XUnit and NUnit

Testing Framework

TABLE OF CONTENTS

Introduction 3

Essential Components of Unit Testing 3

What is XUnit? 3

Implementation 4

Step 1: Creating XUnit Project 4

Step 2: Creating functions to test 4

Step 3: XUnit Testing 5

Setting up Fluent Assertions 6

Unit Testing using Fluent assertions 7

What is NUnit? 9

Implementation 9

Step 1: Creating NUnit Project 9

Step 2: Creating functions to test 9

Step 3: NUnit Testing 10

Running the tests……………………………………………………………………………………………………………..……………………11

Comparison 12

Conclusion…………………………………………………………………………………………………………………………………………….12

Renjith R.T

# Introduction

We as developers miss the most obvious errors and some corner cases. Sometimes we know about them, but we tend to postpone things to solve later. And when it is time of deployment or doing a demo for a client, we see ugly errors getting prompted because we didn’t check certain corner cases.

That’s where testing comes in. Testing will help us use our time productively as it helps us track bugs quickly and testing also prevents us from writing bad codes.

In Test First Development (TFD) or Test Driven Development (TDD) approach the test cases will be written before actual development of the application. This ensures all edge cases are covered during the development process.

In this document we will learn how to test .Net application with XUnit and NUnit in Visual Studio.

# Essential Components of Unit testing

We use a concept called “AAA” in unit testing. We see that a unit test is a three-step process.

AAA -> Arrange, Act, Assert

Arrange:

First step of unit testing. Here we will make necessary arrangements for the testing to work. For E.g.: creating object of class which we are going to test.

Act:

In this step we will do the actual unit testing and store the result from the function we test.

Assert:

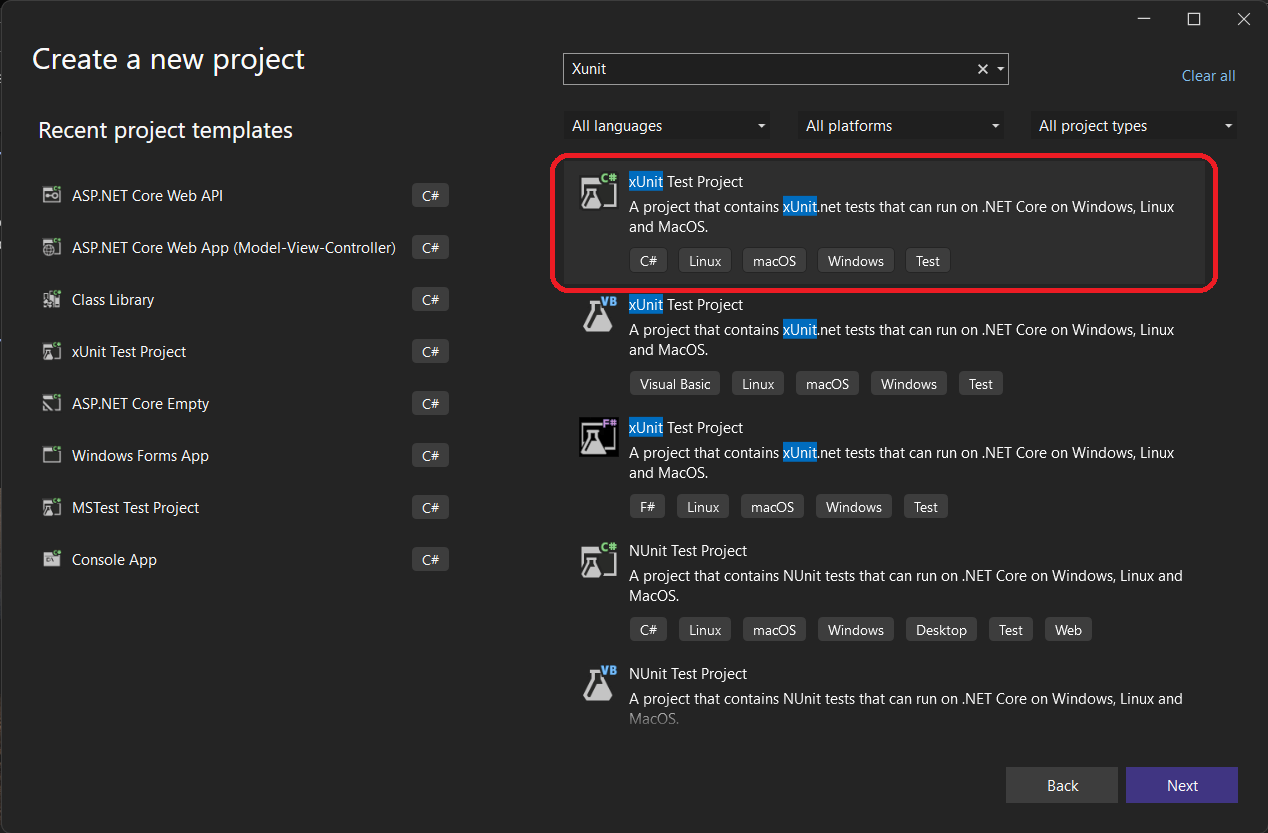
In the last step we will check and verify the stored result with expected results.

# What is XUnit?

XUnit.net is a free open-source unit testing tool for the .NET Framework licensed under Apache 2. Using XUnit we can create isolated test methods which means new instance will be created for each test method.

# Implementation

## Step 1: Creating XUnit project



Create an XUnit Test Project for C#.

## Step 2: Creating functions to test

Text

Description automatically generated

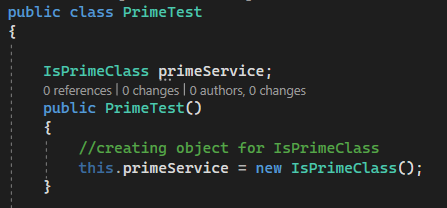
We have created IsPrimeClass class to define a function named IsPrime which returns whether a number is prime number or not.

* If number is less than or equal to 1, it is not a prime number
* From 2 to square root of the number, if the number is divisible. It is not a prime number
* Else it is a prime number

## Step 3: XUnit Testing

Now let’s test the IsPrime function we wrote.

Let’s do the arrange part in constructor.



Here we have created an object for the IsPrimeClass. So that we can use the IsPrime() function in every test method. The constructor part will run before each test method.

**Let’s check if the function returns False for non-prime number**

Text

Description automatically generated

* In the Act part we are calling the IsPrime() function with a value of 10.
* In the Assert part we are checking if the function returned false.

**Let’s check if the function returns True for prime number**

**Text

Description automatically generated**

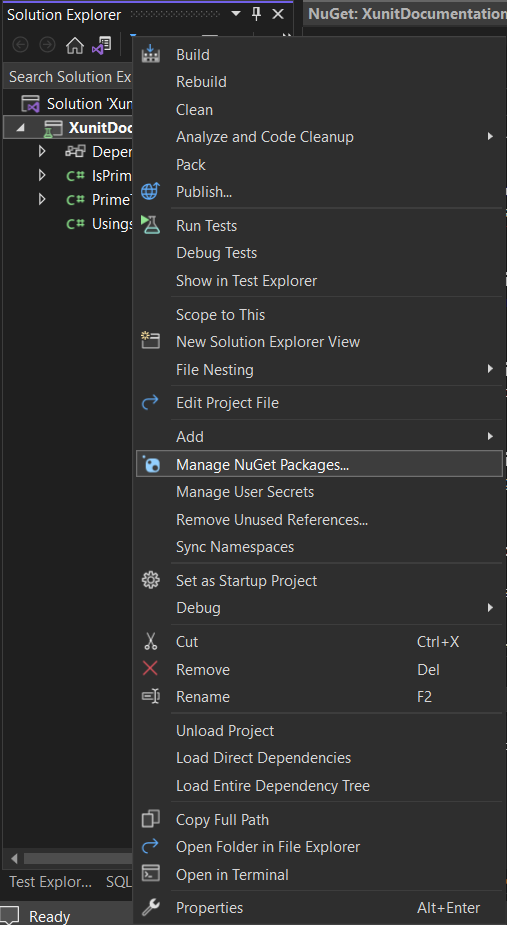
* In the Act part we are calling the IsPrime () with the value 5
* In the assert we are checking if the function returned true
* This test case won’t run, as we marked the Fact as Skip.

We must provide [Fact] before every test case. If not, the test runner won’t consider the function as a testing function.

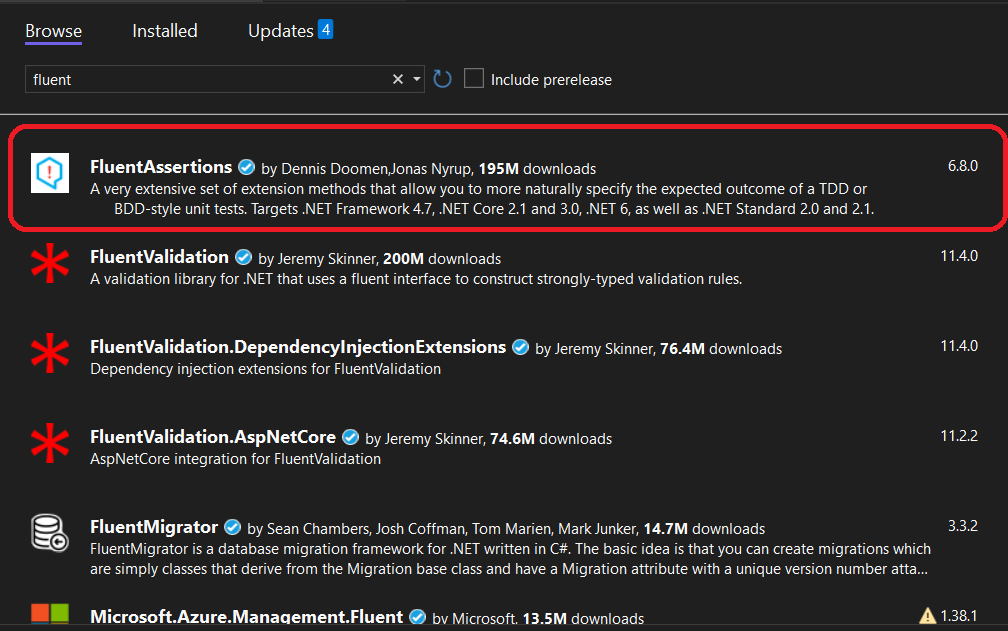
## Setting up Fluent Assertions

**Downloading Packages:**

Right click on the test project and select Manage NuGet Packages



Search for Fluent Assertions and install the package



## Unit Testing with Fluent Assertions

With Fluent Assertions we can check the results with more features like Should(), Be() etc.. There are a lot of external assertion packages we can explore and use.

**Checking if the function returns False for negative number**

**Text

Description automatically generated**

* In the Act part we are calling the IsPrime () with the value -1
* In the assert we are checking if the function returned false

**Checking if the function returns False for one**

**Text

Description automatically generated**

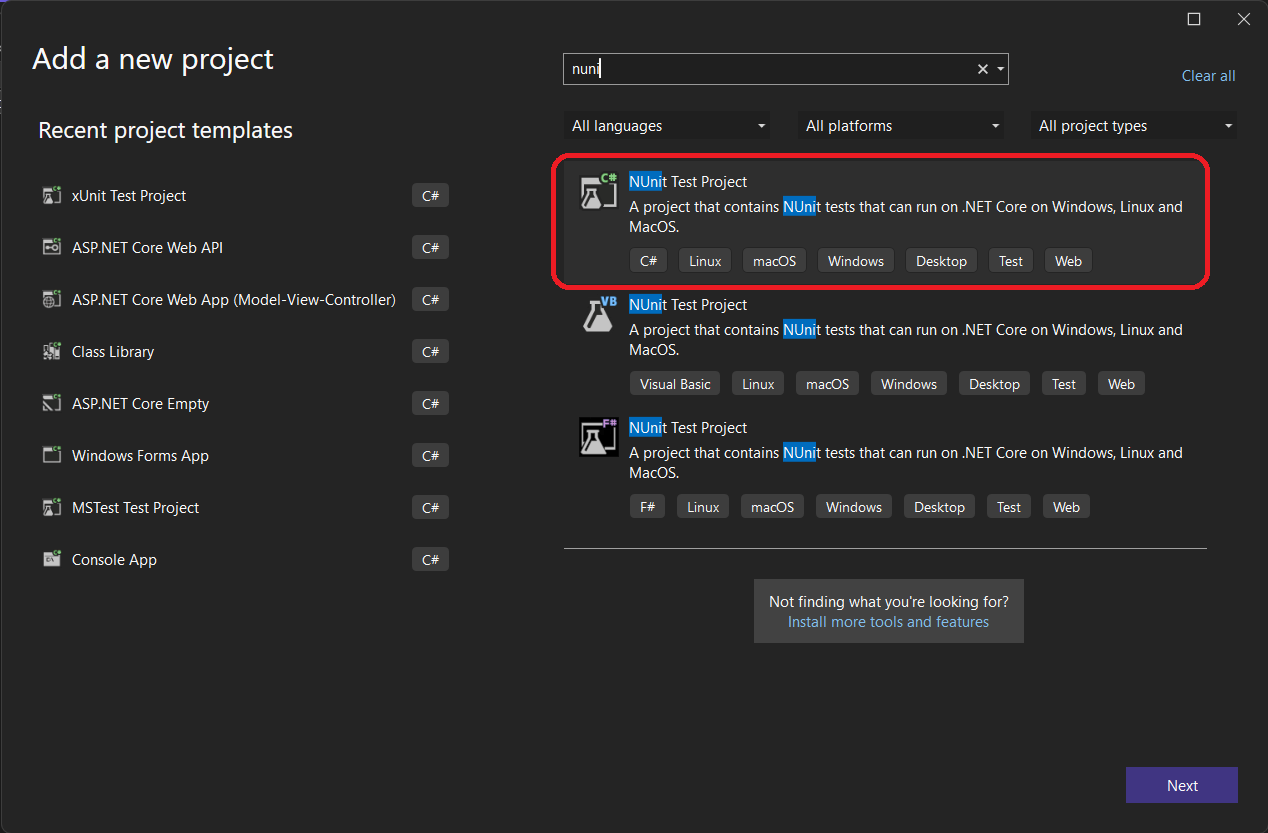
* In the Act part we are calling the IsPrime () with the value 1
* In the assert we are checking if the function returned false

# What is NUnit?

NUnit is an open-source unit testing framework for the .NET Framework and mono. It serves the same purpose as Junit for JAVA. With the knowledge we have in XUnit, we can easily get a grab on NUnit as both are almost the same.

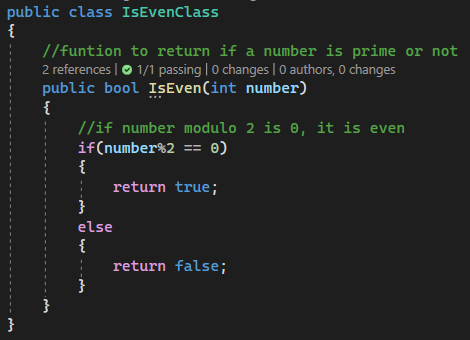
# Implementation

## Step 1: Creating NUnit project



Create an NUnit Test Project for C#.

## Step 2: Creating functions to test



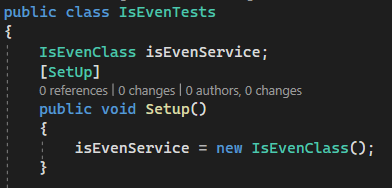
We have created IsEvenClass class to define a function named IsEven which returns whether a number is even or not.

* If the number modulo 2 is 0, It is even number
* Else it is odd number.

## Step 3: NUnit Testing

Now let’s test the IsEven function we wrote.

Unlike XUnit, we do the arrange part in Setup() function in NUnit.



Here we have created an object for the IsEvenClass. So that we can use the IsEven() function in every test method.

**Let’s check if the function returns True for even number**

**Text

Description automatically generated**

* In the Act part we are calling the IsEven() function with a value of 10
* In the Assert part we are checking if the function returns true.

**Let’s check if the function returns False for odd number**

**Text

Description automatically generated**

* In the Act part we are calling the IsEven() with the value 23
* In the assert part we are checking if the function returns false
* This test case won’t run, as we marked as Ignore.

We must provide [Test] before every test case. Unlike XUnit, here we use [Ignore(“reason”)] to skip a test.

In the test explorer, ignored test functions won’t even appear. But in XUnit skipped test cases will appear in the test explorer but won’t run.

## Running the tests

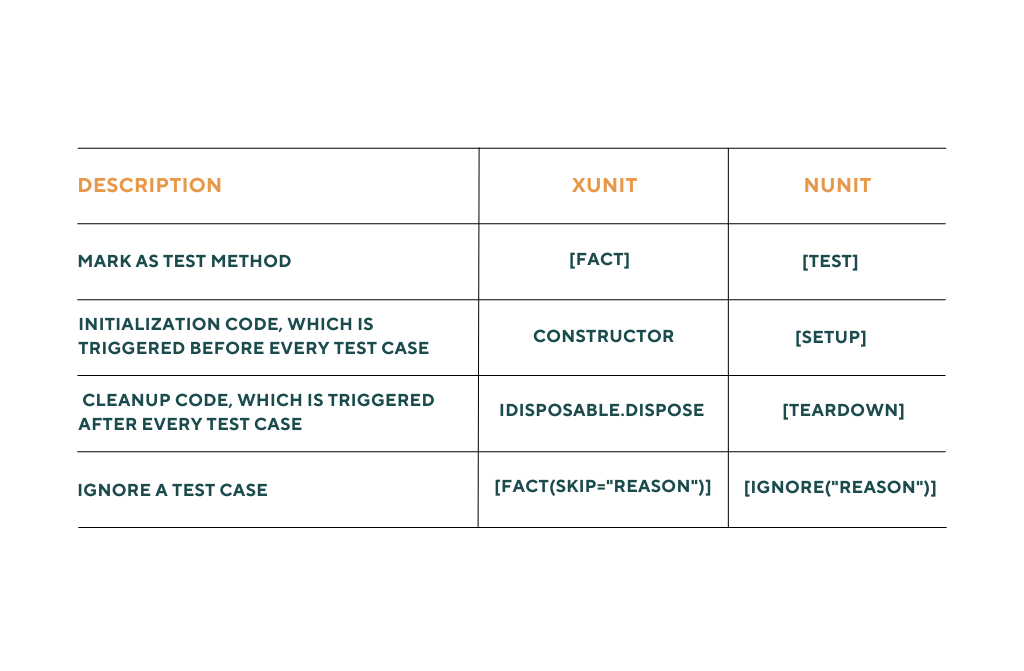
In the test explorer, we can run the tests.

Graphical user interface, text, application, chat or text message

Description automatically generated

We can see all the test cases has been passed. Thus, the functions work fine.

# COMPARISON



# Conclusion

In this document, we have seen how to setup a XUnit and NUnit test project. And saw about AAA fundamental in unit testing. We learnt how to add new packages to our project. Then we wrote some test cases in XUnit and NUnit and with Fluent Assertions.