SauceLabs

WHITE PAPER

Automated Mobile Testing Requires Both Real Devices and Emulators/Simulators

Introduction

Today, businesses compete in an increasingly mobile-centric marketplace. Modern mobile testing teams can no longer take a backseat in ensuring overall application quality. In their efforts to find the ideal environment to run automated mobile tests, organizations are conflicted between real mobile devices and emulators/simulators. One option promises more accurate test results, while the other delivers greater agility.

This white paper argues that organizations should use both these options according to their strengths for a best-of-breed mobile testing environment. It suggests a working model that can be applied to QA workflows right away for maximum benefit.

Table of Contents

Executive summary

Organizations committed to their mobile application quality efforts are faced with making an important choice regarding where to run their mobile tests. The default for development teams in large enterprises is to use real mobile devices. While this gives them more accurate test results, it is not ideal for scaling and automation of testing. Startups and SMBs may ignore real devices altogether as they're too expensive, and opt for the more convenient option—emulators and simulators. In doing so, they miss out on the real-world feedback that a mobile device can provide. This paper discusses the strengths and weaknesses of each of these options, and suggests a way to use them to take QA efforts to the next level.

The stakes are higher with mobile

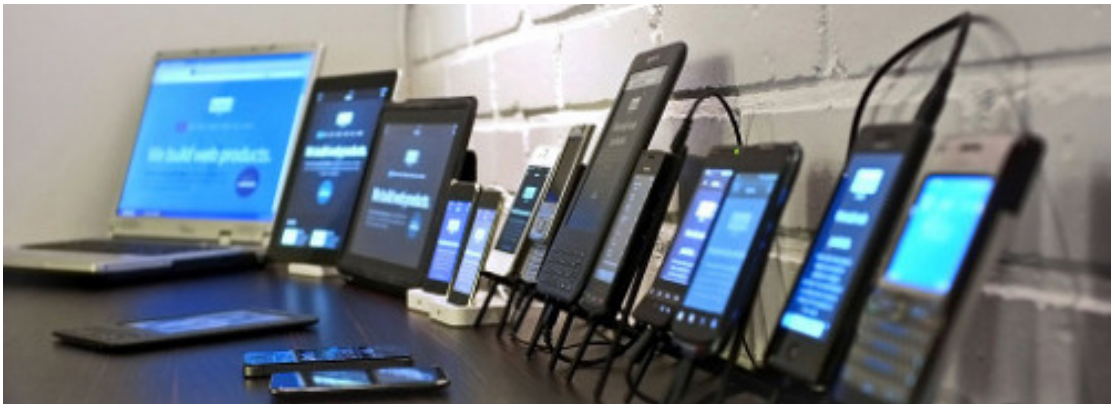
Business has gone mobile. Thus, mobile is now not only an important revenue generator for many organizations, but it is also key to customer satisfaction. Mobile application development, however, is not easy. It introduces several new variables. App stores have high standards for the apps they accept. The complexity of native applications mirrors that of client/server applications with the addition of many more variables, such as the wide variety of device and OS combinations. Testing plays an important role in ensuring apps meet today's high standards.

Similarly, there is a key difference between the release cycles of native applications and mobile web applications. It is much easier to develop and roll out updates for web applications that can be deployed in minutes or seconds, multiple times a day. They can be automatically accessed by the users and be rolled back as needed. For native applications, the release cycles are longer and complex, and fixing a bug in production can be both costly and time-consuming. As an example, an app installed on the device of an end user cannot be rolled back, which can lead to multiple versions of the apps in production. Therefore, testing earlier in the release cycle becomes even critical for native apps.

Mobile users are equally demanding. Because of how integral a mobile device is to their day-to-day routines, users demand that mobile apps have a mobile -first UX with advanced functionality. Additionally, they expect the app to be stable and free from crashes and bugs. Putting user expectations first is important with mobile, because It's harder to acquire new users and easier to lose existing users on mobile than on the Web. Organizations that are out of touch with these trends risk falling behind the competition, losing users, and ultimately, revenue.

Relying On Real Mobile Devices Alone Cripples Mobile Testing

Software testing went from testing in the data center to testing web apps in the cloud using browser automation tools like Selenium. However, with mobile, many large organizations prefer to do the bulk of their testing on real devices. While the use of automated testing is on the rise, many apps are still built on a developer's Mac, and are manually tested on the few devices available in a device lab.



These drawbacks result in inefficient launches that are focused of functionality, and user interface; they ignore important factors like stability, networks, and laggy client performance.

Similarly the lack of user feedback pre-production creates a bottleneck for developers to troubleshoot issues before roll-out. And post-launch, developers have to rely heavily on users to act as their debugging layer so they can fix issues, which can quickly get very costly.

Low-quality releases show in poor ratings that flood the app listing, which result in fewer new installs, lower daily average users (DAUs), and eventually lost revenue.

Cloud-based real devices - Flexibility without the maintenance

Like real devices in a device lab, real devices in the cloud run tests on actual phone hardware and software. However, the key difference is that cloud based devices are housed on a vendor's premises and are accessed remotely by sending test scripts to the devices over the Internet. These scripts are executed on the devices, and test results are sent back in the form of detailed logs, error reports, screenshots, and recorded video.

When building a device lab in-house, a comprehensive range of devices is needed for genuine confidence in the quality of testing efforts. Most organizations aim to test on a list of devices that represents a majority of their user base, preferably around 80%. This could mean anywhere from 20-50 devices. Even if this seems manageable at the start, the device lab would still need to be constantly updated by replacing outdated devices with new ones that become popular every quarter. Further, maintaining all these devices over a period of time can take focus away from core QA activities. Real devices could address these issues by allowing testing on a broad range of devices that won't need updating, or maintenance. This frees up the QA team from the hassle of procuring new devices, and gives them the confidence of having the latest devices available for testing immediately.

Another way real devices in the cloud trump those in the lab is through monitoring and analysis. With a device lab, the available tools for monitoring tasks is inadequate, and troubleshooting is often done manually by a human running each test to replicate the error and find the root cause. With devices in the cloud, vendors build in robust monitoring tools that track and report on every step of the test, and relay it back for analysis. This way testing teams can simply look at a detailed error log, view screenshots, or watch parts of a recorded video to identify the cause. This level of monitoring can be built into an in-house device lab, but the effort takes months, and requires adequate people resources that could otherwise be put to better use. Because vendors try to provide many devices to test on, they have only a limited number of any one type of device. This may pose problems with device availability during peak periods. Some vendors address this issue by charging a premium and dedicating devices to an organization in a private cloud so devices can only be accessed by them. Others offer a large number of specific device types, only supporting the most popular devices, which eliminates any queuing for a device at the expense of breadth of coverage.

Testing for real user conditions (network simulation, localization, GPS, etc.) makes a big difference in the quality of a mobile app. In that sense, real devices in the cloud bring a marked improvement in flexibility, scalability, visibility, and cost efficiency over devices in a lab.

Any mobile testing team that takes the quality of their mobile app seriously should consider real devices in the cloud as a key component of their mobile testing strategy.

Use Emulators And Simulators To Scale & Automate Mobile Testing

In an attempt to move away from testing on physical devices, some organizations have switched to using emulators for their testing.

An emulator, as the term suggests, emulates the device software and hardware on a desktop PC, or as part of a cloud testing platform. It is a complete re-implementation of the mobile software written in a machine-level assembly language. The Android (SDK) emulator is one example.

A simulator, on the other hand, delivers a replica of a phone's user interface, and does not represent its hardware. It is a partial re-implementation of the operating system written in a high-level language. The iOS simulator for Apple devices is one such example.

Key benefits of Real Devices vs Emulators/Simulators

	Real Devices	EMU/SIM
Easy to Provision		✅
Easy to Scale		✅
Facilitates Automation		✅
Detect Hardware Failures	✅	
Advanced UI Testing	✅	
Easy to Maintain		✅
Cost Efficient		✅

Overview of what to test where:

	Real Devices	Emulators/Simulators
Functional testing for large integration builds		✅
UI layout testing	✅	✅
Panel/ compatibility testing	✅	
System testing	✅	
Display testing (pixels, resolutions)	✅	
Replicate issues to match exact model	✅	
Camera mocking	✅	
UX testing	✅	
Push notifications	✅	
Natural gestures (pinch, zoom, scroll)	✅	

Emulators and simulators are much faster to provision than real devices, as they are software-driven.

Additionally, they enable parallel testing and test automation via external frameworks like Appium/ Espresso/XCUITest. Selenium revolutionized the world of web app testing by pioneering browser-based test automation. Today, Appium is its counterpart for mobile app testing. Appium uses the same WebDriver API that powers Selenium, and enables automation of native, hybrid, and mobile web apps. This brings huge improvements in the speed of tests for organizations coming from the manual world of testing on real devices. Similarly, enabling test automation with native frameworks (Espresso/ XCUITest) provides better test reliability, speed, and flexibility for native application testing.

Emulators and simulators enable parallel testing in a way that can't be achieved with devices in a lab. Because tests on emulators and simulators are software-defined, multiple tests can be run on tens of thousands of emulators and simulators at the click of a button without having to manually prepare each emulator/simulator for the tests. However, mobile QA teams that start to use emulators and simulators may swing to the other extreme of stopping all testing on real devices. While this speeds up the testing process, it comes with a critical drawback — emulators/simulators can't fully replicate device hardware. This makes it difficult to test against real-world scenarios using an emulator/simulator. Issues related to the kernel code, the amount of memory on a device, the Wi-Fi chip, layout changes, and other device-specific features can't be replicated on an emulator/simulator. It's not enough to test on emulators and simulators alone. Real devices are an important part of the mobile application quality process.

Emulators, Simulators & Real Devices - Better Together

Today, organizations fall under one of two extremes. They either rely only on real devices, or only on emulators and simulators for their mobile testing. Some organizations test exclusively on real devices with the assumption that they aren't compromising on the quality of their tests, while other organizations test exclusively on emulators and simulators because they are faster than real devices, easier to maintain, and cost much less. However, both of these extremes are a compromise. Real devices have drawbacks in terms of scalability and cost. Though emulators and simulators are an improvement on real devices, they are unable to deliver a real-world testing environment.

The ideal mobile testing strategy employs a mix of emulators/simulators and real devices. This option addresses the scalability and cost inefficiencies that come with real devices, while retaining the ability to test under real usage conditions. It provides the best of both worlds.

Once the decision has been made to add emulators and simulators to the mix, there may still be uncertainty about which tests to run there, and which tests to run on real devices. The exact answer to this question would vary for each organization, but there are certain guiding principles that can help.

A hybrid approach for a quick start

This approach uses emulators, simulators and real devices according to their unique strengths and weaknesses.

One way to decide where to run mobile tests is based on immediate testing needs. For example, if an app is in alpha stage, pixel-perfect UI testing isn't necessary, and the team wants to run multiple low-level tests in parallel, emulators are the best bet. On the other hand, if the goal is a revamp of the user interface of an app, and the look and feel and exact color shades matter, it may be best to lean towards real devices.

This hybrid approach of picking and choosing where to run which tests is a great way to start small and not be overwhelmed by all the changes. The key is to start somewhere, and build upon the starting point. However, as testing matures, a more structured way of using emulators, simulators and real devices may be necessary. The T-shaped approach to testing is a great way to leverage emulators and real devices according to their strengths.

T-shaped approach for the mature QA team

The T-shaped approach is a popular analogy in recruiting that's endorsed by Tim Brown, CEO, IDEO. According to this metaphor, candidates with T-shaped skills are those with working knowledge of a wide range of skills (the horizontal bar in the 'T'), and specialize in one of those skills (the vertical bar). This model can be useful when deciding where to run test scripts.

> According to this approach, emulators and simulators are used for the majority of tests across the pipeline, and gain wide testing coverage. While real devices are used for specific tests that require in-depth testing and to gain a higher degree of confidence in real-world scenarios.

Following the Continuous Integration (CI) model of development, the goal is to iterate fast and frequently, all through the pipeline, but more so in the initial stages. Emulators and simulators are well suited for this because they are cheaper, easier to provision, scale, and manage than real devices.

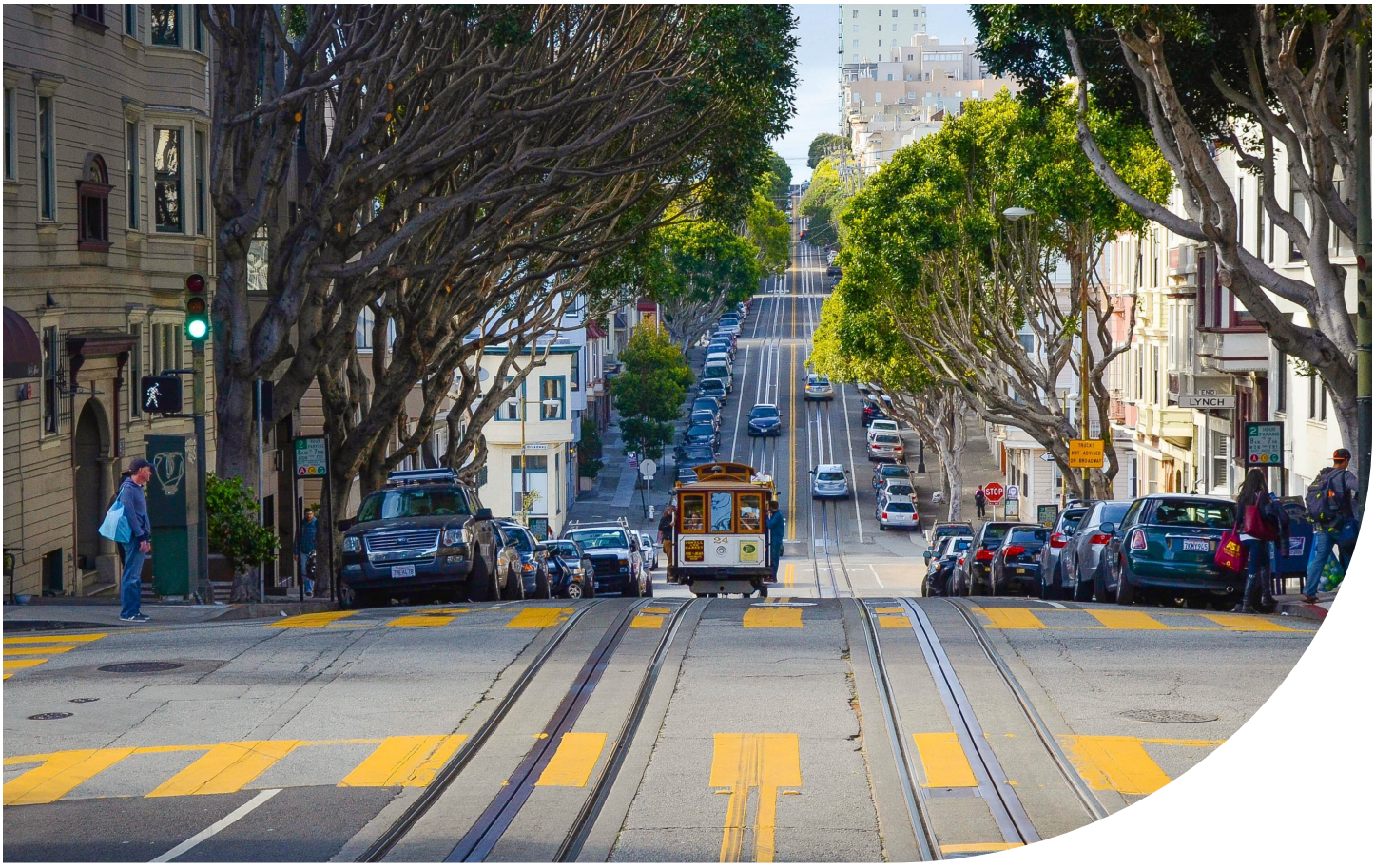
In the later stages, when device-specific features are being tested, real devices trump emulators and simulators. Since most of the basic tests are done with emulators, it's possible to have much fewer iterations using real devices. Because of the real world feedback they deliver, the use of real devices can be treated like system testing prior to release.

This approach allows automation of an entire testing matrix across emulators, simulators, and real devices, which provides real-world results, while optimizing costs.

This is the winning combination to launch an app successfully, and drive accelerated adoption from its user base.

Conclusion

Emulators and simulators are complementary to real devices, but they can't deliver the real-world environment that a device can deliver. Real devices and emulators, when used together in an automated testing environment, enable modern mobile development and testing teams to get the most out of their mobile testing efforts. And, testing in parallel across multiple platforms helps speed up tests while optimizing costs. Any organization that competes in the mobile marketplace cannot afford to ignore the value of using real devices, emulators, and simulators in their mobile quality efforts.



About Sauce Labs

Sauce Labs is the leading provider of continuous test and error reporting solutions that gives companies confidence to develop, deliver and update high quality software at speed. The Sauce Labs Continuous Testing Cloud identifies quality signals in development and production, accelerating the ability to release and update web and mobile applications that look, function and perform exactly as they should on every browser, operating system and device, every single time. Sauce Labs is a privately held company funded by TPG, Salesforce Ventures, IVP, Adams Street Partners, and Riverwood Capital.

For more information, please visit
→ saucelabs.com

saucelabs.com/sign-up

FREE TRIAL

SAUCE LABS INC. - HQ 450 Sansome Street, 9th Floor, San Francisco, Ca Usa 94111

SauceLabs