Objectives

· Understand what is Code Coverage

· Explain about the benefits of Code Coverage

· Demonstrate Code Coverage using Fine Code Coverage

In this hands-on lab, you will learn how to:

· Use Fine Code Coverage in Visual Studio 2019 in order to perform Code Coverage

Prerequisites:

The following is required to complete this hands-on lab:

· Visual Studio 2019 IDE (Any edition)

Notes:

Estimated time to complete this lab is 60 minutes.

Task 1- Create a Sample Class Library Project

Steps to follow

1. Create an Empty solution in Visual Studio 2019 and name it as “CalculatorApp”.

2. Add a new Class Library project and name it as “CalcularorLib”

3. Rename the Class1.cs as Calculator.cs and verify the class name renamed from Class1 to Calculator.

4. Define the following method in the Calculator class

public int Add(int firstNum, int secondNum){…}

5. Implement the above method as per the following business rules.

Rules Condition

Rule 1 If both the inputs are >0 then the method should return the sum of given numbers.

Rule 2 If the first input is >0 and the second input is <=0 then the method should return -1

Rule 3 If the second input is >0 and the first input is <=0 then the method should return -2

Rule 4 If both the inputs are <=0 then the method should return 0

Task 2- Create a Unit Testing project using NUnit

Steps to follow

1. Add new Class Library project under the same solution, CalculatorApp

2. Name the project as “CalculatorLib.Tests”

3. Rename the file Class1.cs to “CalculatorTest.cs”. This will rename the class name to CalculatorTest as well. Verify it.

4. Add the reference of the following packages in the newly created project using NuGet Package Manager. Right Click the test project and choose Manage NuGet Packages… option from the context menu.

NUnit

NUnit3TestAdapter

5. Add TestFixture attribute for the SUT, and TestCase attribute for the methods.

Since there are 4 execution paths in the Add method, you need to write 4 test methods in order to cover it.

One is given for your reference. Write other 3 test cases by your own.

Rule 1 [TestCase] public void Add\_When\_Both\_Inputs\_GreaterThanZero\_Returns\_ExpectedResult() { //Arrange Calculator calculator = new Calculator(); int firstNo = 30; int secondNo = 20; int expectedResult = 50; int actualResult; //Act

actualResult = calculator.Add(firstNo, secondNo); //Assert Assert.That(expectedResult == actualResult, "Verify the add method logic"); }

Before adding other test cases, let’s try to find out how much code is being tested. For that, we will use Code Coverage.

Task 3 – Perform Code Coverage using Fine Code Coverage

Steps to follow

1. Go to Visual Studio menu called Extensions. Choose Manage Extensions menu item.

2. In the Manage Extensions dialog window, type Fine Code Coverage in the search text box. Choose the one which is shown below.

3. Download the plugin and Install it.

4. Launch the Fine Code Coverage UI using View->Other Windows->Fine Code Coverage from the Visual Studio IDE.

5. Fine Code Coverage window will be tabbed next to the Output window which normally appears in the Visual Studio status bar.

6. From the Test Explorer window, run the test method that you wrote for the Rule 1 scenario. Observe the test result as it is passed.

7. Now click the Code Coverage window to see the graphical representation of the test code covered during the test run. Statement/Line coverage is indicated using green/red color. Green indicates the covered code whereas red indicates the uncovered code. Go through the below screenshot to see the details.

8. Navigate to the Calculator class and see the green color in the margin of the code editor window which indicates the covered code as shown below.

Similarly, write the remaining 3 test cases so that you can achieve 100% code coverage while writing unit test code to test your application code.