September 6, 2020

#Python | #pytest | #blog

Types of tests

Other pytest articles:

Why testing is important

Types of tests

Test driven Development

Hello, World!

Selecting tests with pytest

Testing HTTP client with pytest

Testing database with pytest

Advanced fixtures with pytest

Pytest plugins

There are many types of tests. Brian Marick came up with this chart, which is widely used to show which types you should care about in order to deliver a high-quality application.

In this diagram, he categorized tests according to whether they are business-facing or technology-facing, and whether they support the development process or are used to critique the project.

Copy

```
                    Business facing
                         ^
                         |
 *Automated*             |    *Manual*
 Functional acceptance   |    Showcases, usability
 tests                   |    tests, exploration
                         |
<-support-----------------------------critique->
 programming             |              project
                         |
 *Automated*             |   *Manual/Automated*
 Unit test               |   Non functional
 Integration tests       |   acceptance tests,
 System tests            |   (capacity, security)
                         V
                   Tech facings
```

Business-Facing Tests That Support the Development Process

These tests are usually called functional or acceptance tests. Acceptance tests should be written before development starts. They can test all kinds of attributes of the system being built. In an Agile environment acceptance tests answer two important questions, “How do I know when I am done?” for developers and “Did I get what I wanted?” for users. User stories or requirements can be said to be done when their respective acceptance tests pass.

There three different paths in a user story: The happy path is a single canonical path through the application in terms of user actions. There should always be a test for a happy path

In complex systems, there could be variations in the initial state, the actions to be performed, and the final state of the application. Sometimes, these variations form distinct use cases that are referred to as an alternate path.

The sad path is when some variation in the initial state should cause errors and exceptions

Technology-Facing Tests That Support the Development Process

These automated tests are written and maintained exclusively by developers. There are three types in this category: unit tests, component tests, and deployment tests.

Unit tests test a particular piece of the code in isolation. They often rely on simulating other parts of the system using mocks. Unit tests should not involve calling the database, using the filesystem, talking to external systems, or, in general, the interaction between components of a system. And because of that, they run very fast so you can get early feedback on whether changes have broken any existing functionality. These tests should also cover virtually every path in the system (called code coverage).

But isolating code won't allow us to stop bugs that result from an interaction between different pieces of the application. Component tests (also known as "integration tests") test larger clusters of functionality so that they can catch problems like these. They are slower since they can perform more input-output operations like talking to databases

Deployment tests check that your application is correctly installed, correctly configured, able to contact any services it requires and that it is responding.

Business-Facing Tests That Critique the Project

These manual tests verify that the application will in fact deliver to the users the value they are expecting. An important form of these tests are showcases. Agile teams do showcases at the end of every iteration to demonstrate the new functionality. It's also wise to demonstrate functionality to customers as often as possible to avoid misunderstanding of specifications and requirements.

Finally, you can give your application to real users using beta testing or A/B testing programs. Many companies (Booking.com, Netflix) continually release new features to selected users without them even noticing. These organizations gather statistics on how the new feature is used, and get rid of it if it doesn't deliver sufficient value.

Technology-Facing Tests That Critique the Project

There are two kinds of acceptance testing: functional tests and nonfunctional tests. Nonfunctional means all the qualities of a system other than its functionality, such as capacity, availability, security, and so forth. The

tests used to check whether these criteria have been met, and the tools used to run the tests are usually quite different from those used functional tests.

These tests often require considerable resources and special environments to run on and specialized knowledge to set up and implement

Regression testing

Regression testing is particularly important. It's a crosscutting category so they are not mentioned in the chart. Regression tests represent the entire set of your automated tests. They ensure that when you make a change you don't break existing functionality. They also enable you to easily refactor code because they verify that you haven't changed any behavior when refactoring is done.
The Test Pyramid The simplified view on test organization that is widely used is a test pyramid. Mike Cohn came up with this concept in his book Succeeding with Agile. It's a great representation telling you to think about different types of testing. It also tells you how much testing to do on each layer. more

Copy

```
integration   _________       slower
^            /         \         ^
|           /_ui tests__\        |
|          /             \       |
|         /_service tests_\      |
|        /                 \     |
V       /____unit tests_____\    V
more                          faster
isolation
```

It seems overly simplistic, but due to its simplicity the essence of the test pyramid serves as a good rule of thumb when it comes to creating your own test suite. Some of the essentials are:

- Write tests with different granularity
- The more high-level you get the fewer tests you should have
- Stick to the pyramid shape to come up with a healthy, fast and maintainable test suite: - Write lots of small and fast unit tests. Write some more coarse-grained tests and very few high-level tests that test your application from end to end.

In this unit you've learned what types of tests exist. In the next one you'll get acquainted with most popular testing methodology - TDD.

Similar articles:

- Advanced fixtures with pytest
- Hello, World!
- Pytest plugins
- Selecting tests with pytest
- Test driven Development