Getting Started with

MOBILE APPLICATION TESTING

Perfecto
Seek Perfection

perfectomobile.com

Table of Contents

Introduction

You're thinking of building a mobile app or perhaps you've already built one; depending on the approach you've taken, it could be a web app[1], a native app[2], or both.

If you have a single developer working on the project, it's possible, even likely, that each feature of the app was tested thoroughly during the development process. As more developers are added to a project, or as apps get more complex, ad-hoc testing performed by developers during development just isn't enough.

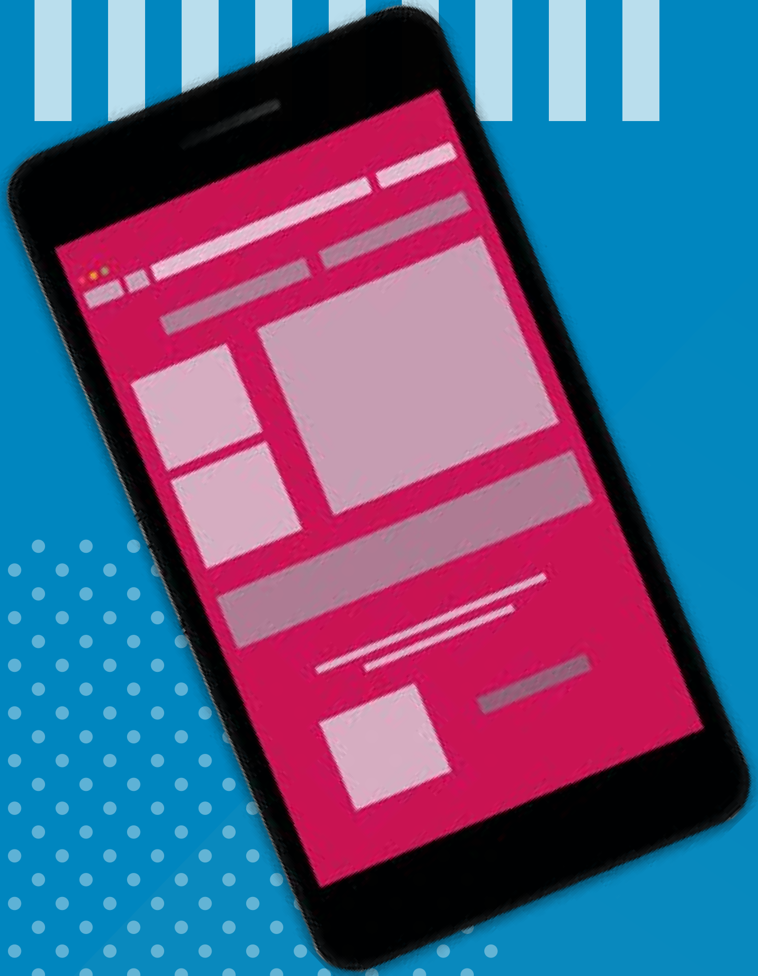
In a perfect world, you'd code your app once and have it work everywhere. Unfortunately, that world doesn't exist. By their very nature, web applications should run on any browser, but all browsers are not created equal. Vendor-specific implementations of web features can break some applications—no matter how well they're coded to standards.

Mobile devices complicate testing further. An app that runs on one device might not run as expected on another, even if both devices are running the same operating system (OS) or OS version. As with desktop browsers, mobile browsers are implemented differently across mobile devices or mobile device OS, so there's no guarantee that what runs on one device will work the same on another.

So what should you do and where should you start?

This guide should help you get started with mobile testing basics and some mobile app testing tips as well. We hope you find it useful!

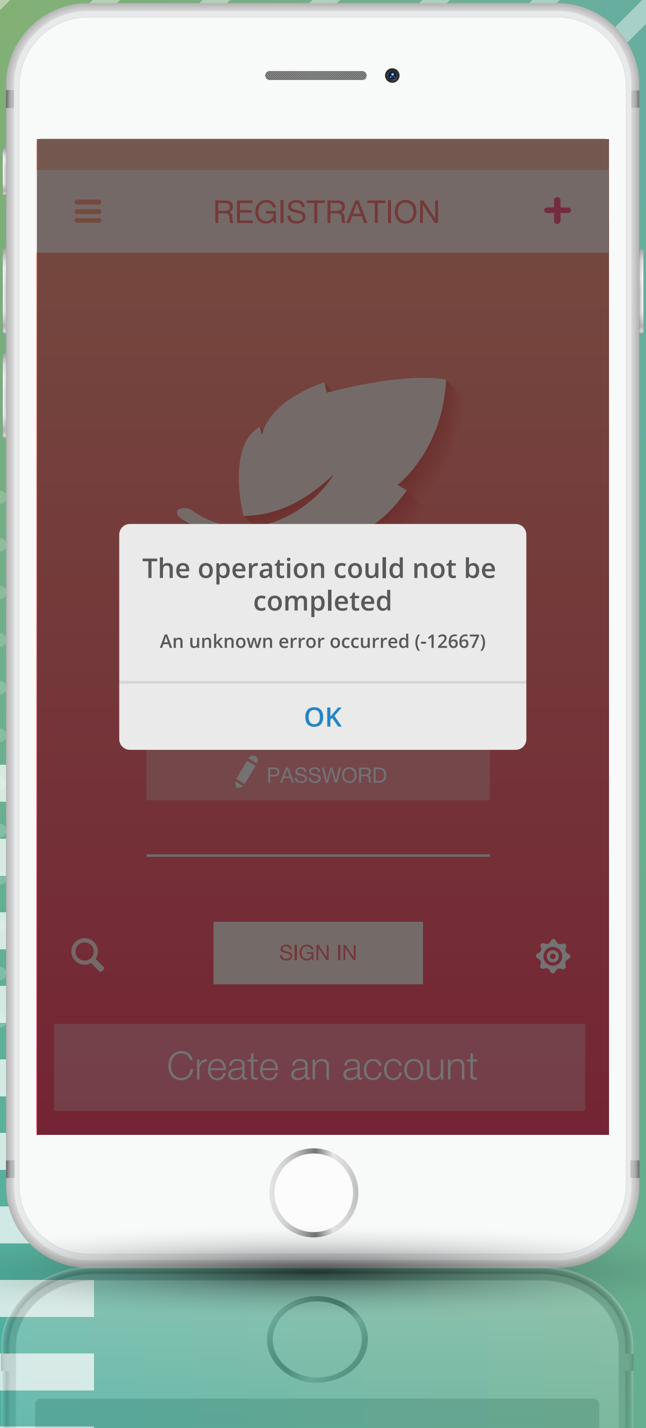
```
            <li><a class="scroll" href="#contac
                Contact           </a></li>
            <div class="clear"></div>
        </ul>
    </nav>
    <div class="logo">
        <a href="./">
            <img src="web/images/logo3.png" id="
```



6 Steps to Getting Started with Mobile Testing:

1. Culture changes and new tools
2. Testing the app with every code check-in
3. Testing the app under real world conditions
4. Testing apps on real devices
5. Testing the app on the right mix of mobile devices and desktop browsers
6. Implementing automation

BEFORE WE JUMP INTO EACH OF THESE TOPICS, LET'S REVIEW WHY TESTING IS SO CRITICAL TO YOUR BUSINESS.


REGISTRATION
The operation could not be completed
An unknown error occurred (-12667)
OK
PASSWORD
SIGN IN
Create an account

WHY TESTING MATTERS

Developers are fallible. Even with best intentions, they make mistakes, usually without being aware of it. Development organizations thoroughly test their apps in order to increase the likelihood they will find those mistakes before users do. As an app's complexity increases or the number of developers on a project increases, you'll need a more formal testing process and a dedicated set of tools to use. The best mobile app testing tip we can offer is, test every part of your app in every release. The sections that follow highlight initiatives driving testing's importance.

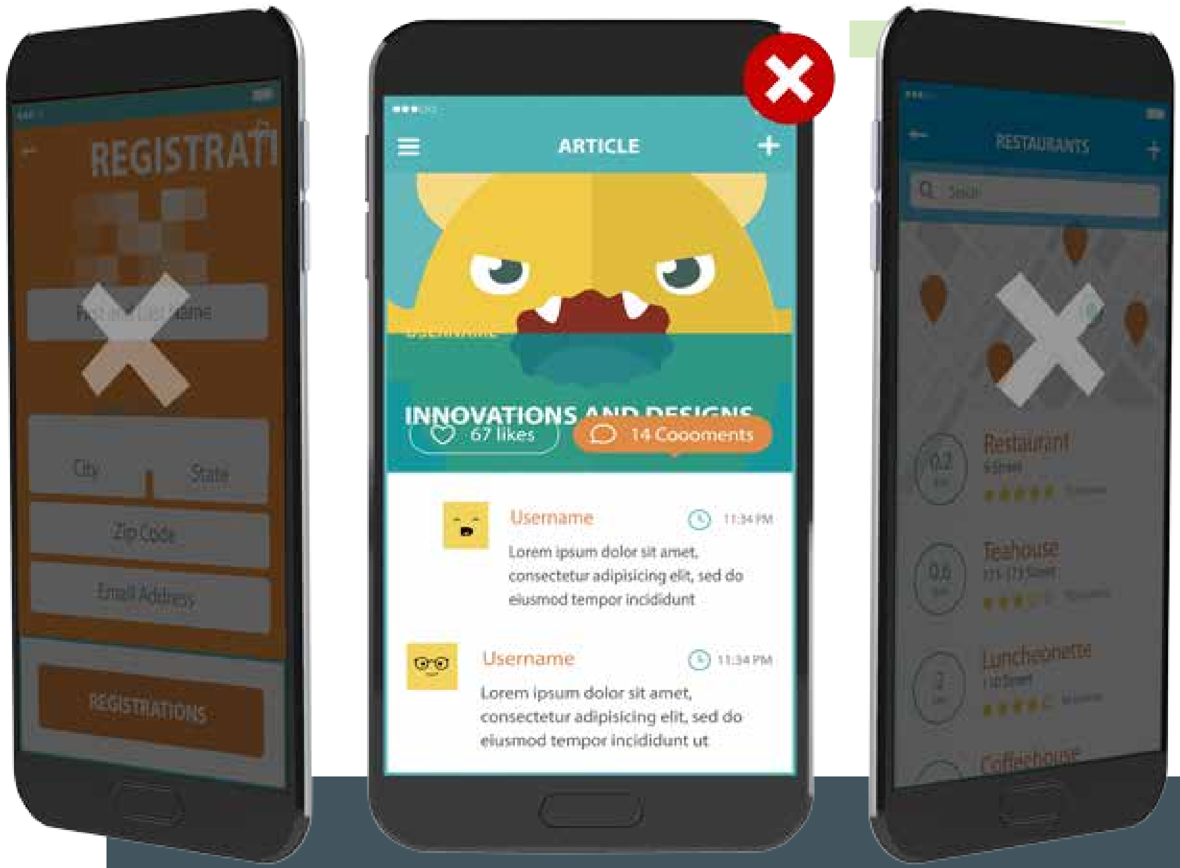
Why Testing Matters

Delivering on App Quality

Mobile testing basics dictate that the only way to ensure the app you're delivering is going to make your users happy is to test it thoroughly, test it often and test it in multiple ways. Simply testing that an app works in the way you want it to (called functional testing) isn't enough, and can prove to be detrimental for consumer-facing organizations in two major ways.

First, the first users who install your app and quickly discover that it's terrible, will simply delete the app or write an app store review to let the world know how bad your app is. Subsequent users interested in installing your app will see bad reviews on the app store before beginning the installation and decide to not bother.

Even if you've enticed a user to install your app, it must deliver continued value and be easy to use for users to want to come back to it later.



Mobile apps with two stars are downloaded 50,000 times, while those with four-star reviews are downloaded 270,000 times.

—Sept 2015 Apptentive survey

Why Testing Matters

Protecting Users from Harm

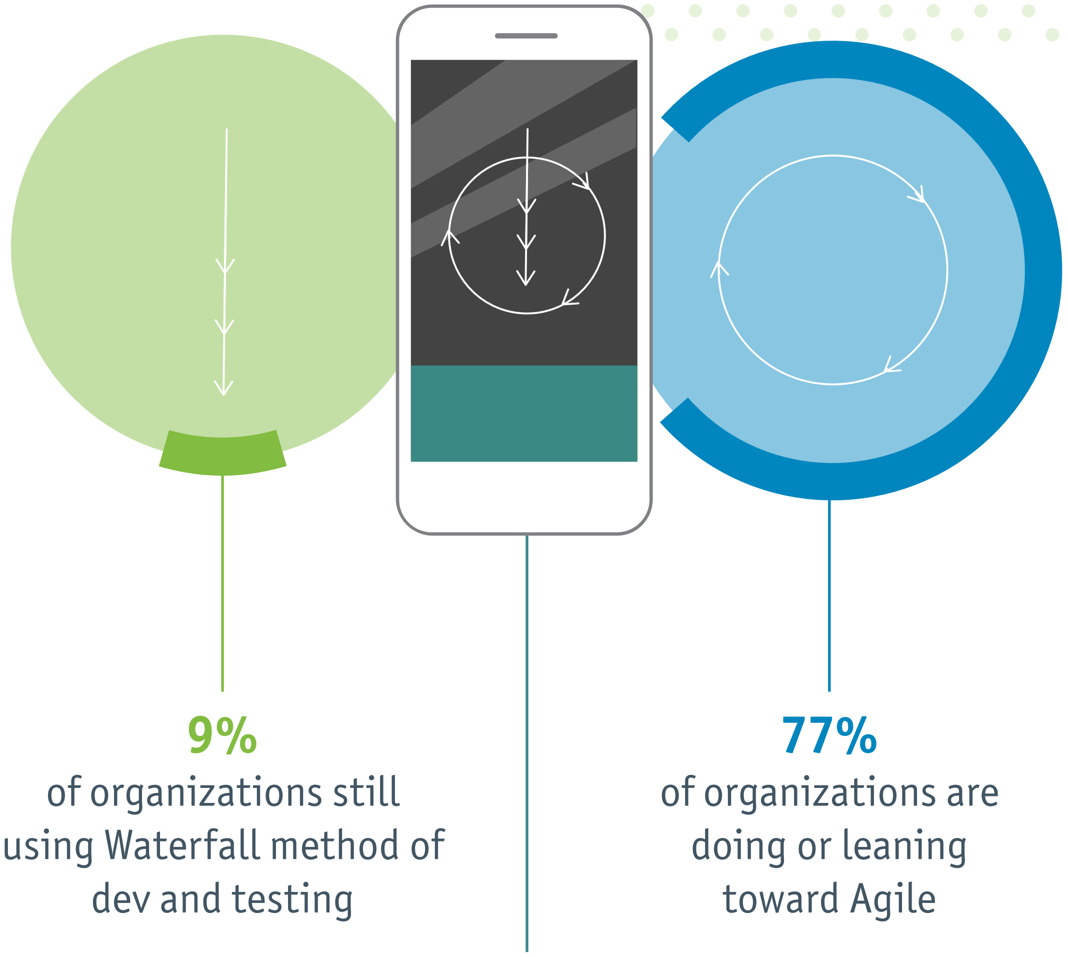
If your app is processing transactions on behalf of the user or if the app is storing or managing the user's personal data, then that code must be rock solid. The only way to ensure the quality of that potentially risky code is to set up a set of complete test scenarios and validate them with every release.

Code security matters here as well. The development team should work closely with the company's security and risk team to implement measures to validate the security of delivered code, assess the risks added by any included third-party or open source libraries and scan the application for any vulnerabilities. More details can be found by learning about the **OWASP Mobile Security Project**.

Protecting the Value of Your Brand

Nothing tarnishes a brand more than negative publicity. Whether it's low app store ratings, a security breach that makes headlines or the inadvertent use of a third-party library with known vulnerabilities, carelessness in any of these areas can have a real impact on the company's brand and, ultimately, the company's bottom line financials. Thorough testing, most specifically performance and security testing, helps mitigate the risks in this area.

Agile Becoming the Norm



9%
of organizations still using Waterfall method of dev and testing

77%
of organizations are doing or leaning toward Agile

24%
doing a mix of Waterfall and Agile

—2015 survey by HP

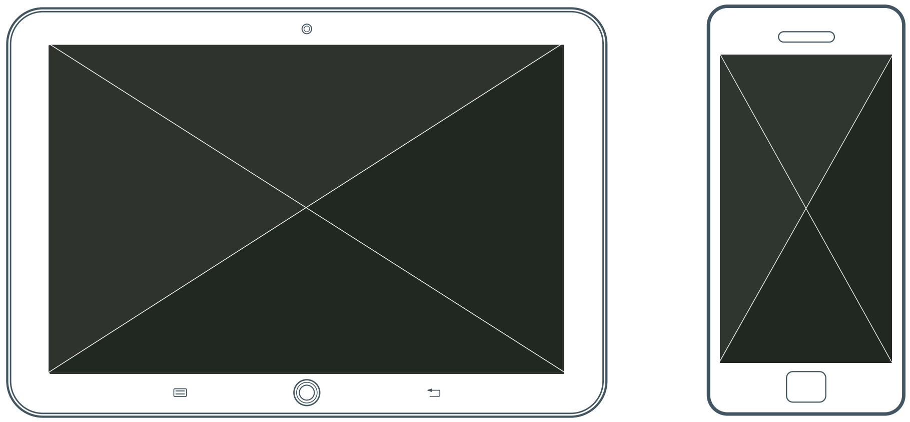
HOW MOBILE APP TESTING DIFFERS

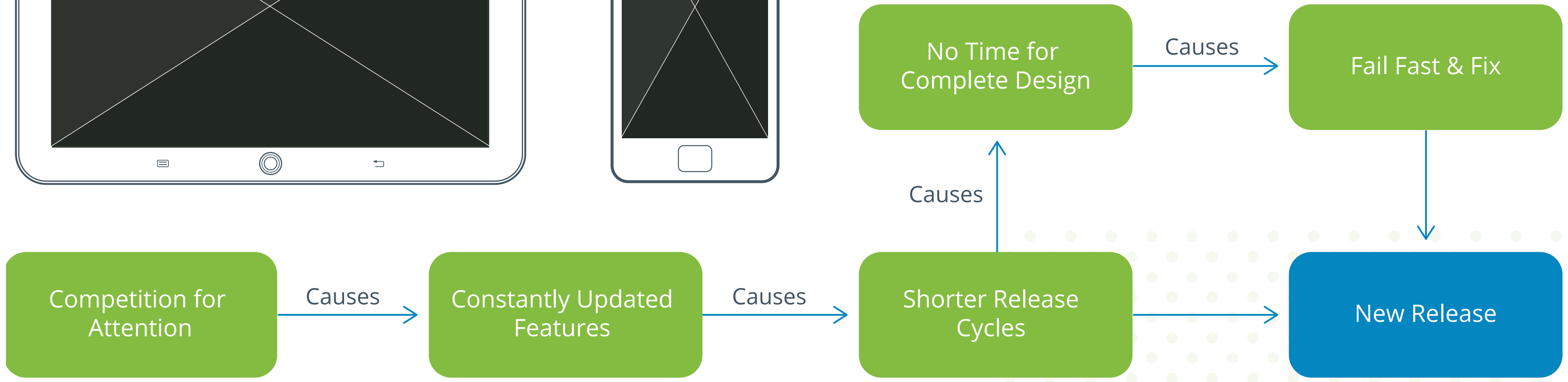
The speed at which the mobile market moves and the very nature of mobile applications and their consumption, make the testing of mobile applications different. Of course, mobile application testing will always require basic testing of internal code units,
the user interface, business logic and so on. But there are far more considerations for mobile application testing including the environments, device constraints, sensors, platform diversity, coverage and more.

How Mobile App Testing Differs

Mobile testing is impacted in other ways too. Competition for attention is fierce and users expect frequent updates for their apps. This forces the need to release new features in shorter cycles, giving less time for complete development and testing. Developers know they need to fail fast but deliver quick fixes to compete.

The Impact of Mobile On Software Releases

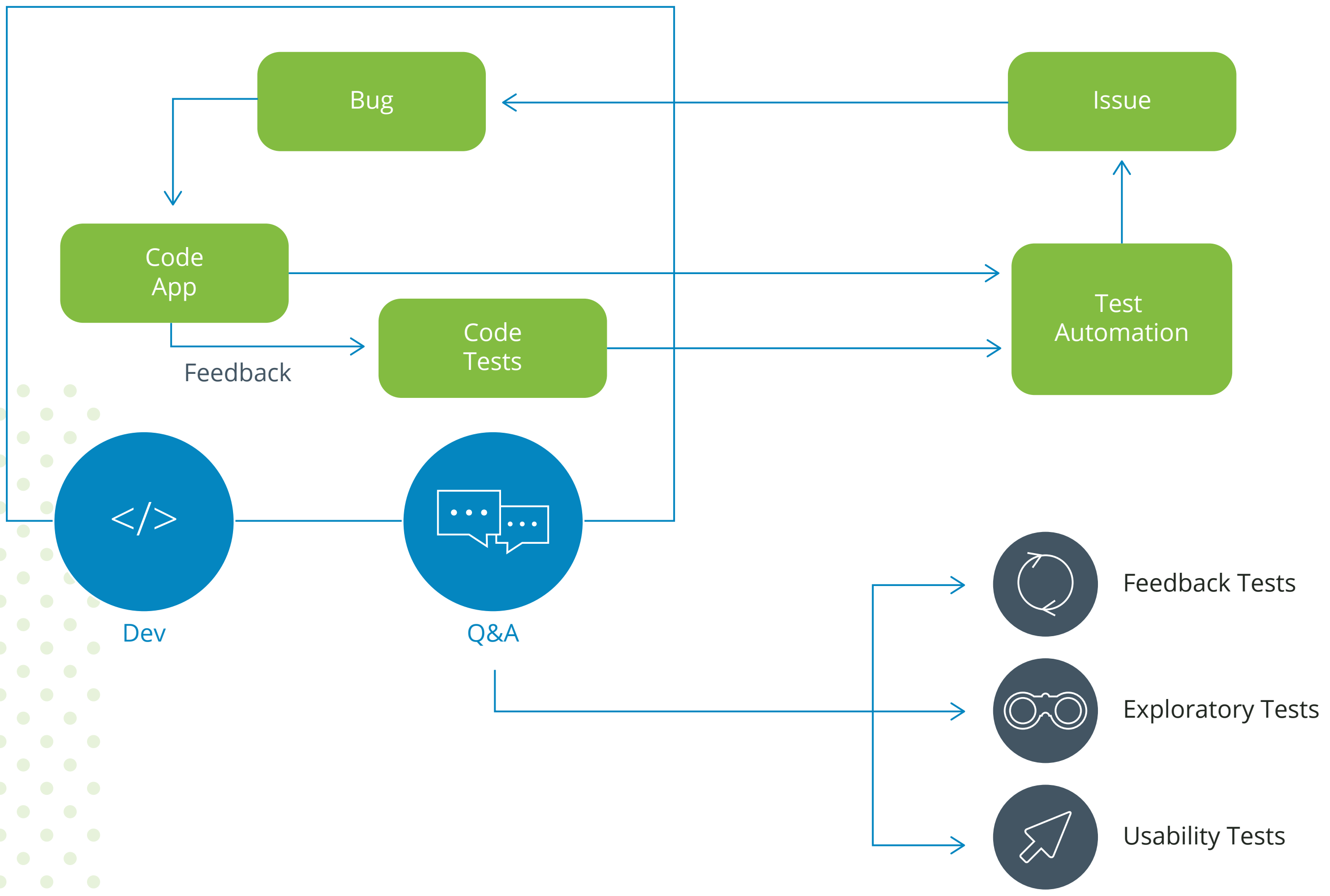



Competition for Attention
Causes
Constantly Updated Features
Causes
Shorter Release Cycles
Causes
No Time for Complete Design
Causes
Fail Fast & Fix
New Release

How Mobile App Testing Differs

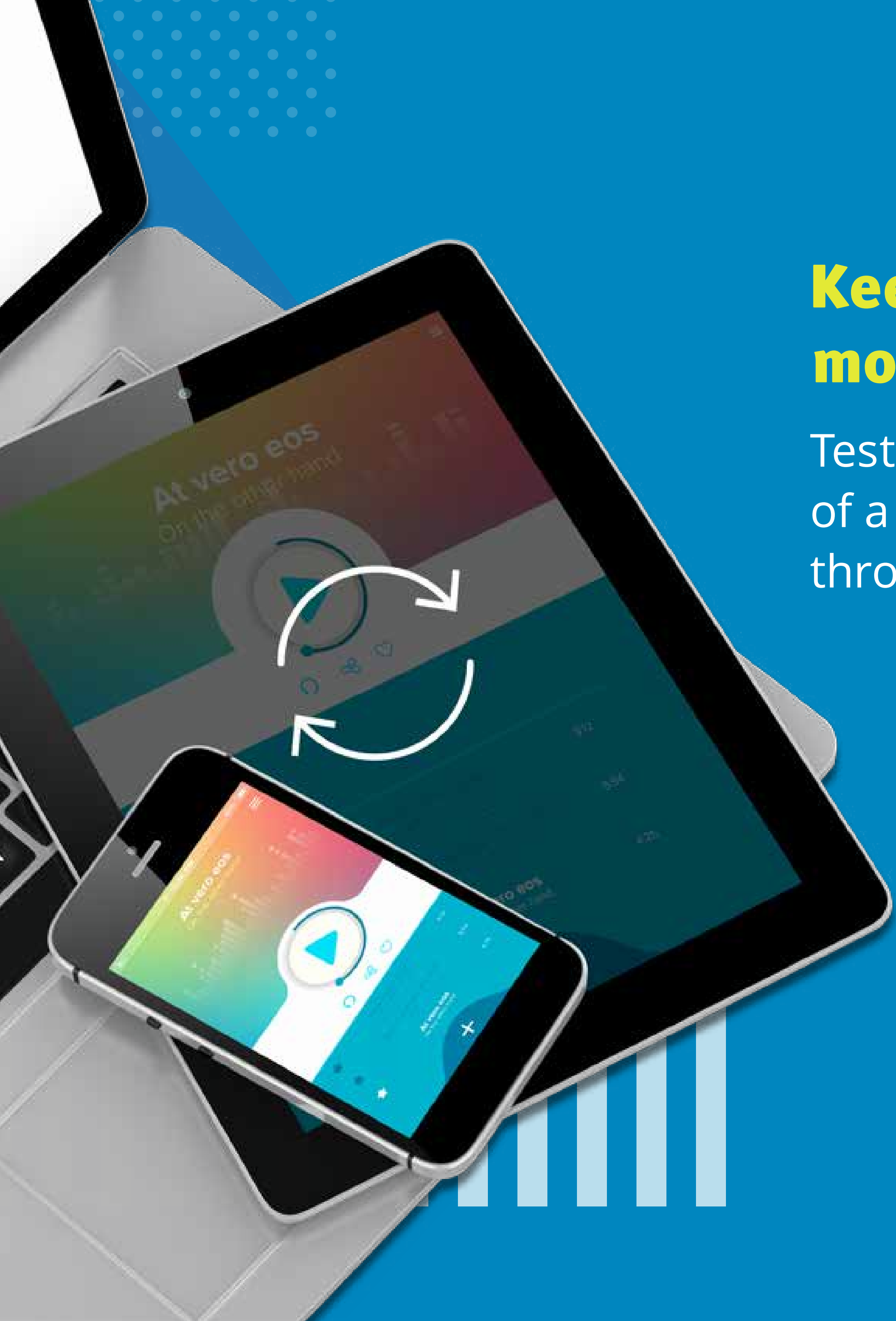
With the increased pace of mobile development, development organizations must reorganize in order to deliver the efficiencies required by these short release cycles. In the best organizations, testing has moved to the left, enabling development and QA resources to work more closely together to save time and deliver better apps. More testing responsibilities fall on the developer's shoulders and testers work closer to the design and implementation of features, so there's less need to hammer developers with questions about expected behavior.

The Impact of Mobile On Software Releases



Keep in mind this critical mobile app testing tip:

Testing is no longer done at the end of a development cycle, it's done all throughout the development process.



BEST TEST TARGETS FOR YOUR APP

Your mobile app must be tested on a complete representation of the environment under which the app will operate.

This includes testing the app with all possible permutations of the following:

- **Target device models**
- **Target device OS versions**
- **Mobile and Wi-Fi networks**
- **Target locales**[3]
- **Target languages**

This makes mobile app testing very complicated! For any potential commercially successful app, it would be a herculean task to accommodate every possible permutation; you'll have to do the best you can with the resources you have available to you. You can use our **Digital Test Coverage Index** to help you with your test strategy.

Environments Vary, Employ The Best Test Targets for Your App

The Downside of Emulators and Simulators

When testing a mobile app, you'll get the best results when testing the app on a physical device.

Unfortunately, mobile devices are expensive, and provisioning them for mobile network access makes them even more expensive. You can probably get away with testing on a subset of the available devices by selecting the most popular ones, but that is an incomplete strategy.

Each mobile device manufacturer provides device emulator or device simulator[4] applications that developers can use for testing. These applications create an operating environment that looks and functions like the mobile device model, but with potentially severe limitations. Any app built for the target platform will run on an emulator or simulator.

However, some simulators don't expose all of the functionality available in the target device such as the camera or hardware sensors.

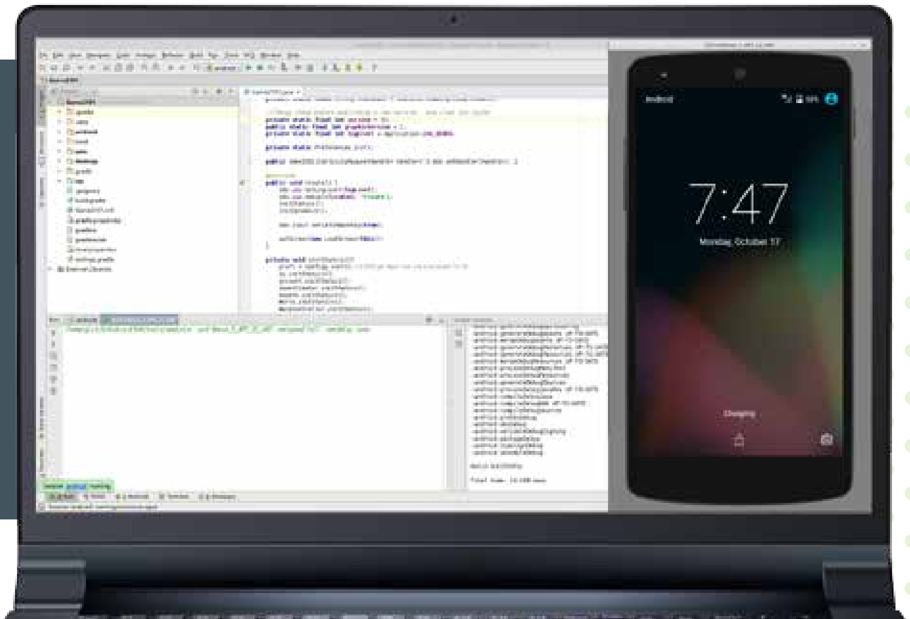
Emulators, on the other hand, have to emulate the target device processor, and therefore are often much slower than a real device.

Additionally, since these applications are running on a PC, leveraging system resources from the PC, it's not possible to obtain real performance (both app and network) data from tests.

For these reasons, emulators and simulators are suitable for lightweight testing of app features, but for best results, you should use real devices for most testing scenarios.

Drawbacks of Testing with Simulators and Emulators

- Designed for basic functional testing
- Can't test hardware components
- Can't test real world user conditions
- Can't test gestures
- Emulators test under ideal hardware, OS and network conditions, commonly resulting in false positives



Environments Vary, Employ The Best Test Targets for Your App

Networks are Fickle

Any app that interacts with server-based data sources may experience delays in obtaining the data it needs. Developers must accommodate this potential latency in their application user interfaces by implementing progress dialogs, activity indicators and more to help the user understand the app isn't stuck, it's merely waiting for data.

To properly test these applications, you must execute tests in real world scenarios such as varying network performance and other apps running on the device. Network characteristics vary globally, so be sure to test your app in each target environment. It's acceptable to do functional testing on devices over WiFi, but be sure to also test over cellular networks, or implement network virtualization into your testing process, when executing performance tests.

Another mobile app testing tip is to remember that network carriers in various locations also differ in their coverage of LTE, latency and download speed, all of which impact the user experience. Nothing can replace testing against real network conditions. It ensures a great user experience and strong app adoption.

Environments Vary, Employ The Best Test Targets for Your App

Language and Locale: English Is Not the Only Choice

When distributing apps globally, you must test your app on devices configured for each language and locale your app supports. Language direction, date, currency and number formatting can wreak havoc with an app's UI. Ideally, the app should be tested by native language speakers to ensure translations are technically accurate and politically correct. Start with English, of course, but use web analytics data to identify the other languages your site visitors use and prioritize those languages for inclusion as well.

Getting beyond the basics of mobile application testing for language and locale will mean keeping multiple translation files and other configuration files within the app (or on the server) that gets loaded to the app upon a language change. These files will help you go beyond just language changes, but could also change the app workflow and supported features per geography.

Web Browsers Are Not Created Equal

So far, most of what we've covered has addressed native mobile apps. Web apps are impacted by differences between device models and device OS versions, but not to the same degree as native apps. There's no guarantee that the desktop browser renders content the same as the mobile browser. When testing web apps, you must test the app on each browser, each browser version and each OS you expect visitors to use.

Browser combinations, standards and methods to test against are complex and need to be prioritized. **Check out this table** to get a peek into how deep and vast you can go with browser testing.

Start by using the default browsers for each target OS. Once the app is deployed, use the analytics capabilities of your web app to determine which browsers are the most popular and trim or expand your browser testing accordingly.

WHAT COMPONENTS DO I NEED TO TEST?

As you start planning your testing strategy and build out your test environment, one of the first questions you'll need to ask is: "What do I need to test?" The answer to that question will drive resource allocation and tool selection for your environment. We'll highlight some common areas of test coverage and describe the test types associated with these activities.

Areas of Test Coverage

Application

- Usability
- Performance
- Functionality
- Security

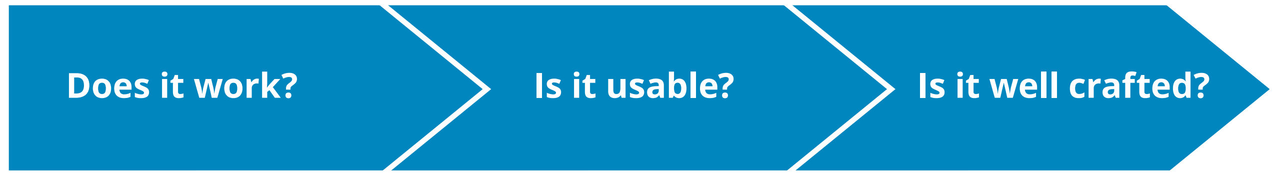
Server

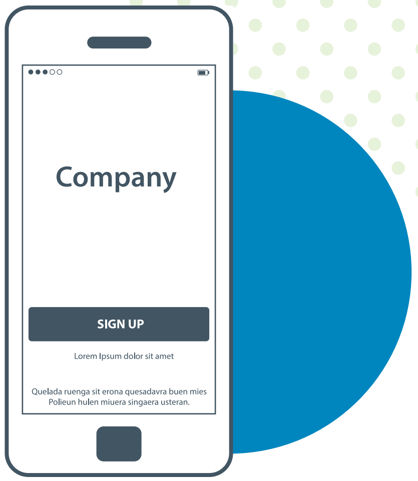
- Scalability
- Performance
- Functionality
- Security

WHAT DO I NEED TO TEST?

There are many types of tests that development organizations execute to validate app functionality, usability, performance and security. Some tests are executed during every test cycle and others may only be executed at specific milestones. Additionally, there are different software solutions and tools for each type of test. We'll categorize and describe each test in the following sections.

Test Types: What Do I Need to Test?




Company
SIGN UP
Lorem Ipsum dolor sit amet
Quelada ruenga sit erona quesadavra buen mies Polieun hulen miuera singaera usteran.

Does it Work?

The first group of tests answer the question: "Does the app work?" These are tests that put the app's modules, UI, field validation and business logic to the test.

Unit Tests

Unit tests are tests developers create within their application project to test the smallest consumable parts of the application: units.[8] Units are the module methods consumed by other parts of the application. You can validate these units while testing the app's UI, but there's no guarantee you won't be finding issues with the UI code at the same time. Unit tests are typically executed as a separate, stand-alone process, before passing an app on to QA. Unit tests are often executed automatically, as part of the build or check-in process, but they can be executed manually as well. Unit tests are written in the same language as the units being tested.

Functional Tests

Functional tests validate that the software component under test operates as described in the application's specification.[9] For a mobile app, functional tests focus on validating what the application does, testing the application's UI, and exercising the business logic behind it. Functional tests execute on a computer system or mobile device running the software under test. These tests can be performed manually, but mobile-savvy development organizations typically make heavy use of automation for these tests.

Many commercial and open source solutions exist for automating functional testing. A mobile testing basic is to focus on open source tools for maximum flexibility and to avoid vendor lock-in.

Test Types: What Do I Need to Test?

Real-World Test Conditions Matter

It's fine that your app works on your developer's test device or in your sterile lab environment, but for best indicators of quality, you must ensure that your app is repeatedly tested in dirty environments (low battery, intermittent network access, noisy environments and so on). Any assumptions you make about your app's operating environment will be wrong unless you plan for the worst. Don't forget to test how your app deals with being interrupted by other apps, notifications, phone calls and so on.

Manual Tests

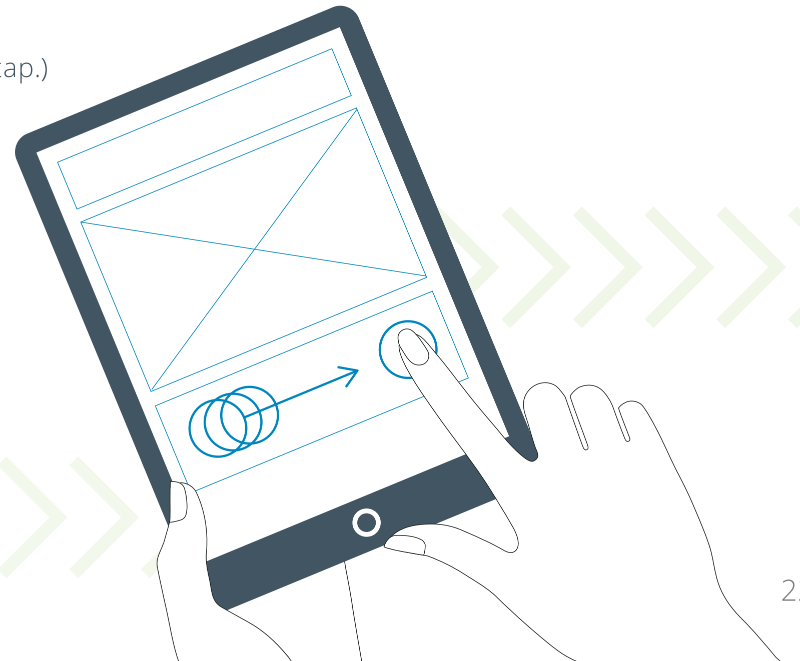
Manual tests are non-automated tests where testers validate the application's functionality and search for defects.[10] In these tests, testers assume an end user persona and exercise each feature of the application looking for issues. Manual tests are performed on a computer system or mobile device running the software under test. Testers use physical devices, emulators, and/or simulators and poke at the screen with their fingers or a mouse trying to mimic end user behavior.

Manual testing is an important part of the testing process but it is neither complete nor comprehensive. (There are only so many screens that human fingers can tap.)

Exploratory Tests

Exploratory testing is a form of manual testing where the tests have a specific scope, with a script to guide testing efforts.[11] Testers follow the script and identify issues encountered during the process. Exploratory tests are performed on a computer system or mobile device running the software under test. Exploratory testing is often integrated with test management software that enables testers to easily track progress through tests, capture screen shots and report issue details back to developers.

To go beyond mobile testing basics and ensure completeness, automation must be part of your testing strategy.



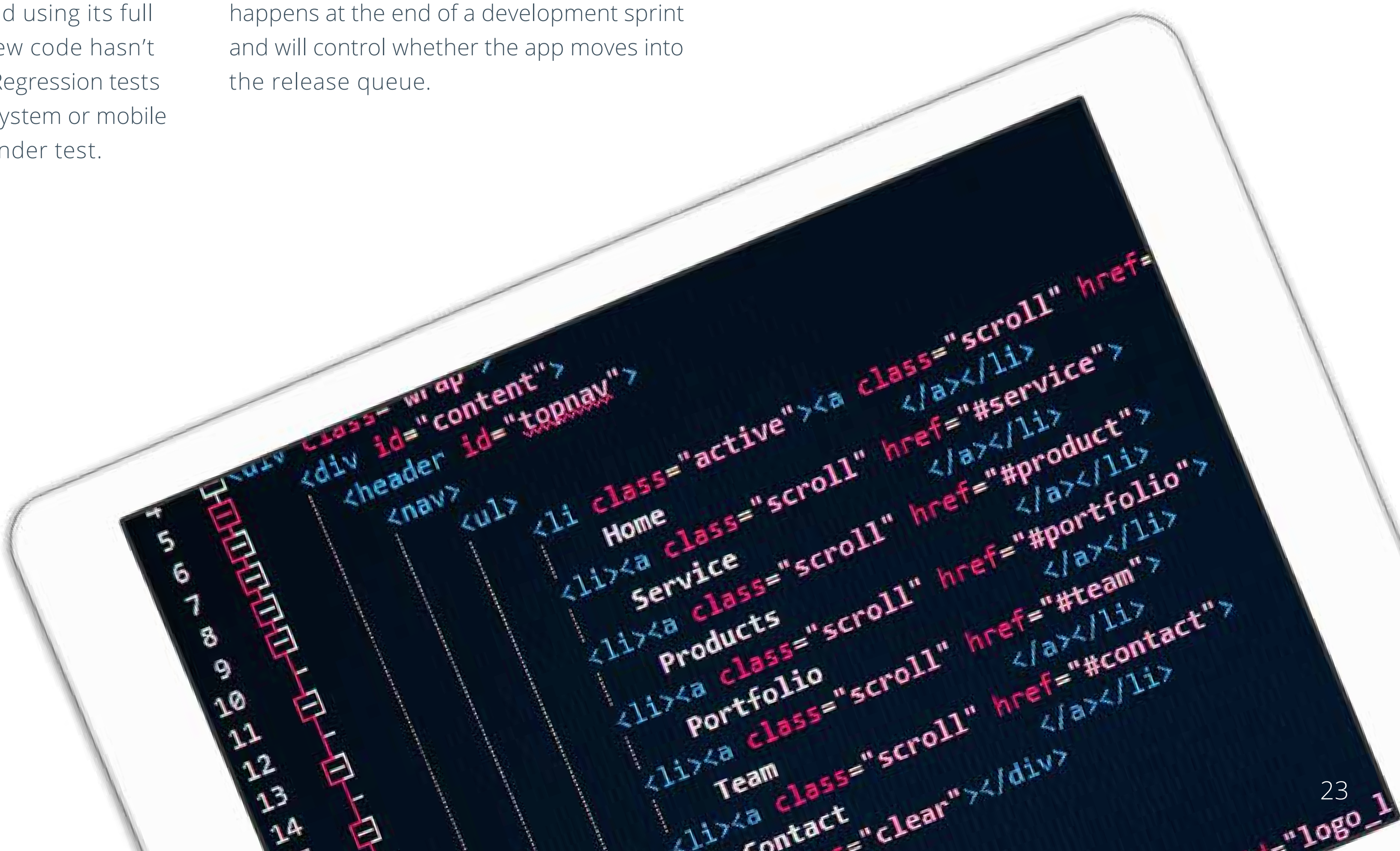
Test Types: What Do I Need to Test?

Regression Testing

When developers add features to an app, or enhance an existing app, developers or testers update the test suite to validate the new code. In this case, the new tests simply validate new or updated functionality. With regression testing, a software component is tested using its full suite of tests to ensure that new code hasn't broken existing functionality.[12] Regression tests are performed on a computer system or mobile device running the software under test.

Acceptance Testing

In acceptance testing, the product owner or product stakeholders review the app and validate that the implemented features meet their expectations and comply with the specification.[13] Acceptance testing usually happens at the end of a development sprint and will control whether the app moves into the release queue.



```
<div class="wrap">
  <div id="content">
    <header id="topnav">
      <nav>
        <ul>
          <li class="active"><a class="scroll" href=
            Home                        </a></li>
          <li><a class="scroll" href="#service">
            Service                     </a></li>
          <li><a class="scroll" href="#product">
            Products                    </a></li>
          <li><a class="scroll" href="#portfolio">
            Portfolio                   </a></li>
          <li><a class="scroll" href="#team">
            Team                        </a></li>
          <li><a class="scroll" href="#contact">
            Contact                     </a></li>
    ="clear"></div>
    ="logo_1
```

IS THE APP USABLE?

Once an app's functionality is validated through the "does it work" tests, development organizations must begin to deal with whether delivered features actually make sense and are usable by the app's target audience. The product development toolkit includes several tools that QA uses to validate whether designers and developers got it right.

Is the App Useable?

Usability Testing

With usability testing, testers poke and prod at app features with an eye toward whether features or feature implementations make sense.[14] The purpose of these tests isn't to find bugs, although new ones are often found, but merely to judge whether the app is usable for the purpose for which it was designed.

This type of testing cannot be automated. Tests are performed on a computer system or mobile device running the software under test. Usability testing is often performed by the QA department, but you'll also want to use testers who are not familiar with the app. The best usability testing results come from interested users who have limited experience with the app.

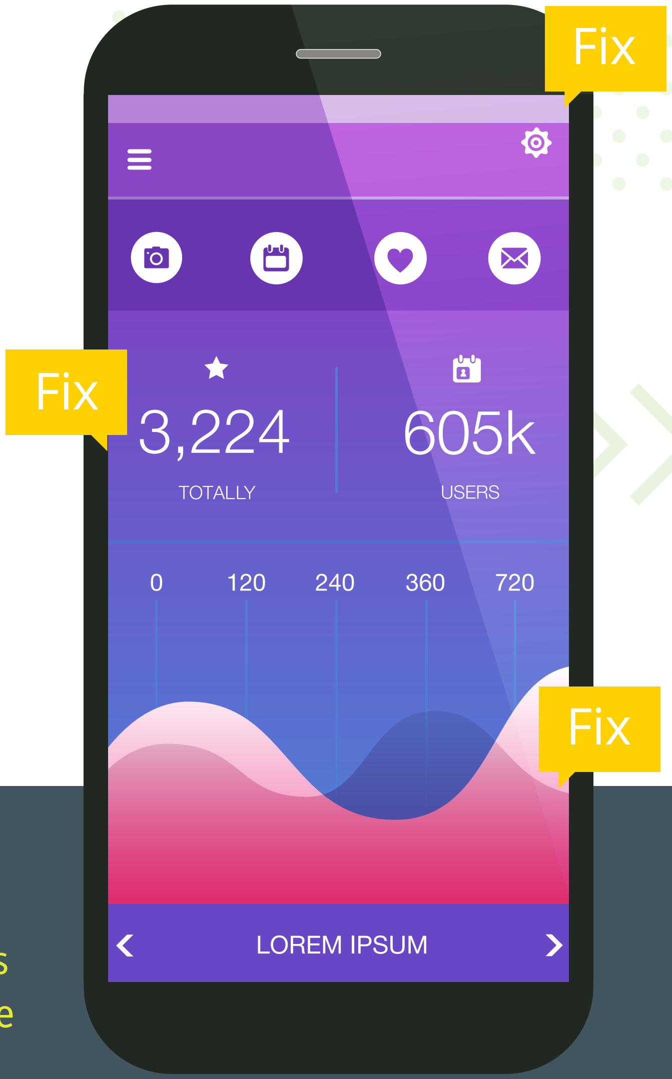
Development organizations often recruit usability testers from the following sources:

Company employees who are not part of the development or QA teams.

Existing users who volunteer to be beta testers.

Focus groups or random people on the street or in a mall.

Non-professional testers provided through third-party crowd testing services.[15]



Development and QA teams are too close to the app to see its warts. Allocate time and resources to obtain a third-party perspective as you implement key features.

IS IT WELL CRAFTED?

In parallel with determining whether the app does what it should be doing and that the app's features make sense for potential end users, are tests that validate developer craftsmanship and how well the app is made.

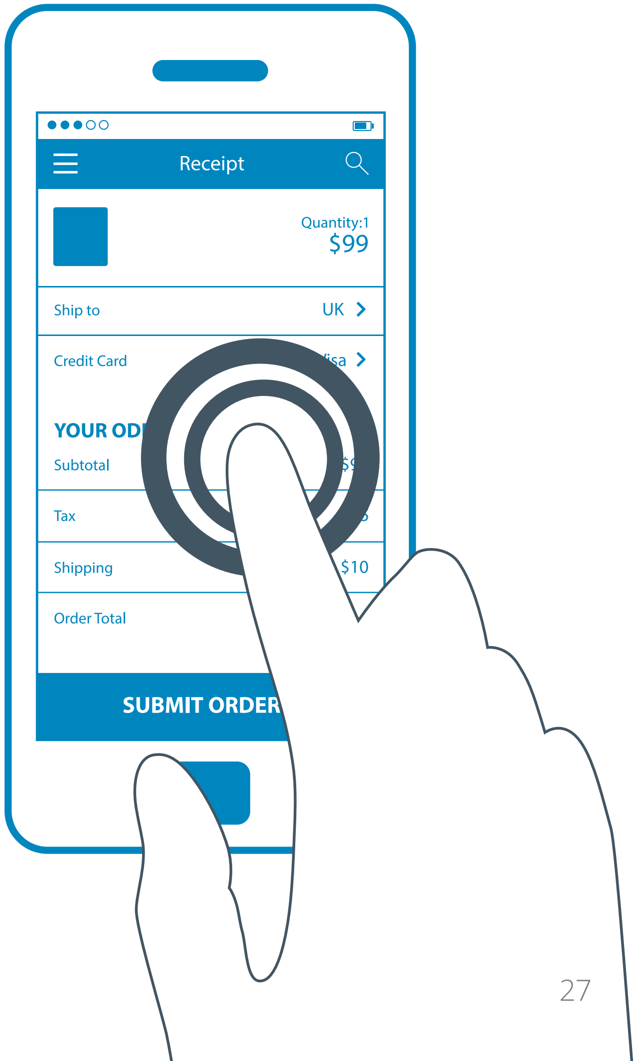
Is It Well Crafted?

Monkey Testing

A well-crafted UI should accommodate every possible way that a user will use the app. It's a reasonable goal, but developers don't always have an intimate understanding of the flows a user will try to take through their app. By exposing your app to a variety of users, both internal and external, you increase your chances of catching bugs.

But sometimes you just need more. Monkey testing is the process of randomly poking at different parts of the screen and entering random data into the app's input fields to determine whether the app handles it gracefully.[16] It's designed to find input validation errors and help you deliver a more solid product.

Monkey tests can be performed manually (although few people probably would do so), or there are automated solutions available as well. Google's Android Studio includes a monkey testing capability called the Monkey.[17] And there are several open source monkey testing tools for iOS applications. For mobile apps, you can run monkey tests on emulators, simulators or physical devices.



Is It Well Crafted?

Security Testing

Security testing assesses an application for vulnerabilities. Modern developers simplify many complicated development tasks using third-party libraries and open source solutions in their code. Additionally, developer blogs and support websites like Stack Overflow contain snippets of code that developers copy and paste into their own apps. These practices, coupled with the developers' own code, can contain known or unknown vulnerabilities that must be identified and removed.

Code reviews offer a manual way to identify vulnerabilities. With this approach, developers check other developer's code for issues before the code is released into production.[18] Most development organizations, however, implement software solutions for security testing. Security testing usually targets the application's source code, but some will run against the system under test to identify data leaks and so on while the app runs.

Penetration testing is another form of security testing. With penetration testing, one or more external systems poke and prod at an app (front-end or back-end) to see if it exposes any vulnerabilities.

Performance Testing

One of the most important aspects of app quality is performance. In performance testing, functional areas of the mobile app, as well as the back-end services that the app consumes, are measured as the app is tested.[19] Product stakeholders define key performance indicators (KPI) for the app, and the measurements are compared to the KPIs throughout the development process and beyond to ensure that the app performs as expected.

AUTOMATION

The pace of modern mobile development coupled with the variety of tests required to ensure delivery of a quality product have driven development organizations to implement automation solutions. From a testing standpoint, automation includes implementing software solutions for performing the various tests plus an automated execution environment for triggering test execution.

From an automation standpoint, organizations typically implement continuous integration[20] and/or continuous delivery[21] processes as a way to manage execution of most of the activities that happen after a developer checks updated or new code into the version control system. Many open source and commercial automation products are available.

Popular CI tools include **Bamboo**, **Hudson**, **Jenkins** and **Travis CI**.

Did You Know?

66% of enterprise organizations believe increasing test automation is the key to faster app releases.

—Why Apps Succeed report, Perfecto, June 2016

Get Your Copy

Automation

Common Automation Tasks Include:

Execute unit tests: Execute the project's unit tests to ensure that the app's core functionality is free of errors.

Execute security tests: Execute code scanners to identify vulnerabilities in the app's source code.

Build the target application: For mobile apps, execute the native SDK to package the app's source code into an executable application.

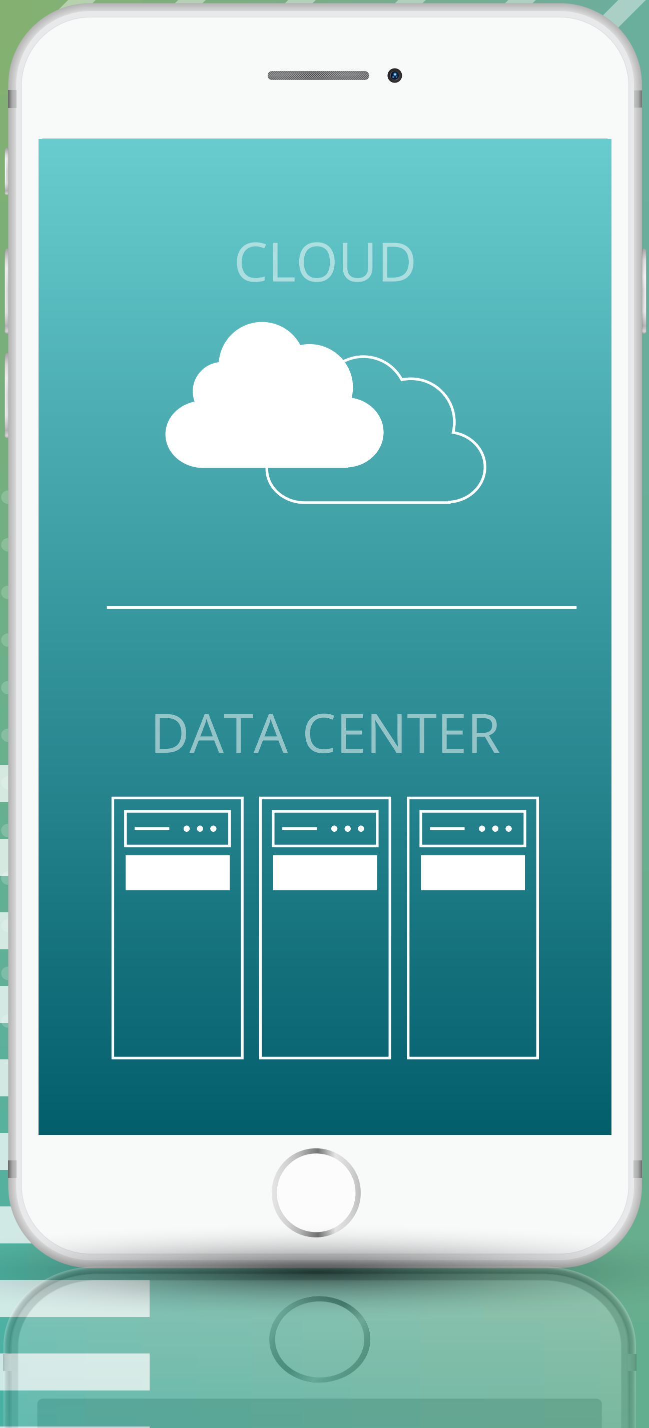
Execute functional tests: Kick off an automated validation of the app's functional tests scripts.

Continuous integration drives more frequent app testing. With the proper tools in place, a complete set of unit, security, functional and regression tests can be executed every time code is checked in. Without them, the QA department will collapse under the weight of all the tasks they must perform.

4 Benefits of Test Automation:

1. Fast feedback
2. Improved time to market
3. Broad test coverage
4. Keep pace with the market

Automated functional testing requires access to a robust and reliable device lab. Mobile app testing requires a large catalog of mobile OS and devices that must be tested with every release. Any ad-hoc or manual process for bringing devices online for testing will simply slow down the process and prohibit quick releases.



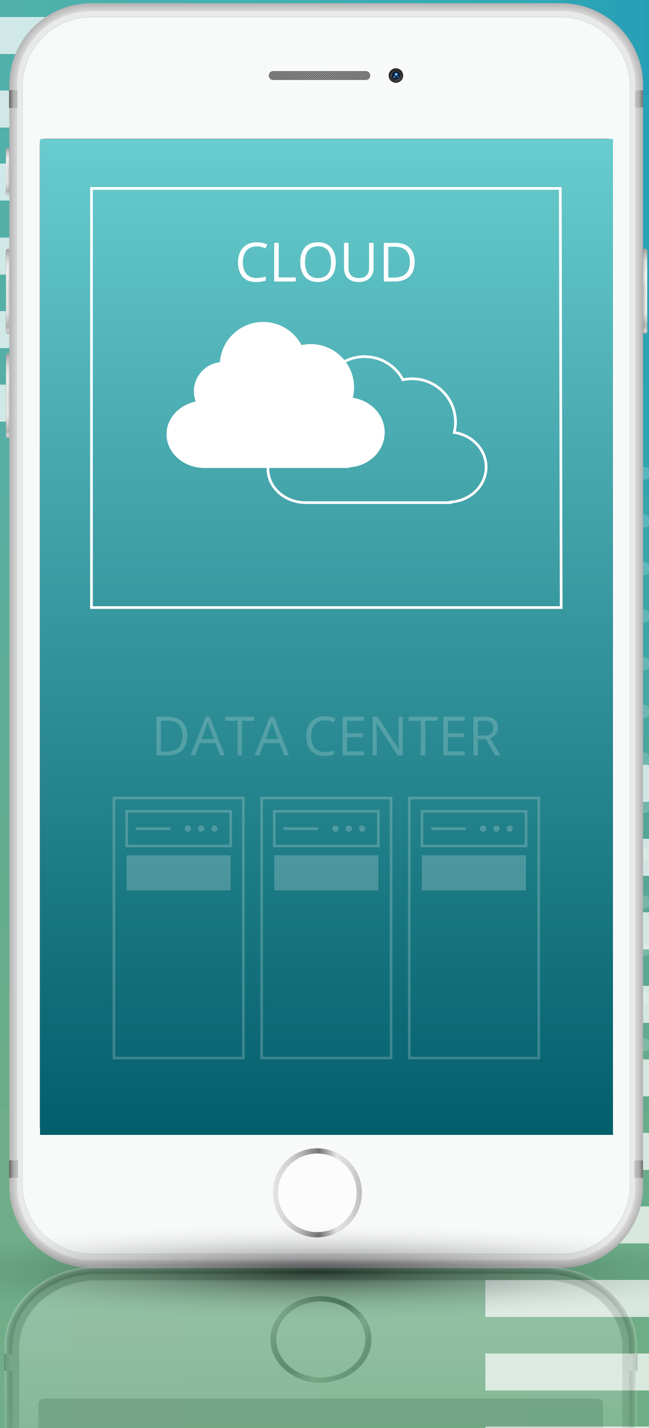
WHERE SHOULD I RUN MY TESTS?

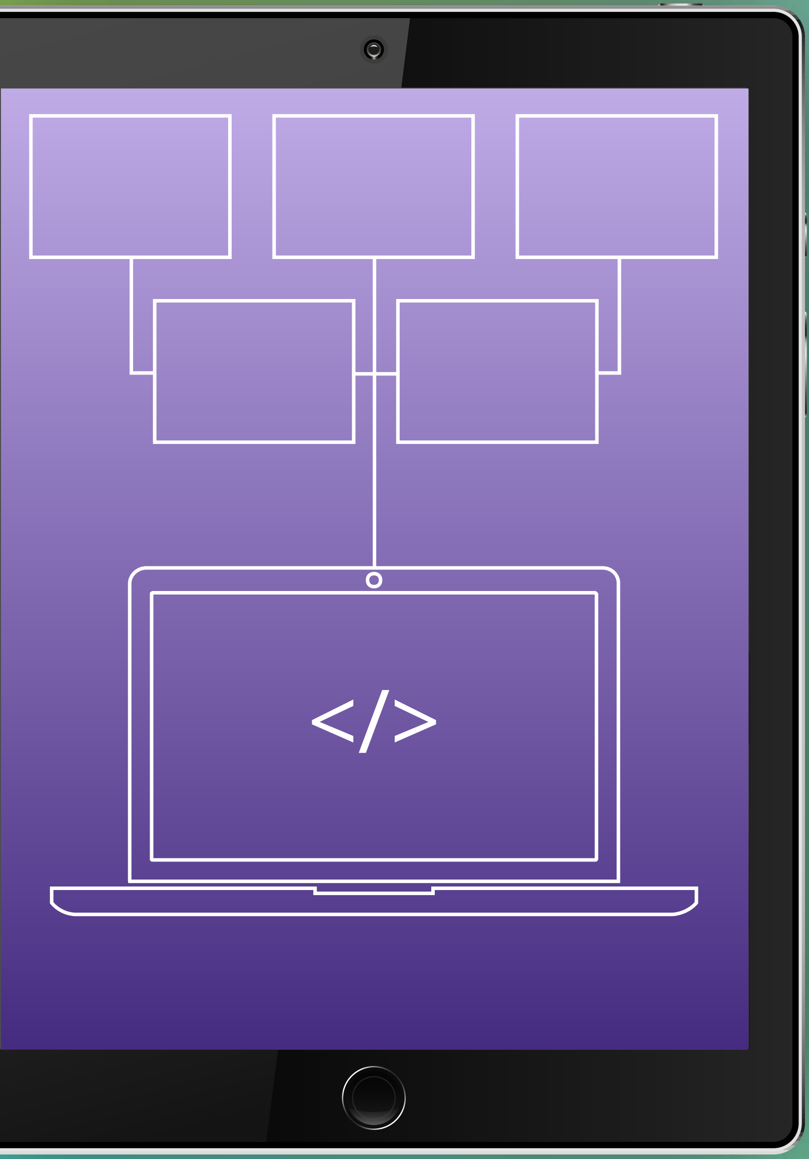
With all of the tools and capabilities we've described here, you're probably wondering whether all of this runs inside your data center or whether you can set up a cloud environment instead. The good news is that either option is valid. There are solutions that support one or the other deployment model and many support both. Your mobile app testing requirements will grow as you add new apps and enhance the ones you have, so any environment you use will need to be scalable.

WHICH IS BEST?

Once you have your testing technologies selected, it really doesn't matter where it runs. But building and maintaining a robust and reliable device lab can be both painful and expensive. The work there never really ends. Offloading device testing to the cloud provides considerable cost savings over in-house labs.

Deciding whether to build or buy?
Find out what it really costs.





TESTING CAPABILITY INSIDE YOUR ORGANIZATION

Depending on the structure of your development organization and the configuration of your development environment, you may or may not have what you need to deliver on your mobile testing requirements.

Key Takeaways for Setting Up a Modern Testing Capability Inside Your Organization

As you assess your testing capabilities or build your mobile app testing capabilities, keep the following mobile app testing tips in mind:

Organize for Success: Your existing development team structure may not work in the fast paced world of mobile app testing. Any silos surrounding dev or QA resources could hamper the efficiency of your delivery capabilities. Dev and QA should work closely together, eliminating barriers to quick and easy communication and facilitating collaboration between teams.

Invest in Automation: There's never enough time to manually test every app feature on every supported device. Also, certain types of tests such as performance and network testing simply can't be performed manually. It's only through implementation of automation solutions that you can deliver a fully tested mobile app within the timelines your users expect.

Recognize That Testing Never Ends: As app features increase and the development team grows, compounded with hardware and software launches and continuous market changes, QA will quickly find that they simply can't keep up with demand. Test maintenance becomes a full-time job and only through proper resource management and automation can apps be fully tested.

Structure Tests for Easier Maintenance: Over time, the catalog of tests that must be executed against your app will grow. As new features are added, or existing features changed, testers will abandon existing tests, write new ones or simply edit existing tests. Extra effort expended at the beginning of the project to structure tests for maximum efficiency will pay off later, as project complexity increases.

Regularly Optimize and Prune Test Suites: Implementing tests for new features or updating tests for changed features generates a fair amount of jetsam as developers jettison old code or duplicate code to test new use cases. QA must allocate a percentage of their time for cleaning up and refactoring existing tests for best efficiency.

Keep an up-to-date object repository that makes it easy to reuse existing code for new tests.

AM I DONE TESTING ONCE MY APP IS IN THE APP STORE?

No, not really.

There are many things that can cripple app performance after it's released:

OS Updates: Device manufacturers deploy OS updates that change APIs or API performance and the updated version may have a detrimental effect on your app.

New Device Models: New devices may appear that are not compatible with your app.

App Server Scalability or Availability: Your back-end infrastructure may have issues dealing with the load. The app may perform well, but when it reaches out to the back-end, things start to crawl.

Carrier Network Issues: Certain markets may have capacity issues or may be late in implementing upgrades, and your app stalls.

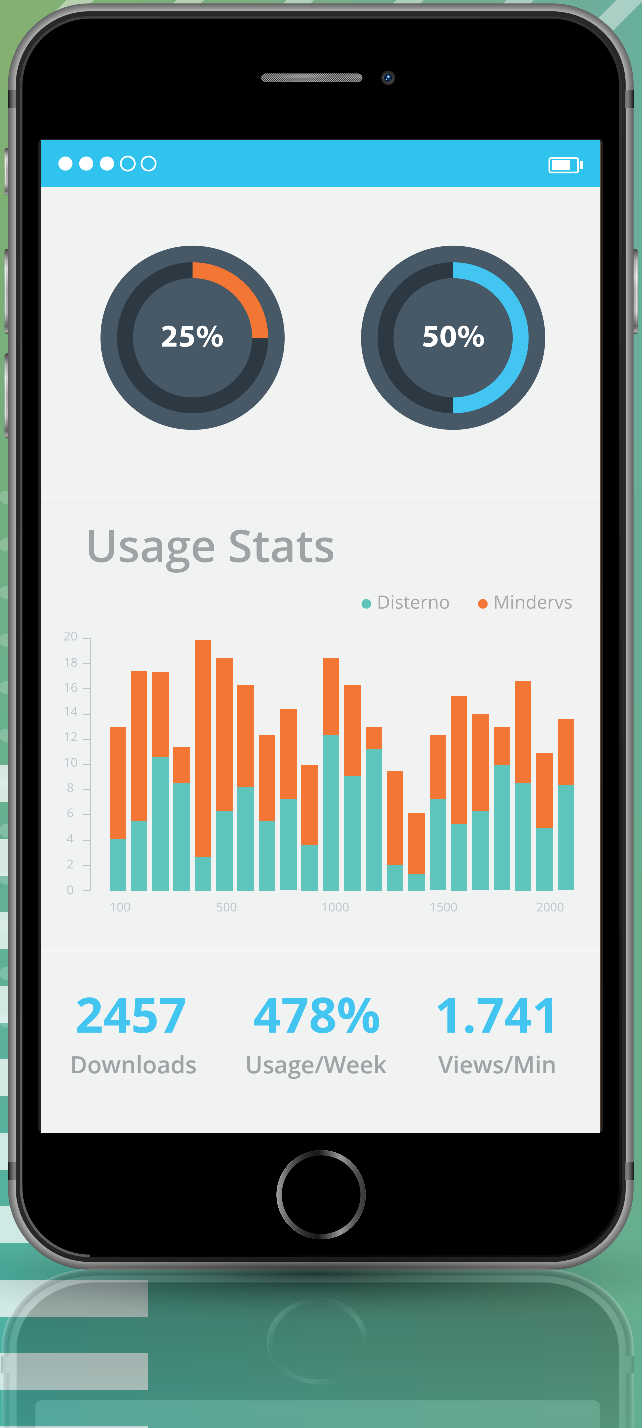
Am I Done Testing Once My App is in the App Store?

There are two ways for you to learn about these issues: you can wait for your end users to let you know, or you can pro-actively and regularly test your app to keep an eye on performance. This means you'll need to:

Keep your eye on the market and add new OS versions to your test suites as soon as they're released (and you have devices in your device lab running the new version).

Keep your eye on the market and add new devices to your device lab and execute your tests on them.

For your primary markets, keep a bevy of network provisioned mobile devices active in your lab and periodically execute performance tests on each. Keep a baseline of normal performance and set triggers to alert when KPIs aren't met.



ADDITIONAL TOPICS TO CONSIDER

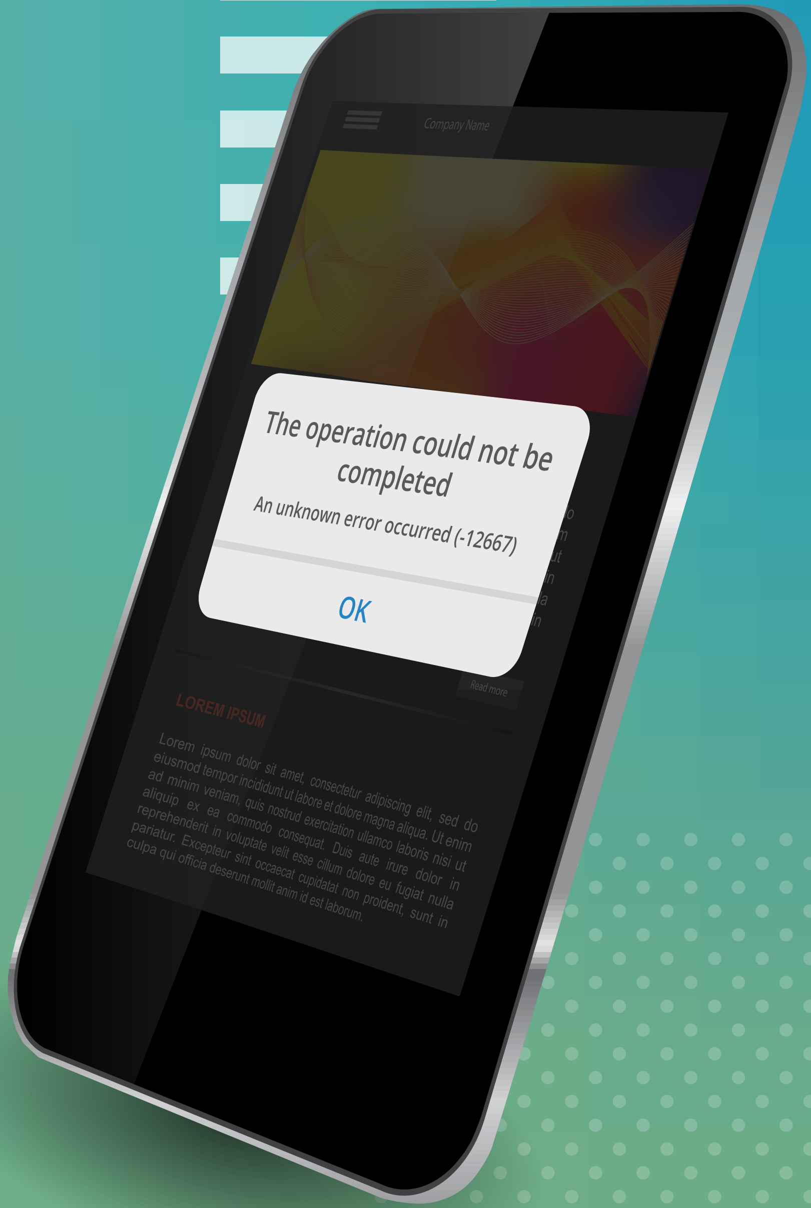
As you go beyond mobile app testing basics, you should focus your attention on several other topics that were not addressed here:

App Analytics: Instrument your app so you can monitor usage and identify flaws in your app's navigation or other features. Use collected data to focus future development efforts and to help make the case for continued investment in the app.

Business KPIs: Closely measure the impact of your app on business success. Identify which features drive the most revenue and adjust your focus accordingly.

Crash Detection: Instead of waiting for your users to let you know when your app has problems, implement technology that captures data about application crashes and other issues. Then use this data to fix the issues before users complain.

Monitor App Store Feedback: Watch app store ratings to identify issues you were unaware of. You can do this manually, but there are software solutions available as well. Many use sentiment analysis to gauge the feelings behind user feedback to streamline the feedback process.


The operation could not be completed
An unknown error occurred (-12667)
OK

GO FORWARD AND MAKE YOUR APPS AWESOME

Sources

1. A web application is an application written using web technologies (HTML, CSS, JavaScript, etc.) targeting desktop and/or mobile browsers.
2. A native mobile app is an app built using the native software development kit (SDK) for the target mobile platform. The app could be a pure native app, written using one of the native languages for the device SDK, or a hybrid app, a native app written using a non-native language for the device, but packaged into a native application for execution. Examples of this are Apache Cordova and Adobe PhoneGap apps that use web technologies to build native apps, apps built using React Native, NativeScript, TabrisJS, and others that are native apps written using JavaScript, and Xamarin apps, apps written using a non-native language like C# for all apps.
3. https://en.wikipedia.org/wiki/Locale_(computer_software).
4. In this context, a simulator is an application that mimics the capabilities of a specific mobile device, but may not execute the same code as the physical device. An emulator is a software application that behaves like the physical device because it's running the same software as the device.
5. https://en.wikipedia.org/wiki/Personally_identifiable_information.
6. https://en.wikipedia.org/wiki/Denial-of-service_attack.
7. https://en.wikipedia.org/wiki/Transport_Layer_Security.
8. https://en.wikipedia.org/wiki/Unit_testing.
9. https://en.wikipedia.org/wiki/Functional_testing.
10. https://en.wikipedia.org/wiki/Manual_testing.
11. https://en.wikipedia.org/wiki/Exploratory_testing.
12. https://en.wikipedia.org/wiki/Regression_testing.
13. https://en.wikipedia.org/wiki/Acceptance_testing.
14. https://en.wikipedia.org/wiki/Usability_testing.
15. Crowd testing services are interesting in that they provide the ability to select testers based on a wide range of criteria plus many offer tools that enable customers to view recorded audio or video of the user's reaction to the app.
16. https://en.wikipedia.org/wiki/Monkey_testing.
17. https://developer.android.com/studio/test/monkey.html.
18. https://en.wikipedia.org/wiki/Code_review.
19. https://en.wikipedia.org/wiki/Software_performance_testing.
20. https://en.wikipedia.org/wiki/Continuous_integration.
21. https://en.wikipedia.org/wiki/Continuous_delivery.

Perfecto

Seek Perfection

perfectomobile.com

About Perfecto

Perfecto enables exceptional digital experiences. We help you transform your business and strengthen every digital interaction with a quality-first approach to creating web and native apps, through a cloud-based test environment called the Continuous Quality Lab™. The CQ Lab is comprised of real devices and real end-user conditions, giving you the truest test environment available.

More than 1,500 customers, including 50% of the Fortune 500 across the banking, insurance, retail, telecommunications and media industries rely on Perfecto to deliver optimal mobile app functionality and end user experiences, ensuring their brand's reputation, establishing loyal customers, and continually attracting new users. For more information about Perfecto, visit www.perfectomobile.com, join our community follow us on Twitter at @PerfectoMobile.

Get content just like this delivered to your inbox!