SDLC :

1. Problem identification
2. System Analyse **Set some objectives here for System**
3. Design
4. Testing : Test / Check whatever was set in SRS , are we getting those or not
5. Implementation
6. Maintenance

Background Verification >

BGV

Testing : Whatever was specified in SRS (Software Requirement Specification)

Whatever objectives or requirements we have set , are we able to achieve them or not

Testing > Static / Dynamic

Static Testing > We are testing code without running it

How : Code Review, You can go through your code

Dynamic Testing > We test the code by running code

Testing cud be done at different levels

Unit Testing : We test single unit , Developers themselves do this testing

Integration Testing : Here we can combine some functions and will test after integrating they are working properly or not , development team will do testing

System Testing : When you test system as a whole , development team

User acceptance Testing : We test complete system , end users

Black Box Testing : When tester does not know anything about the code he is testing

White Box testing : When the tester is aware about the internal statements

Regression Testing

Load testing

Steps

1. Create Class Library

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TestingLibrary

{

public class NumericFunctions

{

public int AddNumbers(int x, int y)

{

return x + y;

}

}

}

Step 2 : Create one more class library that will contain test cases, so this is our test project

In this , we need to install NUnit , NUnit3 TestAdapater

Add Reference to project that you want to test

using NUnit.Framework;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TestingLibrary.Tests

{

[TestFixture]

public class NumericFunctionsTests

{

[Test]

public void AddNumbers()

{

// Testing happens in 3 steps

// AAA

// 1. A - Arrange (Set environment for testing)

// 2. A - Act (Call method)

// 3. A - Assert (Check whether test is passed or failed)

// Arrange

TestingLibrary.NumericFunctions numericFunctions = new TestingLibrary.NumericFunctions();

// Act

int result = numericFunctions.AddNumbers(10, 30);

// Assert

Assert.AreEqual(40, result);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LibraryToTest

{

public class NumericFunctions

{

public int AddNumbers(int x, int y)

{

return x + y;

}

public string GetDiscount(int Age)

{

if (Age > 60)

return "Discount is 60%";

else if (Age >= 40 && Age <= 60)

return "Discount is 40%";

else

return "No Discount";

}

}

}

using NUnit.Framework;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LibraryToTest.Tests

{

[TestFixture]

public class NumericFunctionsTest

{

[Test]

public void AddNumbersTest()

{

// AAA

// 1. A : Arrange , create environment for testing

// 2. A: Act, call the method

// 3. A: Assert , check whether test is passed or failed

// Arrange

LibraryToTest.NumericFunctions numericFunctions = new LibraryToTest.NumericFunctions();

// Act

int result = numericFunctions.AddNumbers(10, 30);

// Assert

Assert.AreEqual(40, result);

}

[TestCase (10,20,30)]

[TestCase(20,40,60)]

[TestCase(30, 40, 70)]

public void AddNumbersTest(int x, int y, int expectedOutput)

{

// AAA

// 1. A : Arrange , create environment for testing

// 2. A: Act, call the method

// 3. A: Assert , check whether test is passed or failed

// Arrange

LibraryToTest.NumericFunctions numericFunctions = new LibraryToTest.NumericFunctions();

// Act

int result = numericFunctions.AddNumbers(x, y);

// Assert

Assert.AreEqual(expectedOutput, result);

}

[Test]

public void GetDisountMoreThan60()

{

LibraryToTest.NumericFunctions numericFunctions = new NumericFunctions();

string message = numericFunctions.GetDiscount(80);

Assert.AreEqual("Discount is 60%", message);

}

[Test]

public void GetDisountEquals40()

{

LibraryToTest.NumericFunctions numericFunctions = new NumericFunctions();

string message = numericFunctions.GetDiscount(40);

Assert.AreEqual("Discount is 40%", message);

}

[Test]

public void GetDisountMoreThan40()

{

LibraryToTest.NumericFunctions numericFunctions = new NumericFunctions();

string message = numericFunctions.GetDiscount(50);

Assert.AreEqual("Discount is 40%", message);

}

[Test]

public void GetDisountLessThan40()

{

LibraryToTest.NumericFunctions numericFunctions = new NumericFunctions();

string message = numericFunctions.GetDiscount(30);

Assert.AreEqual("No Discount", message);

}

[TestCase(30,"No Discount")]

[TestCase(40, "Discount is 40%")]

[TestCase(70, "Discount is 60%")]

public void GetDisountTest(int x, string expectedOutput)

{

LibraryToTest.NumericFunctions numericFunctions = new NumericFunctions();

Assert.AreEqual(expectedOutput, numericFunctions.GetDiscount(x));

}

}

}