Exploratory Testing Guide

# 1. Context

What is exploratory testing?

The aim of exploratory testing is to identify and address issues with a feature. Acceptance Criteria (defined in Stories) never covers every combination of; input, state, behavior, dependency, downstream, environment, visual aspect etc.

What is an issue?

Anything that does not honour the spirit or the letter of the Acceptance Criteria, or introduces a regression.

When should Developers do exploratory testing?

Generally when Stories are stuck in the testing phase of the Sprint, the team should help to unblock the Story. There are two caveats:

1. The Dev or Dev pair should have some understanding of what is involved in order to complete the testing.
2. The underlying cause for the block should be looked at. The block might be avoided with; more refinement, smaller stories, more test automation etc

# 2. Collaboration with QA

Before starting exploratory testing, the Dev and QA should collaborate to check that the Story is in a state to be tested, so that the Dev can move on to another Story.

The Dev and QA will check the following:

* The Acceptance Criteria in **Jira** have been met
* Sufficient coverage in Integration/BDD tests including logging. ***If there is missing coverage, and the proposal is to test manually, please resist and encourage automation.***
* Fallback considerations
* Perf and security test coverage

If there are issues with the handover please ask the Dev to address and leave the ticket at ‘In Dev’.

# 3. Exploratory Testing

If the handover has been successful, please check the following where relevant:

* Happy and sad paths for ACs
* Any documented manual regression (nb. should be very few)
* Validate the test automation
* Different approaches; boundary analysis, model based test
* Break limits eg. enter large string, html, images (see SecList)
* Time sensitive queries
* Downstream / end-to-end tests
* Visual look and feel
* Mobile devices
* Cross browsers and resolution
* Known gotchas [Gotchas link](https://theculturetrip.atlassian.net/wiki/spaces/QAS/pages/861667373/Gotchas)
* Penetration test opportunities (OWASP top ten, eg. sql injection xss + mobile)
* Performance feel
* Brittle area focus eg. complex algorithms, redirection loops
* Fallback and recovery
* Config correctness
* Network conditions
* Logging

Testing that we repeat every Story should be automated.

# 

# 

# 3. Issue Flow

If you find an issue, please follow the flow below, to ensure good collaboration:

[Issues Flow](https://docs.google.com/presentation/d/1wTI2H68HL2iAmhmxnBCsjUz5lpuPu-5yCt96t0LbXzA/edit#slide=id.p)

# 4. Raising a Bug

Depending upon the Squad approach, raise a subtask on the ticket or a separate ticket linked to the Story should be created.

Assign it to the Dev or the PM.

Add the following detail to the ticket:

* Version of the app
* Environment
* Recreation steps
* Screenshots or a recording
* Log file output

If it is a candidate for automation please remind the Dev to add an Integration/BDD test.

# 5. Tools

* NightwatchJS / Selenium for integration f-e tests
* Postman for calling APIs <https://www.getpostman.com/>
* Firebug/Inspect for browser analysis
* Wireshark for network hacking <https://www.wireshark.org/download.html>, ChromeThrottling
* JSON validator <https://jsonlint.com/>
* HTML validator <https://validator.w3.org/>
* SEO calculator <https://sitechecker.pro/>
* Web performance measure <https://www.webpagetest.org/>
* Screen recorder [Screenshot tool](https://chrome.google.com/webstore/detail/awesome-screenshot-screen/nlipoenfbbikpbjkfpfillcgkoblgpmj?hl=en), [Recording tool](https://chrome.google.com/webstore/detail/loom-video-recorder-scree/liecbddmkiiihnedobmlmillhodjkdmb/related), QuickTime