Basics

How do you deal with flaky tests?

When flaky tests occur, they need to be verified that they are not false positives. A way to this is to replicate the issue by inputting the same data, and under the same test condition to achieve the reported result. Repeating the tests also gives more useful information on what conditions might be responsible for the failures.
Once the tests are repeatable, one can now investigate further to find out the root cause.
A way of investigating this is to isolate the test such that it is not dependent on another test, and only one result is expected from it. If the root cause could not be discovered on time, or there are more priority tests to treat, the flaky tests should be quarantined, and then fixed at a later time so that the integrity of the results are not affected.
When the root cause is found and the flaky test fixed, the test need to be rerun several times to ensure that they are no longer flaky before they are put back into the suite.

Let's suppose there is a test pipeline taking about 1 hour to finish, what would you do to decrease the time of it?

- The time can be reduced by running multiple tests in parallel.
- Refactoring can also be done to speed up the test execution. For example, the order of the tests could be changed such that services or browsers do not need to be restarted for each test case.
- Running test on headless browsers also help.
- Avoid using implicit waits, instead, use explicit waits
- Avoid having a single test for many functionalities. In order words, the tests should be atomic and in parallel
- Using selector that are known to be faster in an implementation. For example, finding an ID by a CSSSelector is faster than finding a name by XPath in Selenium.

Imagine you have the possibility to ask software engineers to develop tools for you that will increase your productivity as full-stack QA, please describe to them your requirements

The tool that will increase productivity of a full-stack QA the most will be a test automation tool, and the requirements for such will be:

- The first thing is that tool has to be capable to achieving the desired result, in this case it has to be able to reduce the execution time of tests, have capabilities to run tests in parallel and scale to large data sets
- It should be written in a language that the team is already familiar with

- Documentation should be available or easily sourced
- The tool should be easy to use and maintain so that any team member with little technical knowledge can make use of it
- It is important to have meaningful results and reports that can be analyzed and presented to the internal stakeholders
- Cross-browser functional and be able to run on different environments is also important
- The cost, amount of time, effort and other resources used in developing the tool should not outweigh the benefits

Test Case Challenge

Test Objective

The test objective of this exercise is to verify that the login feature of the app functions as intended under normal business environment.

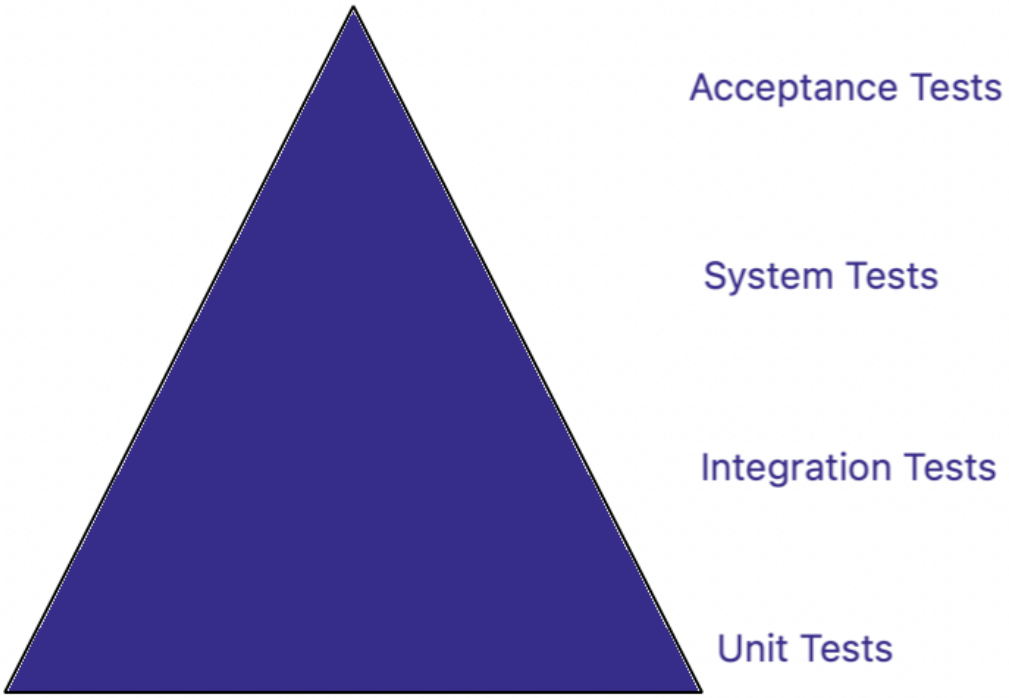
In Scope

Login module
Mobile platforms
Desktop browsers
API integration

Not in Scope

Other modules

Test Levels



Unit Tests

Each component or function of the login flow is tested to catch errors and ensure refactoring is easily achieved

Integration

The integration of the each component with another component is tested to ensure communication and exchange of data is accurate.

System Testing

The login flow of the application is tested on all mobile and desktop environments

Acceptance

This is done to ensure that the user interface & industry requirements are met. The critical flow of the user interface will be **automated** and executed as regression tests.

Environments

Web

Cross-browser tests will be executed on the latest versions of the following web browsers:

- Chrome
- Firefox
- Safari
- Edge

Mobile

- Sanity test will be carried out on the minimum supported and latest OS versions
- It is also important to check analytics tracking to know the most used devices, Android and iOS versions by the app customers, and also test the app on them
- Other mobile related scenarios should be tested such as: app Installation & uninstallation, updates, interrupts by phone calls & notifications, poor network simulations, screen locks, etc
- Performance test like battery consumptions, memory leakages and hardware requirements also need to be tested
- Security and privacy checks should be prioritized such as ensuring the only necessary permissions are requested and granted
- Screen: Smaller screen sizes to tablets sizes, as well as switching orientation between portrait and landscape modes

Clusters

Simulate network requests from locations handled by each cluster. This enables QA to experience what users in different markets see while using the app.

Localization and Languages

Test translations to ensure that texts, figures, symbols, currency, sizing, legal requirements, etc are displayed in selected language and location.

Deployment

- Tests will be made on Staging
- Issues reported as ticket comments
- Fixes will be made according to the problems that the team finds

- Once all tests pass and only minor issues are open, this ticket will be deployed to Production
- Decision maker will the Product Owner
- Smoke tests will be performed on production

Non Functional Tests

- Response times should be acceptable limits
- Ensure authentication and authorization are enforced
- Only one request from a registered client should be processed at once.

Test Scenarios

Functional Tests

S/N	Description	Expected Result
1	Verify that a user will be able to login with a valid email and valid password	Login is successful and user is taken to main screen of application
2	Verify that a user cannot login with a email and an invalid password	Login is not successful. Error message is displayed telling user that an invalid email or password was entered. User should remain in login page.
3	Verify that a user cannot login with unregistered email and an valid password	Login is not successful. Error message is displayed telling user that an invalid email or password was entered. User should remain in login page.
4	Verify that a user cannot login with unregistered email and an invalid password	Login is not successful. Error message is displayed telling user that an invalid email or password was entered. User should remain in login page.

5	Verify that user can not submit form if email field is blank	Email field is highlighted Error message is shown on field asking user to enter valid email
6	Verify that user can not submit form if email field is invalid	Email field is highlighted Error message is shown on field asking user to enter valid email
7	Verify that user can not submit form if password field is blank	Password field is highlighted Error message is shown on field asking user to enter password
8	Verify that user can not submit form if password field is less than permitted character length	Password field is highlighted Error message is shown on field asking user to enter valid password

Non Functional Tests

Security Tests

S/N	Description	Expected Result
1	Verify that system does not reveal if email is registered when random email is entered	Error message is displayed telling user that an invalid email or password was entered.
2	Verify that password field is not in plain text	Password field should be in asterisk
3	Verify that user can no longer login with old password if password was changed successfully	Error message is displayed telling user that an invalid email or password was entered.
4	Verify that user is logged out if session expiry time is reached	User will be redirected to login page
5	Verify that user can not use the back button to access a secured page when logged out	User will be redirected to login page

6	Verify that user can not access the app on a new window with the same browser with more than 1 user account	The same user will be logged in the new window

Performance Test

- Response times of requests sent to the server should not exceed set threshold
- Very that the service can process an expected amount of load under business requirements
- Test how many request might break the service
- Verify all links on application pages return expected status code

API Test

- Verify get requests return 200 status code
- Verify secured page returned unauthorized error if no valid authentication is in the request
- Verify payload is required for POST log in request as expected

Automated Tests

- Only tests that have been manually verified will be automated
- Tests that are repeatable, stable, and which might iterate over some data are great candidates to be automated
- The automated tests will be performed at the user acceptance test level, and to be run as regression tests

Automation Test Challenge

Repository can be found in https://github.com/KunleOduwobi/MyTheresa

- Ensure Node.js is installed
- Clone the repository and navigate to the project root directory
- To use the Cypress component to run test:
    - Run: npx cypress open --component
    - Select MyTheresa-main
    - Select E2E Testing
    - Pick a browser, and click the Start button
    - Select any test in the spec folder

- If you want to run all the test cases in the terminal, run:
    - npx cypress run

Please note that this command runs test cases 1 & 2 completely. Test case 3 needs to be executed from the Cypress component

- More command options are in the read me