Background Verification > BGV

SDLC > System Dev Life Cycle

1. Problem Identification
2. Problem Analysis (SRS)
3. Design
4. Coding
5. Testing
6. Implementation
7. Maintenance

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);

}

}

}