Acceptance Testing

1. Test the IDEA:

Ideally the product idea is tested before any time and money is invested in code.

This can be done through:

1. market research
2. prototyping
3. mockups
4. surveys
5. conversations with humans.

2. Test the code design (unit tests):

We use TDD to experiment, learn, refine code and design.

These tests morphs into the unit tests that is used for regression testing, providing us a safety net when we change code.

3. Test the programming interfaces:

We also call this integration testing

We can use the same tools and frameworks as for TDD (e.g. JUnit),

or perhaps even different tools

(e.g. Postman) to test code on an API level

(i.e. public interfaces, or web service/HTTP API endpoints).

This typically entails that different modules in the service must be running, e.g. the server must run so we can do tests against it.

4. Test the human interfaces:

This involves manual or automated testing directly on the UI layer.

Kathalon/Appium/Vysor/Selenium

Repeatable tests must at the minimum be:

1. Scripted

to tackle

Use cases

Scenarios

Be easy to do manually

2. Ideally

be runnable by an automated tool

Kathalon

5. Acceptance testing:

Here we test correct functionality as expected by the clients, users or other stakeholders.

Acceptance testing is typically a combination of integration tests and UI tests, and we can use testing frameworks like JUnit here as well.

Integration tests + UI tests = Acceptance tests

{JUnit}

In this iteration, we will focus on acceptance tests.

6. Test the outcomes:

The most difficult is testing the actual impact our product is having

Is it having an impact on jobs to be done for users?

For this we measure and analyse different usage metrics over time, and create feedback channels for users.

Codescene