**Name: Deepanshu Rathore**

**Superset ID: 6358199**  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Week 2  
1. NUnit-Handson  
  
Hands-on.  
  
Unit testing means verifying the behaviour of the smallest pieces of code in isolation. For example, when you write a method that adds two numbers, a unit test will call that method with various inputs and confirm that it returns the correct result every time. These tests focus on single classes or methods and use mock implementations to stand in for any external components such as databases or web services. Functional testing treats the application as a black box and exercises entire features or workflows from end to end, simulating how a user interacts with the system or how different modules work together.

There are many kinds of testing beyond unit and functional. Automated testing refers to any set of tests that can be run without human intervention, whether they are unit tests, integration tests, or even performance scripts. Performance testing measures how the system behaves under high load by generating large numbers of requests. Integration testing verifies that different parts of the application communicate and work together correctly. As you grow more familiar with these approaches, you might explore security testing, usability testing, or acceptance testing to cover other quality aspects.

Automated tests provide fast feedback on every code change, catching errors early and preventing regressions. This rapid feedback loop lets you find and fix bugs before they reach production. Automated tests also serve as living documentation. When someone new joins the project they can read the tests to understand exactly how each part of the system is supposed to behave.

Designing code that is loosely coupled and testable means writing classes that depend on abstractions rather than concrete implementations. For example, if your code relies on an interface rather than a specific class, you can easily swap in a mock or stub when you run tests. This separation makes it simple to test each component in isolation and also makes your system more flexible and easier to maintain over time.

When you write your first test program in NUnit you will use attributes such as TestFixture and Test to identify your test classes and test methods. You will use SetUp to create any objects you need before each test and TearDown to clean up after each test finishes. If there is a test you want to skip temporarily you can mark it with Ignore so that the test runner reports it as skipped rather than failed.

Parameterized test cases allow you to run the same test logic with different inputs and expected outcomes without duplicating code. Instead of writing separate methods to test adding two plus three and negative one plus five you write one method and supply each scenario as a TestCase attribute. The test runner then invokes your method once for each scenario and verifies the actual result against the expected value. This approach keeps your test code clear concise and easy to extend as you add more cases.  
  
**Code:**

using NUnit.Framework;

using CalcLibrary;

namespace CalcLibrary.Tests

{

[TestFixture]

public class CalculatorTests

{

private SimpleCalculator calc;

[SetUp]

public void SetUp()

{  
 calc = new SimpleCalculator();

}

[TearDown]

public void TearDown()

{

calc = null;

}

[Test]

[TestCase( 2.0, 3.0, 5.0)]

[TestCase(-1.5, 1.5, 0.0)]

[TestCase( 0.0, 0.0, 0.0)]

public void Addition\_AddsTwoDoubles\_ReturnsCorrectSum(double a, double b, double expected)

{

double result = calc.Addition(a, b);

Assert.That(result, Is.EqualTo(expected));

}

[Test]

[Ignore("Example of Ignore attribute")]

public void ThisTestIsSkipped()

{

Assert.Fail("Ignored");

}

[Test]

public void Division\_ByZero\_ThrowsArgumentException()

{

Assert.That(() => calc.Division(5, 0),

Throws.ArgumentException

.With.Message.Contain("Can't be Zero"));

}

}

}

**Output:**A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.