Unit Testing

Unit tests are typically automated tests written and run by software developers to ensure that a section of an application (known as the "unit") meets its design and behaves as intended.. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, or an individual method.. By writing tests first for the smallest testable units, then the compound behaviors between those, one can build up comprehensive tests for complex applications..

To isolate issues that may arise, each test case should be tested independently. Substitutes such as method stubs, mock objects,. fakes, and test harnesses can be used to assist testing a module in isolation.

During development, a software developer may code criteria, or results that are known to be good, into the test to verify the unit's correctness. During test case execution, frameworks log tests that fail any criterion and report them in a summary. For this, the most commonly used approach is test - function - expected value.

Writing and maintaining unit tests can be made faster by using parameterized tests. These allow the execution of one test multiple times with different input sets, thus reducing test code duplication. Unlike traditional unit tests, which are usually closed methods and test invariant conditions, parameterized tests take any set of parameters. Parameterized tests are supported by TestNG, JUnit and its .Net counterpart, XUnit. Suitable parameters for the unit tests may be supplied manually or in some cases are automatically generated by the test framework. In recent years support was added for writing more powerful (unit) tests, leveraging the concept of theories, test cases that execute the same steps, but using test data generated at runtime, unlike regular parameterized tests that use the same execution steps with input sets that are pre-defined