, type “vstest.console.exe UnitTestProject3.dll /logger:trx”. Then you can see the test result in the window, also you can find the sentence like this “Results file: …path\name.trx”

“C:\cocoro\_project\PC-Agent\SHPCAgent\_28072020\UpdatedCode\SHPCAgent\_28072020\SHPCAgent.Tests\bin\Debug>vstest.console.exe SHPCAgent.Tests.dll /logger:trx”

1. That is the test result file you want, and you can convert it from xml to HTML by using third-party tools such like trxer. You can type “ReportUnit.exe TestResultFileName.trx” in cmd to convert it to HTML using trxer.

“C:\cocoro\_project\PC-Agent\SHPCAgent\_28072020\UpdatedCode\SHPCAgent\_28072020\APIConnector.Tests\bin\Debug\TestResults>ReportUnit.exe APIConnectorHTMLReport.trx”