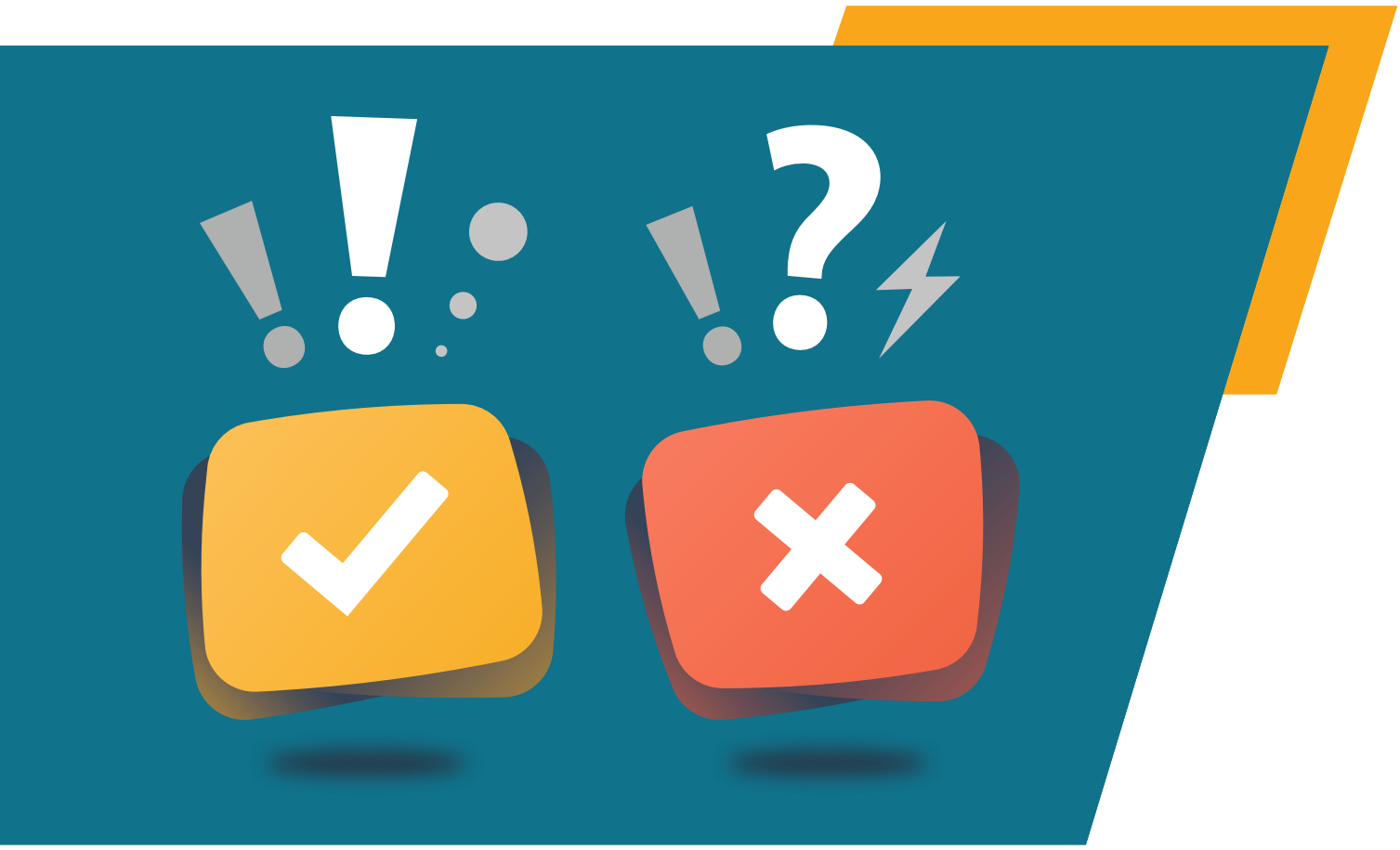


Dealing with False Positives and False Negatives in Mobile App Test Automation

INDEX

INTRODUCTION

In the digital space, speed and accuracy are two non-negotiable pillars. A crack in any one of these pillars puts the app in danger and can even be fatal. Where apps are concerned and more so in the mobile app world, users are spoilt for choices! Hence every app has to ensure superior user experience and keep abreast of the rapid digital innovations that are constantly striving to cater to ever-changing user preferences. The 'Go-to-Market' race is also foremost in the agenda of every mobile app developer and tester – be it a new app or an existing app's frequent updates. In this scenario, mobile app development and testing have to heavily depend on automation, as the development and testing demands are far beyond the best of human capabilities. Automation in any area has tremendous benefits but also comes with problems of its own, and automated mobile app testing is no exception.

One of the major concerns in automated mobile app testing is the false or erroneous results that are sometimes generated. These erroneous results could be False Positives or False Negatives. A False Positive is a test result that indicates that there is an error when in reality there is none. A False Negative is a test result that indicates that the software is fine, although in reality there are bugs and glitches. Both these spell bad news for mobile app testers, with False Negatives being the greater evil.

This whitepaper seeks to address this vexing problem faced by mobile app testers and will explore ways in which to contain or reduce the possibilities of both False Positives and False Negatives.

ABSTRACT

In the area of mobile apps, automated tests are meant to detect errors and bugs, and ensure that the app is thoroughly verified. Automation is an absolute necessity in today's digital world but unfortunately automated testing comes with its share of problems that surface as False Positives and False Negatives.

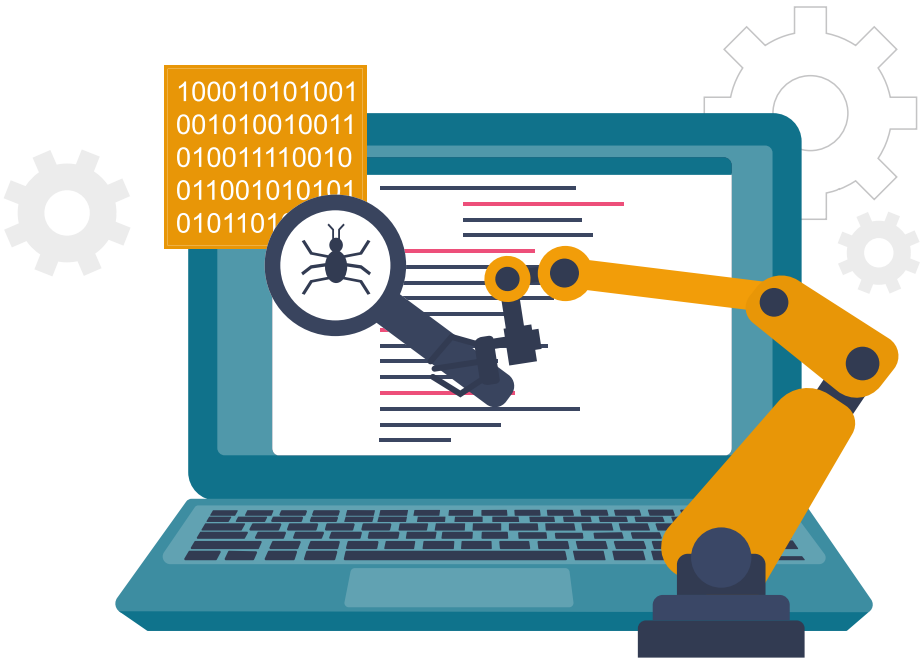
In mobile app testing, a positive result is meant to indicate the existence of a bug or error in the software and a negative result implies that the test detected no bug, which in turn conveys that the mobile app is working fine. However, at times the automated test outcomes can be False Positives or False Negatives. False Positives result in a waste of precious time, chasing errors that don't exist. A False Negative on the other hand, is more dangerous, as it clears the software of the mobile app for deployment even though there are errors in it.

This whitepaper will throw light on this worrisome issue by reviewing it in two parts.

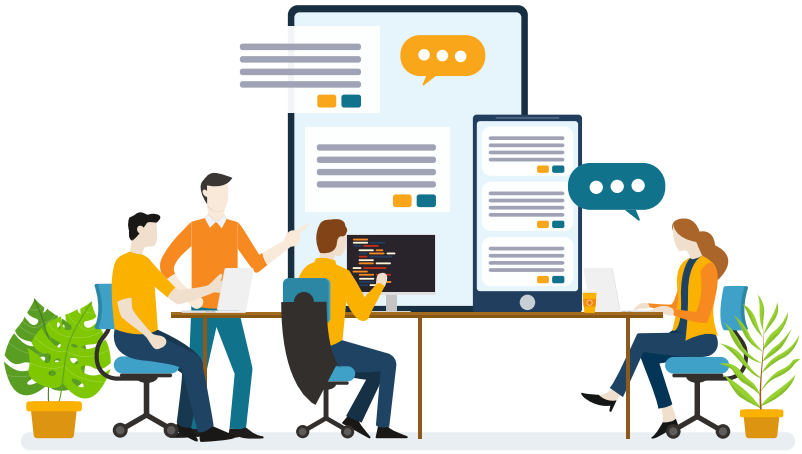
The first part is titled **Understanding False Positives and False Negatives in Mobile App Test Automation** and will explore what False Positives and False Negatives are, why they occur, and the implication of these false errors for mobile app testers.

While false automated test errors can be quite unnerving, abandoning automation is definitely not an option, given the fast paced digital environment. Hence it is important to work towards avoiding or at least reducing these false results. This is the purpose of the second part of this whitepaper which is titled **Avoiding False Positives and False Negatives in Mobile App Test Automation.**

It is hoped that what follows will help the reader reduce erroneous test results and generate a higher degree of confidence in automated mobile app testing.



UNDERSTANDING FALSE POSITIVES AND FALSE NEGATIVES IN MOBILE APP TEST AUTOMATION



The first step in addressing any issue is to understand it in all its aspects.

Automated mobile app tests are blindly trusted and hence the results are taken at face value and necessary actions follow. It would naturally be considered unnecessary and a waste of time to check whether automated results are right or not. This is neither expected nor desired of testers, given the severe time constraints. However, the fact is that automated tests do give false results at times and these are classified as False Positives and False Negatives. This section seeks to help the reader understand what False Positives and False Negatives are; why they occur; and their implications.

Let's begin with the lesser of the two evils – False Positives. As mentioned earlier, a False Positive in test automation is a test result which indicates that there is a problem in the mobile app software, although there is none. These are test events that fail despite there being no defect in the mobile app under consideration. In fact, the test itself may be the problem, but it gives an impression that the app is defective. Some of the reasons for these occurrences are as follows:

- Change in code or product functionality; automation approach; written automation script; or implemented framework.
- Changes in identifiers which are the ways in which the tool connects to a specific UI element. E.g. If the submit button is inserted inside three HTML division markers, then inside a pre tag, then the 2nd division marker, and the structure of the page changes, then the code will no longer be able to find that element and it will show up as a False Positive.
- Changes in information required in the mobile app. E.g. If earlier, country field was mandatory and in the new form it is removed as it is no longer relevant, then on automation clicking submit without filling in the new form, it will show up as an error although it's not so in the new scenario.

- Changes in workflow. E.g. Changes in a tab or button, or introduction of a new page on a wizard.• No appropriate waiting may be actualized for a question before the test interfaces with it.
- The test data may be incorrect i.e. the test may be defined for a data field that does not exist in the mobile app. E.g. the app may not have a field for middle name, but the test has been defined to ensure that first name, middle name and last name are mandatorily displayed.

On the face of it False Positives seem harmless, because they are not defects in reality. But the fact is that precious time and money are wasted by sending testers on a wild goose chase in trying to get to the root cause and fix errors that do not exist. The cause of the False Positive is an inadequately composed test. However, since automation is expected to give right results, testers are bound to dig deeper and identify the root cause of the error. Unfortunately they end up uselessly analyzing the mobile app software, and their efforts come to naught. Besides unproductive time and money spent, False Positives can also be very frustrating for testers as the build procedure and deployment are unnecessarily slowed down, which in turn extends the go-to-market time.

False Negatives are test executions that show that there are no defects, although the mobile app has bugs. This can be extremely dangerous as it generates false confidence in the app. The mobile app software will move through successive stages in the SDLC, happily presuming all is well because the automated test says so. This could badly backfire when the bug is discovered late in the SDLC or worse still when the mobile app has been launched in the market.

Some of the reasons for False Negatives or False Pass rates are as follows:

- Non-coverage of a feature.
- Difficulty in setting up a scenario and therefore ignoring it.
- Lack of co-ordination between mobile app testers.
- Arrival of the automation tool after the codebase resulting in tests being created for the changes and not for the original functionality.
- Missing Assertions in the test i.e. the parameters needed to verify that the features give the right results were never created.
- The initial state of the database is incorrect.
- Issues related to test environment setting, the browser, or the network.

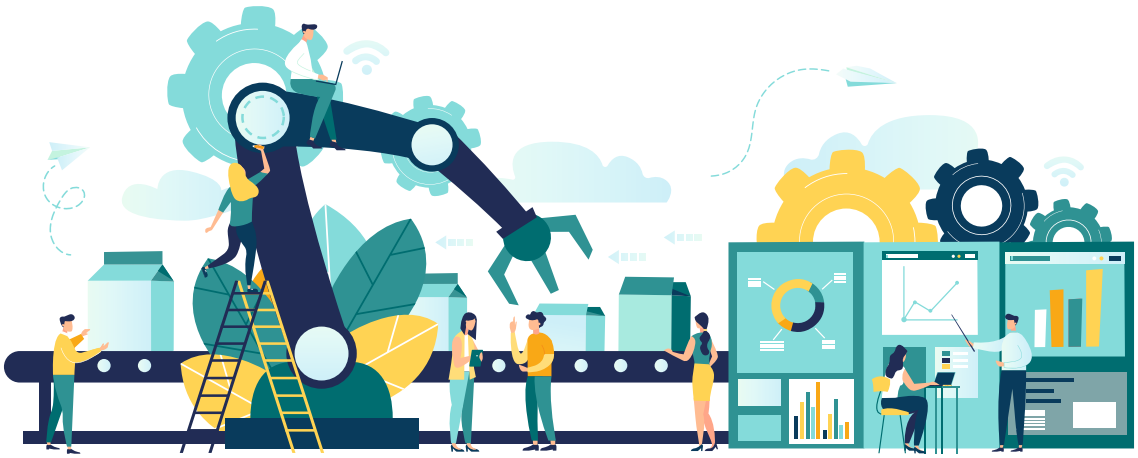
When the number of false errors is high, confidence in automated results gets shaken and there may be a temptation to ignore these errors or even abandon automated testing. But neither of these is a good option. Let's take a simple analogy that is universally fresh in the minds of everyone, to understand the need for going ahead with mobile app test automation, despite the false errors that show up.

The recent pandemic brought increased amount of testing, albeit in the medical field. Tests and testing kits were hurriedly put in place; vaccines were feverishly explored and invented in laboratories across the world. The approach to contain the pandemic was two-pronged: Widespread Testing to isolate the infected and slow down the spread of the virus; Widespread Vaccination to build immunities to fight the virus. There was a tearing need for testing kits and vaccines to contain the pandemic, but there were also commercial reasons to get to the market before competitors and make up the millions of dollars spent in medical research!

Were all the results true and correct? – Definitely not. Were all the vaccines safe and 100% effective? Again the answer is a resounding NO! Did that stop the testing and vaccination drive? NO – again! Was the drive effective? The answer this time is thankfully YES, because two years down the line the world is bouncing back to normalcy.

Something similar happens in the mobile app testing world. Automation is mandatory to cope with the speed that drives the digital world. True, automation does generate some False Positives and False Negatives – these are its uninvited by-products. But then discarding automation would amount to throwing the baby out with the bathwater! After all manual testing is not an option in this fast paced digital world, no matter how efficient and meticulous the manual tester may be. Drawing from the analogy – Discarding automation would be like discarding the Covid testing and vaccination drive because there were some false results and adverse vaccine effects. But that would have meant letting the virus spread and wreak exponentially more havoc than the false results and side effects!

While striving for perfection in the digital world is important, there is also a practical need to weigh options vis-à-vis the pros and cons; make decisions based on the path of least impairment; and simultaneously work towards refining systems.



AVOIDING FALSE POSITIVES AND FALSE NEGATIVES IN MOBILE APP TEST AUTOMATION

The previous section has clearly established the need for embracing automation in mobile app testing despite the possibility of false results. However, in the digital world there's no room for complacency and hence every effort must be made to avoid or at least reduce false results. This section will provide some guidelines on how to avoid False Positives and False Negatives in mobile app testing.

01 Decide the Type of Tests to be Automated

While majority of the tests will be automated, it is important to avoid automation for areas that are unstable or have frequent UI changes. It is also advisable to avoid scenarios that are not supported by the automation tool being used. Automation should be avoided in areas which have been identified to have performance issues and also areas which cannot be identified using unique locators.

02 Ensure that the Automated Test is Well Written

A review of the structure of the test script, workflow, and tear down fixture should be done, to ensure that the script contains only the relevant strips and verifications points, so that test cases have a 1:1 mapping. Best coding practices, commenting and naming conventions should be followed and hardcoding of data should be avoided.

03 Set up a Controlled Testing Environment

To reduce False Positives in mobile app testing, it is important to ensure that the environment is controlled and is only accessible by automated test cases. This avoids constant changes in data which prevent reproducing problems detected by the tests.

04 Set up a Controlled Testing Environment

Highly customized Frameworks and Libraries lack the advantage of widespread usage. Tried and tested mobile app testing systems have lesser chances of unidentified errors as their widespread usage speeds up error identification and rectification.

05 Use Dynamic Synchronization of Objects

Dynamic Synchronization is the process of waiting for a specific amount of time for the target element to become available for the automation event. It helps in improving the time of execution by ensuring wait time only when the element is not available. Dynamic waits are superior to static waits as they ensure that the wait is not so long that tests become unnecessarily lengthy; nor so short that rendering isn't finished. Insufficient wait time can generate false results.

06 Check Mobile App Test Data Setup

Test data setup can change the basic scenario and lead to false positives e.g. If the setup is defined for use by a person with Admin rights, but a guest logs in, it will break the test flow resulting in false positives.

07 Move the Setup Methods out of the Test Methods

Segregating these methods makes the test suites more stable by permitting uniformity in the setup method. Moving the setup to a common place increases reusability of the code. Problems with the set up like launching a browser, passing desired capabilities, setting resolution, and other such issues will then need to be fixed only in one place. This saves time in case of false results.

08 Ensure Frequent Peer Code Reviews

Peer code reviews can make the script robust and help identify or reduce false positives in mobile app testing. The review should be done by a person with good understanding of the product and its underlying functional changes; with good working knowledge of the language in which the framework and scripts are developed, so that the script can be modified with functional knowledge.

09 Keep Test Assertions and Coverage Concurrent

Mobile app tests should always be relevant to the existing program, since redundant coverage results in false positives.

10 Use Unique Identifiers

In case of Identifiers, it is advisable that every button and textbox has a unique identifier or named element in HTML, in order to reduce false positives in mobile app testing. Using accessibility hooks, such as alt tags, has the advantage of increasing accessibility.

11 Apply Caution when Internationalizing HTML IDs

When internationalizing HTML IDs, the ID name changes as per the region's communication language. Though users are not negatively impacted, it can be a reason for mobile app test automation to throw up false results, if abundant caution is not exercised.

12 Minimize Logic in the Code

Complex logic in the code increases the chances of things going wrong. Hence it is preferable to keep the automated mobile app tests simple, more so because the test code is untested.

13 Practice Mutation Testing

Mutation testing consists of changing the code, inserting an intentional bug or error in the mobile app software and then running the test responsible for detecting the bug. If the test passes it implies that the result is a false negative. It is of course not practical to prepare every error, compile it, deploy it, and verify whether the test finds the fault. But mutation testing can be achieved by varying the data of the test or altering different aspects or parameters. Practicing mutation testing before committing the code for a feature or a bug fix can go a long way in reducing false negatives.

14 Ensure Triggers for Companion Tests

It is also suggested that any change in source code should be checked to ensure that it automatically triggers a review of the companion test cases. This will help prevent false negatives due to refactoring.

15 Audit the Failures

It is very important that false positives and false negatives in mobile app testing are not just addressed and forgotten. Getting to the root cause and maintaining a record for future review is vital. If the frequency and intensity of these false errors are high, it is advisable to review test automation – perhaps a test or two a day until all assertions are cleaned up.

If after best efforts false results do occur, here are some pointers to find out why things went wrong:

- Check whether the test data was off-base.
- Check whether an element's functionality changed.
- Check if there was a change in the functionality of the code.
- Analyze whether the necessities were doubtful or whether they changed.
- Separate the test cases to see where things went wrong.

CONCLUSION

Automation is an inevitable part of the digital world and consequently of mobile app testing too. While the benefits of automation are immense, there is no denying that results could sometimes be false. False Positives can be a costly drain on time and resources and hence can be very frustrating for mobile app testers who are already struggling with limited testing time. But False Negatives are what testers fear the most, because it undermines their capabilities and shakes their confidence in automated mobile app testing.

Despite these risks, testers are left with no choice but to go ahead with test automation. Hence utmost care needs to be exercised to avoid False Positives and False Negatives. These false results should not just be dealt with for resolving the current problem, but in order to have a robust mobile app testing program, it is imperative to record, review, analyze and audit the false results from time to time. If need be, a test a day should be reviewed until all assertions are cleaned up. This will go a long way in reducing the future stress of testers and building confidence in the automated mobile app test results. Opting for a well-established automated framework that is meticulously tried and tested will greatly help avoid False Positives and False Negatives in app testing.

For Mobile App Testing, BOTm is a good error free automated framework that keeps abreast with the latest in technology. It provides mobile app testing for the entire spectrum under one umbrella. Visit **www.botmtesting.com** and sign up for a Free Trial to experience stress-free mobile app testing.