Disciplines - Testing

# General

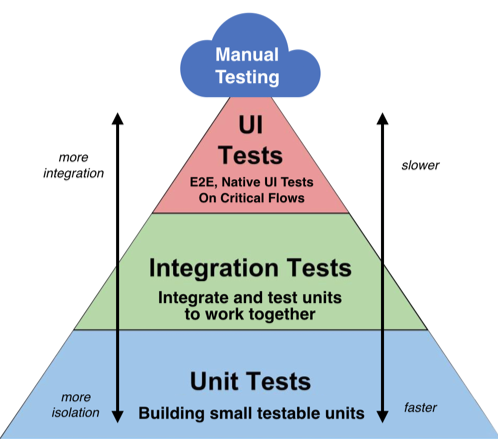
## What Is Testing

Testing code is common practice to make robust systems. It is typically automated and explicit to allow for strong confidence in the state of the system during deployment and changes to the code base.

Tests provide assurance of that the system is functioning as required and designed. Additionaly, testing provides important documentation of expected inputs, outputs, and of behaviour and system.

## Types

Generally, a systems full suite of tests follows the triangle rule where there are few expensive end to end tests and many inexpensive unit tests:



* Manual - Tests performed by user groups
* End to End (E2E) - Tests the whole system behaves correctly when used in the way it will be deployed. E2E tests should hit the real backend and use test credentials for external services. Due to expense of the tests the coverage will be less than full, generally completing only valid user flows.
* UI - Tests the full user interface, generally with a mocked backend. Coverage should be complete, including all invalid user flows and actions.
* Integration - Since unit tests mock all external services, integration tests are required to test that all units of code integrate correctly.
* Unit - Tests individual units of code, with a test per piece of functionality. Tests are short and mock any external influence with the expected behaviour from them.

# Best Practice

## Unit Tests

Tests should be grouped by method, described by method name.

Test names should explain the context of the test and let the test code explain the expected behaviour.

<https://www.betterspecs.org/>

## Testing Cases

There are several different types of cases for tests:

* Input check - check the system only accepts and outputs the correct type of value
* Edge case - occurs at the extremes of a system (maximum and minimum values)
* Corner case - occurs when two or more variables are simulatinously at extreme levels (maximum or minimum values)
* Specical cases - non-obvious non-boundary special cases, such as special inputs like minimum allowable floating point numbers etc

To ensure an effective and well documenting suite of tests, the test suite should cover:

* Input checks - to define accepted inputs and output response
* Edge cases - to define the boundary of the system
* Corner cases - if they cover additional behaviour not expected from standard edge cases
* Special cases - cover behaviour with special values which cause unexpected behaviour