

Mobile App Testing



**SPECIAL
REPORT**

In today's digital-driven world, enterprise mobility teams are tasked with controlling the chaos of testing multiple applications across multiple platforms, operating systems, and device types.

In this Mobile App Testing Special Report, you will learn how to create a mobile application testing strategy that combines the best of manual and automated testing using the industry's best technology and tools.

In this Mobile App Testing Special Report

From Selenium to Appium: Mobile Web to Mobile App Testing

How did we go from mobile web testing to mobile app testing, and how did this trend result in the birth of Appium? The answer—Selenium.

Visual Regression Testing: A Critical Part of a Mobile Testing Strategy

Despite our best efforts to replicate customers' behavior in our test automation suites, teams often forget about nonfunctional requirements. An important one is visual perception—how users see and feel each application they use. Find out how visual regression testing can fill a significant gap in user experience expectations.

3 Strategies for an Efficient Mobile App Testing Project

Developing a testing strategy for a mobile app means facing multiple challenges that are part of the field. But by employing these three strategies, you can get valuable feedback about your mobile app's functionalities before it is published—while still keeping time and budget in mind.

Need A Hand? Benefits of Real-Time Manual Testing

Today's manual testers are on the front lines of QA teams bringing the delicate art of "people-testing" to mobile app and mobile website testing. But, in an industry that is constantly demanding faster, better, and more dynamic mobile experiences, what exactly is the role of manual testing in modern enterprise mobility?

Using Computer Vision to Reduce Test Automation Blind Spots

The standard test automation toolkit fails to detect certain elements on desktop and mobile content-based applications. Computer vision (CV) uses deep learning technology to determine objects in pictures, which helps machines orient in space and perform repetitive detection tasks. Find out how CV can help automate the testing of a broader software product list.

Should I Use a Simulator or Real Device When Dev Testing Mobile Apps?

While simulators and emulators play a role in the mobile app development and testing process, there are certain tasks that are best suited for a real device. This checklist will help you decide which to use for mobile app development and testing.

Additional Mobile Apps Testing Resources

3

From Selenium to Appium: Mobile Web to Mobile App Testing

6

Visual Regression Testing: A Critical Part of a Mobile Testing Strategy

7

3 Strategies for an Efficient Mobile App Testing Project

9

Need A Hand? Benefits of Real-Time Manual Testing

15

Using Computer Vision to Reduce Test Automation Blind Spots

18

Should I Use a Simulator or Real Device When Dev Testing Mobile Apps?

23

Additional Resources

From Selenium to Appium: Mobile Web to Mobile App Testing

By Jamie Moore

It is often difficult to remember life before mobile. From bulky car phones in bags, to flip phones, to today's smartphones and tablets, it is amazing how quickly technology has evolved and its influence on consumer behavior.

As the types of available devices expanded, and internet connectivity continued to speed up, so did the abilities of mobile devices to connect, entertain and captivate the minds of consumers via the mobile web. And then, the explosion of mobile apps happened, which was a game changer for testing teams. Now simply testing mobile web was not enough – teams needed a solution to test these robust mobile apps.

With mobile apps and mobile web, consumers are able to do virtually anything on their mobile devices. Thanks to the rapid growth of mobile apps, testers found themselves with an entirely new platform for testing.

Before the explosion of mobile, web served as the primary platform for testing. Testers relied on tools such as MicroFocus' UFT One (formerly HP) and Selenium. But recently, Appium has made a huge impact in enter-

prise mobility for testers interested in test automation for mobile apps. When it comes to generating industry buzz, Appium has been a hot topic from thought leaders and vendors at enterprise mobility events worldwide.

But, how exactly did we get to this point? How did we go from mobile web testing to mobile app testing, and how did this trend result in the creation of Appium?

To answer this question, it is important to take a look at the history of web testing and Selenium.

Selenium and Web Testing 101

Before Selenium, when it came to web testing tools, UFT One was the clear market leader for testers. As a commercial tool, UFT One offered a robust support community and testers benefited from being connected to a larger ecosystem of tools formerly available from HP. Today, testers using UFT One are now a part of the MicroFocus ecosystem, thanks to its acquisition of UFT One. Today, testers leveraging UFT One automation enjoy using a stable tool that has stood the test of time and is ideal for testers who are not as savvy when it comes to writing scripts and complex programming.



Although many testers embrace commercial tools there are other testers who thrive in an open-source environment and chafe against the restraints from commercial tools like UFT One. Therefore, Selenium was developed to bring more freedom to web testing.

COPYRIGHT 2020

3

3

From Selenium to Appium: Mobile Web to Mobile App Testing

6

Visual Regression Testing: A Critical Part of a Mobile Testing Strategy

7

3 Strategies for an Efficient Mobile App Testing Project

9

Need A Hand? Benefits of Real-Time Manual Testing

15

Using Computer Vision to Reduce Test Automation Blind Spots

18

Should I Use a Simulator or Real Device When Dev Testing Mobile Apps?

23

Additional Resources

This freedom in scripting languages opened up the world for testers by bringing much need flexibility when writing test scripts.

Selenium prides itself on being a tool that simply automates web browsers. As an open-source solution, testers are empowered to harness Selenium as a means of automating web applications for testing. Testers with the appropriate skill sets and hunger for innovation can really make the tool their own through customization.

As Selenium began to rise in popularity it pushed UFT One out of the #1 spot for web testing tools. In fact, Selenium captured such mindshare in the industry that it is actually the core technology used by other browser automation tools, APIs and testing frameworks thanks to its powerful capabilities.

In addition, Selenium supports many different scripting languages, while commercial tools usually support only one language. This freedom in scripting languages opened up the world for testers by bringing much need flexibility when writing test scripts.

Selenium also utilizes the web browser's native API when testing and easily integrates with other open source tools for additional options for web testing and integration with other tools in the testing lab.

It is important to note that unlike commercial solutions, Selenium requires testers to have programming skills to write tests. For teams without deep programming skills, Selenium may be a challenge to use. For teams that have programming capabilities, Selenium allows testers to write tests in their language of choice and to their desired specifications.

At-A-Glance: Benefits of Selenium

- Open source
- Supports a variety of scripting languages
- Utilizes the web browser's native API
- Easily integrates with other open source tools

Even though Selenium is still a robust testing tool used today for web testing, the growth of mobile apps required a new tool for testing on this platform. Thus, Appium was born to fulfill this need and has its roots in the Selenium framework.

Appium and Open-Source Community

Like Selenium, Appium was developed by the open-source community to test mobile apps. Basically Appium is "Selenium for Apps," as it is based on Selenium's Webdriver technology. Appium functions by using the native APIs of iOS and Android to communicate to the apps or web browser on a mobile device.

For history buffs, you can check out this website to get a comprehensive history of Appium. But in a nutshell, Appium was born from Co-Creator Dan Cuellar's need to test an iOS app while he was a test manager at Zoosk in 2011. To accomplish this task, he built iOSAu-

3

From Selenium to Appium: Mobile Web to Mobile App Testing

6

Visual Regression Testing: A Critical Part of a Mobile Testing Strategy

7

3 Strategies for an Efficient Mobile App Testing Project

9

Need A Hand? Benefits of Real-Time Manual Testing

15

Using Computer Vision to Reduce Test Automation Blind Spots

18

Should I Use a Simulator or Real Device When Dev Testing Mobile Apps?

23

Additional Resources

to, using the same underlying philosophy as Selenium by leveraging a native API. In 2012, Cuellar met Selenium's co-creator, Jason Huggins and together they created Appium using Selenium Webdriver's wire protocol over HTTP.

Although Appium in its earliest form has been around since 2012, it is only recently that Appium has really begun to resonate on a wide-scale in enterprise mobility. As a testing tool for apps, it offers many of the same benefits that Selenium does for web testing. Today, Appium has an active, open source community that works to expand the tool's capabilities.



Also, many thought leaders and vendors in enterprise mobility have started to add Appium support capabilities into their current solutions. As Appium continues to grow in popularity, expect to see more features and functions from vendors and more insight from thought leaders around this testing framework.

Perhaps one of the most interesting things about the relationship and connection between Selenium and Appium lies in the power of pairing the two solutions for testing—bringing a powerful suite of tools to today's testing labs for web and mobile app testing.

Selenium + Appium = A Powerful Toolkit for Testing

For testers that are familiar with Selenium, and have used the tool, getting up to speed with Appium for mobile app testing is an easy learning curve. Because Appium is built on Selenium's framework, the skills are easily transferable and intuitive.

For teams that are using Selenium and Appium, testers can also leverage current web browser test assets (provided the object names are the same). It is important to note that for both native and hybrid apps while the structure of the tests are similar, the type of objects in the tests are different.

By leveraging both tools in today's modern testing lab, testers have the best of both worlds.

By leveraging both tools in today's modern testing lab, testers have the best of both worlds. If they are familiar with Selenium and use it for web testing, then bringing in Appium to test mobile apps is a logical next step. If teams are using Selenium for web testing, then these scripts can be easily converted for Appium using a mobile web browser.

As an extension of Selenium that was built using its capable technology, it is easy to understand why the excitement around Appium continues to grow, with Selenium still maintaining its popularity as a trusted tool for web testing.

3

From Selenium to Appium: Mobile Web to Mobile App Testing

6

Visual Regression Testing: A Critical Part of a Mobile Testing Strategy

7

3 Strategies for an Efficient Mobile App Testing Project

9

Need A Hand? Benefits of Real-Time Manual Testing

15

Using Computer Vision to Reduce Test Automation Blind Spots

18

Should I Use a Simulator or Real Device When Dev Testing Mobile Apps?

23

Additional Resources

Visual Regression Testing: A Critical Part of a Mobile Testing Strategy

By Dmitry Vinnik

As a child, did you ever play spot the difference—spending hours trying to find how two pictures differ from one another?

Now imagine doing this with thousands of mobile devices, looking for visual disparities in your app. At some point this game becomes impossible to play, especially at the scale of the modern mobile market.

In the QA community, functional testing has become a must-have activity. Every major software project has a test automation suite to validate the most common scenarios customers might perform in the production.

Unfortunately, despite our best efforts to replicate customers' behavior in these automation suites, teams often forget about nonfunctional requirements. An important one is visual perception—how users see and feel each application they use.

Let's say we have a mobile app where to access user settings, a customer needs to click on the Settings icon in the top right corner. To replicate this scenario, we have a functional test to find that Settings icon

by selectors, press on the icon, and validate that the expected window pops up.

What do you think was the most nuanced phrase in the above scenario? "Find that icon by selectors." Your customers do not use selectors when pressing a button, typing text, or dismissing a popup. If your users don't see a button, they can't push on it, right?

This situation is especially widespread in the mobile industry for a variety of reasons, including screen dimension differences among devices. If you had only ever tested your application on an iPhone X, but your customer tried to use an iPhone 3 with a much smaller screen, there is a chance that the user settings icon got pushed away from the screen so that nobody could press it.

This scenario demonstrates the importance of visual testing in filling a significant gap in user experience expectations. With more operating systems, form factors, and mobile devices available on the market, customers expect their favorite app to work on both mobile and web, with no difference in how the application looks or feels.

If—or, better, when—you're interested in visual testing, there are a few easy ways to get started. Every team can begin by building on top of their functional test suite. For example, if you have Selenium WebDriver automation, your team can enhance existing validation by targeting individual CSS values. To take it to the next level, Selenium can be used to control the width of the browser to account for different platforms like mobile, desktop, or table.

After making the first step with CSS validation, you can move on to screenshot testing. There are tools that let users start with visual testing in a matter of minutes while offering a scalable solution to store and manage test artifacts, whether in a configuration-driven setup or by building on top of existing test infrastructure.

It will not take long for you and your team to notice a difference in how many visual regressions you catch internally versus what comes from your users. Whatever way you choose to get started with visual testing, it is all a matter of taking the first step.

3

From Selenium to Appium: Mobile Web to Mobile App Testing

6

Visual Regression Testing: A Critical Part of a Mobile Testing Strategy

7

3 Strategies for an Efficient Mobile App Testing Project

9

Need A Hand? Benefits of Real-Time Manual Testing

15

Using Computer Vision to Reduce Test Automation Blind Spots

18

Should I Use a Simulator or Real Device When Dev Testing Mobile Apps?

23

Additional Resources

3 Strategies for an Efficient Mobile App Testing Project

By Thomas Raynott

With the mobile app market on the rise, developers strive to bring their products to market as quickly as possible. As a result, quality assurance teams are under more pressure than ever to effectively utilize their resources and deliver app testing with more speed and accuracy.

Teams might be dealing with tens of thousands of testers who need to be directed through the testing process. Getting feedback about the mobile app's functionalities before it is published is crucial. But it's also important that testing teams learn how to reduce the cost and effort of app testing.

Here are three ways to manage a mobile testing project with resource efficiency in mind.

Consider the Main Testing Challenges

My team and I have been testing mobile applications for a while now, and we have become aware of testing challenges that are unique to the mobile environment. To ensure the high quality and performance of the end product, a testing team needs to take these a couple of main aspects into account.

The first is significant device fragmentation. Contrary to desktop or web apps, mobile applications will be used on a variety of devices and platforms. And don't forget that operating systems come in numerous different versions as well. The fragmentation of mobile devices is a huge challenge for developers who want to create different versions of the same app and ensure that it works correctly with different versions of a given operating system. QA departments need to remember that such operating systems have different capabilities, making it harder to secure and manage the app.

Mobile testing tools and resources also have to be considered. We often decide to outsource selected testing areas because of tool availability. Advanced testing tools and efficient methods that allow for testing with multidevice compliance in mind are simply unavailable to in-house testing teams. That's why so many organizations choose to bring in the talent and tools through working with a testing partner.

Because testing budgets aren't increasing and deadlines are steeper than ever, those who decide what the testing process will look like need to seriously con-



sider outsourcing testing activities. Outsourcing can be a good option because it provides developers with access to qualified professionals and testing tools, but it takes a while to find a partner that can be trusted. In our experience, outsourcing certain testing activities can be a huge help to reducing testing costs. Most of the time we choose to outsource marginal app elements and properly focus on the core activities of the app in house.

COPYRIGHT 2020

7

3

From Selenium to Appium: Mobile Web to Mobile App Testing

6

Visual Regression Testing: A Critical Part of a Mobile Testing Strategy

7

3 Strategies for an Efficient Mobile App Testing Project

9

Need A Hand? Benefits of Real-Time Manual Testing

15

Using Computer Vision to Reduce Test Automation Blind Spots

18

Should I Use a Simulator or Real Device When Dev Testing Mobile Apps?

23

Additional Resources

Decide between Emulators and Physical Devices

Another critical decision in developing a testing strategy concerns the choice between physical devices and emulators.

In the early stages of development, we usually use device emulators because they allow us to quickly test different components of our apps and work well in agile development environments. Emulators are cost-effective and are perfect if what you have in mind is testing basic application functionalities. We often use device emulators while developing app features as well.

But that doesn't mean we avoid using physical devices altogether. Experience has taught me that there is no way you could publish a successful app without testing it on a physical device. If you fail to test your app on actual devices, you will never truly understand how its activities play out in real-life scenarios. Only physical devices allow you to test factors such as different battery states, multiple networks ranging from Wi-Fi to 4G, or network density. Physical testing also gives developers a glimpse into how users interact with the app and how the app works on different devices. That type of testing environment simply can't be re-created on an emulator.

Therefore, the best strategy to test an application in an efficient and cost-effective manner is by mixing device emulators and physical devices. The right balance depends on what your QA team needs.

Secure Beta Testers

That brings us to the topic of beta testing and beta testers. It's smart to start looking for beta testers among users who are already loyal to your products, such as people in your online forms or who communicate with your company on social media. In my experience, one of the best ways to identify users who regularly interact with an app is through the help desk. These are often people who are actively engaged with your app, and many of them will be more than happy to become beta testers for your next release.

Some companies use crowdfunding platforms to source beta testers for their apps. They prepare the app in alpha stage and publish it on a platform like

Kickstarter to drive awareness, possibly securing more funding as well.

Another way to get an unbiased set of beta testers is a limited launch. Distributing your app to only a specific country or region lets you test new features of your app before a wide release. However, getting the best selection of beta testers doesn't guarantee good results if your app doesn't include monitoring or analytics tools. That's how you can track the ways in which users interact with the app and check which features aren't working properly.

It's smart to build that feature of your app in a way that enables you to switch on and off different app functionalities. This way, you can have one group of beta testers test one feature and another group test a different feature. That's the best way to get a realistic view of how the app performs when tested in the wild, and it's more effective than asking users to report on their activities or fill out surveys.

Developing a testing strategy for a mobile app means facing multiple challenges that are part of the field. But by employing these three strategies, you can get valuable feedback about your mobile app's functionalities before it is published—while still keeping time and budget in mind.

If you fail to test your app on actual devices, you will never truly understand how its activities play out in real-life scenarios.

3

From Selenium to Appium: Mobile Web to Mobile App Testing

6

Visual Regression Testing: A Critical Part of a Mobile Testing Strategy

7

3 Strategies for an Efficient Mobile App Testing Project

9

Need A Hand? Benefits of Real-Time Manual Testing

15

Using Computer Vision to Reduce Test Automation Blind Spots

18

Should I Use a Simulator or Real Device When Dev Testing Mobile Apps?

23

Additional Resources

Need A Hand? Benefits of Real-Time Manual Testing

By Mobile Labs

Today's manual testers have their hands full literally and figuratively. Daily, manual testers are on the front lines of QA teams bringing the delicate art of "people-testing" to mobile app and mobile website testing. But, in an industry that is constantly demanding faster, better and more dynamic mobile experiences, what exactly is the role of manual testing in modern enterprise mobility?

People testers, the good news is that we're here to give you a "hand." If you're using the human touch to "manually" test mobile apps and websites, and think you need a device in hand to get the full experience, think again.

As all testers know there are simply some tests that are best left to manual testers. A good example of the type of work best left to manual testing often occurs during the development phase, when developers are coding new functionality into an app. Manual testing should also be utilized when testing any mobile app or device hardware features such as a barcode scanner that uses the device's camera.

Although automated testing is often faster, especially for regression testing, manual testing is unique because it puts human judgment and creativity at the fore. Because human judgment and creativity cannot be automated (at least not yet), it is up to manual testers to answer important questions about a mobile app in the testing phase such as:

- Does everything look right?
- Are there new or non-obvious paths?
- How might a user misunderstand what's being presented?
- Is there something a test script might know to do that some users will not find easy or intuitive?
- What happens when the user makes a mistake or does the unexpected?
- How does the app make a user feel?
- Is the app design appealing?

In most modern mobile testing labs, manual testing is a crucial part of the testing process. But things really get interesting when you consider the implications of cloud-based mobile testing.



Cloud-Based Manual Mobile Testing

So, why would a manual tester want to implement a mobile device cloud in their mobile testing lab? What's wrong with using a personal device, or selecting a shared device from a drawer? It really all comes down to the need to be fast and to test in real-time. Unfortunately, valuable testing time is lost when devices are lost, dead, or used by a team member. There has to

COPYRIGHT 2020

9

3

From Selenium to Appium: Mobile Web to Mobile App Testing

6

Visual Regression Testing: A Critical Part of a Mobile Testing Strategy

7

3 Strategies for an Efficient Mobile App Testing Project

9

Need A Hand? Benefits of Real-Time Manual Testing

15

Using Computer Vision to Reduce Test Automation Blind Spots

18

Should I Use a Simulator or Real Device When Dev Testing Mobile Apps?

23

Additional Resources

be a better way to streamline manual testing, which is where a mobile device cloud solution comes into play.

Utilizing a mobile device cloud solution enables testers to test a large variety of devices and OS versions quickly. But sometimes this cloud access comes with Internet lags that slow down the work or even cause some events to be missed. With smooth action, however, cloud-based manual testing can equal the speed and accuracy of having dozens of devices in-hand. With a powerful enough mobile device cloud, manual testers, you can literally fly. Seriously.

Just think of what you could accomplish with this power. The ability to supercharge manual testing lies in implementing real-time manual testing leveraging a high-performance mobile device cloud. A mobile device cloud will enable manual testers to thrive in three distinct areas: Device access, interactivity and productivity.

Let's take a closer look at these three areas:

#1: Access & Share Devices 24/7, No Matter Where You Are

Take a moment and think about where your devices are. Where do you store the devices you use for manual testing? Do you share them with others? How

often do they get lost? What shape are they in when they are returned to you? Clearly, managing mobile devices is a time-consuming process. And due to the ever-increasing number of smartphones and tablets, purchasing multiple devices for testing is expensive.

For mobile developers and testers that are located in different offices worldwide or on different floors of the same building, sharing devices in real time is virtually impossible. Without an organized and instantaneous means to share devices, valuable time that could be used for testing is wasted.

While the goal for modern mobile testing is to be exceedingly faster and more agile, waiting on devices to become available or to be located is a huge time constraint and leads to major delays.

Yes, multiple devices can be purchased, and devices can be shipped, but these solutions also cost a lot of money. Even if a device is located in a locked office down the hall, physical device sharing leads to huge strains on time, cost and productivity that enterprise mobility teams cannot afford to waste.

While many testing teams have attempted solutions like using personal devices, the only reasonable way to have sufficient coverage of form factors, models,

and OS versions is to have a large pool of devices that everyone can share.

A mobile device cloud stores a shared device pool in one central location where the devices are always charged, accounted for, and are available to be reserved and scheduled by testers. With a mobile device cloud all devices are ready 24/7 for instant access anywhere in the enterprise.

#2: Device "In-Hand" vs. Interactivity

Take a moment and imagine what manual testing would look like without physical devices in testers' hands. Imagine the possibilities.

A mobile device cloud with a shared pool of devices enables testers to login to a portal from a computer. Upon login, testers can see available devices and choose one for testing. The device's display contents are shown in real time on the engineer's monitor. A mobile device cloud offers the added benefits of allowing testers to reserve devices ahead of time, allowing devices to be used more efficiently.

While manual testing without a mobile device cloud requires testers to hold a device, accessing a device via a mobile device cloud makes the device appear on screen as if the physical device were in hand. All

3

From Selenium to Appium: Mobile Web to Mobile App Testing

6

Visual Regression Testing: A Critical Part of a Mobile Testing Strategy

7

3 Strategies for an Efficient Mobile App Testing Project

9

Need A Hand? Benefits of Real-Time Manual Testing

15

Using Computer Vision to Reduce Test Automation Blind Spots

18

Should I Use a Simulator or Real Device When Dev Testing Mobile Apps?

23

Additional Resources

user interactions with the device are initiated with the workstation's keyboard and mouse.

Have you ever felt like you're "all thumbs" when manual testing? If typing on small keyboards on a physical device is a challenge, a mobile device cloud allows testers to use the workstation's keyboard to type, copy, and paste information while testing.

By helping to streamline the process of device sharing and by making it easier to test on devices from one central location, manual testing is simply faster and more efficient.

#3: Get More Done with Productivity Gains

What is real-time manual testing anyway? Testing in real-time occurs by ditching the lag and the factors that waste time and slow down testing. By taking devices out of testers' hands, manual testers can test in real-time thanks to not having to wait for physical devices to become available. A smooth, high-performing device cloud replicates the results of in-hand physical devices

and improves productivity in the following areas:

- Testers can view multiple devices side-by-side for comparison and they can save multi-touch gestures.
- Reporting bugs is faster because a mobile device cloud enables manual testers to copy and paste text and add screenshots to a defect report with a physical keyboard.
- For apps that rely on geographical location, such as navigation, manual testers can rapidly test through location simulation by changing the GPS location of a device.

In addition, when new apps are developed or when an updated app release is ready for testing, a mobile device cloud makes installing apps across devices faster. Apps can be installed in real time if the cloud is integrated with the build system so all new builds are installed automatically.

Manual Testing and The Human Touch

On further examination, let's consider deep-sixing the term manual testing in favor of recognizing the

human touch needed to get an app up and running and out the door. Unfortunately, the concept of "manual testing" is sometimes fraught—often denigrated by vendors in our space that sell automation tools and sometimes by industry analysts as well. In many circles, manual testing can have a bad reputation as being inefficient, expensive, and error prone. It certainly isn't fair, but in reality, it all comes down to the strengths and weaknesses found in both manual and automated testing.

It's also evident that most of the good stuff said about manual testing is true when describing things scripts can't do. And most of the bad stuff said about manual testing is true when describing things scripts can do. So, what's a modern-day tester to do? Before making a decision, consider the power of the "human touch" when testing mobile apps.

Implications of the Human Touch

Applying the human touch means professionals try to think like users by doing not only the expected things,

Testing in real-time occurs by ditching the lag and the factors that waste time and slow down testing.

3

From Selenium to Appium: Mobile Web to Mobile App Testing

6

Visual Regression Testing: A Critical Part of a Mobile Testing Strategy

7

3 Strategies for an Efficient Mobile App Testing Project

9

Need A Hand? Benefits of Real-Time Manual Testing

15

Using Computer Vision to Reduce Test Automation Blind Spots

18

Should I Use a Simulator or Real Device When Dev Testing Mobile Apps?

23

Additional Resources



but also the unexpected, unanticipated, “outside the box” things—in short by trying to break things. Script writers, on the other hand, often try to prove that an app is working by writing scripts that run successfully. The conundrum raised is, “But I thought the point of testing was to find bugs!” In the future, artificial intelligence (seen in nascent form, maybe in “crawler” tests that click at random) may encroach on what we’re delegating here to human beings.

But until that day comes, we will have issues that need human judgment—and the speed and convenience of a device cloud can significantly shorten the time needed to apply it. For example, designers or standards groups can quickly check apps or websites on a variety of devices to ensure they meet aesthetics and usability standards, something scripts cannot yet do. Just as important is trying out accessibility features.

A human touch is also needed to judge whether an app will enhance competitiveness and profitability, whether it is easy for people to understand and use, and to determine if the overall user experience will be positive for the company brand. These issues will likely still require the human touch even if we automate the initial bug-hunting that is done manually at present.

Give Yourself A Hand: The Future of Manual Testing

So where does the future of manual testing appear to be headed? The most effective enterprise mobility teams will embrace strategies that erase the challenges of manual testing, eliminate the lag, and transform manual testing into real-time testing, to stay competitive and to continue to deliver high-quality apps.

Sure, manual testing isn’t always practical and automated testing can’t catch everything. So how do you

know when you should use automated or manual testing? Let’s say your organization has an iOS app and Apple releases a new version of iOS (sound familiar?). The more automated test cases you already have on hand, the faster you’ll be able to pinpoint what updates need to be made.

With that in mind, does it make sense to automate all testing? Well, possibly. But, there are times when it may be best to conduct tests manually instead of relying on test automation.

So, let’s break it down and figure out when it is best to manually test.

When to Use Manual Testing: Getting It “Done-Done”

The most common use of manual app testing is during the development phases of a project, while developers are coding new functionality. Developers can quickly access devices using a mobile device cloud solution to help facilitate device management and sharing.

In the Agile world, the completion of incremental function needs quick and early validation that the code behaves the way the developer—and designer—intends. A developer can use a real mobile device to make a quick check of the code and can then hand the

3

From Selenium to Appium: Mobile Web to Mobile App Testing

6

Visual Regression Testing: A Critical Part of a Mobile Testing Strategy

7

3 Strategies for an Efficient Mobile App Testing Project

9

Need A Hand? Benefits of Real-Time Manual Testing

15

Using Computer Vision to Reduce Test Automation Blind Spots

18

Should I Use a Simulator or Real Device When Dev Testing Mobile Apps?

23

Additional Resources

managed device off for remote access by the designer or product owner.

Quickly accessing the right hardware for such confirmation makes a faster path to what Agile calls “done-done”—complete code that satisfies the user story and the product owner. The goal of manual testing in this case is to get a quick course correction to ensure that app functions are correct. There may be no return for automating test cases for code that won’t ship until it has been modified or completed.

Some mobile apps or hardware features can require manual testing. For example, you may want to integrate with a camera function such as barcode scanning. Since there really isn’t a way to simulate this function, manual testing is the best way to go.

Manual testing may also be handy for validating the externals and the appearance of the app. Manual tests can sort out whether the app has the right icon, if it handles being interrupted and then re-dispatched, and whether the overall look and feel of the app is correct.

This last dimension is truly aesthetic and takes a designer’s eye. Automation hasn’t yet progressed into the judgment realms of “cool” and “beautiful.” For the overall effect, the mind’s eye can hardly be bettered.

Finally, manual testing may be useful to verify spelling, placement of objects, proportions, consistency of UI standards, and general design standards.

Are the buttons all supposed to say “Logon” or “Log-in?” Do we “delete” or “remove?” Are the names of the states spelled correctly? Is the right version of the company logo being used and are the colors consistent with marketing’s defined palette?

These details are mostly fit-and-finish items and don’t tend to vary once they have been done correctly. Of course, if they do vary, then automated tests can be used to ensure that the spellings and terms remain correct.

Manual Testing v. Automated Testing: Which Is Best?

So, how do you know which tests are best? When you’re testing new functionality or a single function, manual mobile app testing is often the way to go. On the other hand, whenever you’re performing cross-platform tests or simple regression testing, automated mobile app tests can deliver a sound return on your investment. As with most things mobile, developing and testing mobile apps is a bit of a balancing act.

On the one hand, development and QA teams are

Automation hasn’t yet progressed into the judgment realms of “cool” and “beautiful.” For the overall effect, the mind’s eye can hardly be bettered.

tasked with quickly deploying apps that have been fully tested and perform according to plan. On the other hand, app development times can run longer than anticipated, putting increased pressure on QA to run comprehensive functional and regression tests across all device types and operating systems with far less testing time than originally planned. This puts extra pressure on the QA organization since they still need to keep a close eye on budgets, app backlogs and manage high expectations for app acceptance and usability.

It should come as no surprise that enterprise QA teams are also trying to determine if manual testing alone is the best approach. The very nature of mobility, with rapid changes and multiple form factors,

3

From Selenium to Appium: Mobile Web to Mobile App Testing

6

Visual Regression Testing: A Critical Part of a Mobile Testing Strategy

7

3 Strategies for an Efficient Mobile App Testing Project

9

Need A Hand? Benefits of Real-Time Manual Testing

15

Using Computer Vision to Reduce Test Automation Blind Spots

18

Should I Use a Simulator or Real Device When Dev Testing Mobile Apps?

23

Additional Resources

Just because automated mobile app testing is a possibility, it doesn't mean that it's always the best fit for all enterprise testing needs.

paired with the pressure to quickly deploy numerous mobile apps, makes automation a popular test strategy for mobile applications.

Organizations can quickly realize value from an automated mobile application testing approach because it can minimize the time it takes to get apps ready for release. However, just because automated mobile app testing is a possibility, it doesn't mean that it's always the best fit for all enterprise testing needs. Mobile apps come with a lengthy list of outside variables. Until testers have isolated and addressed these variables, manual testing still has its place in the testing process.

Perhaps the best option is a hybrid approach. In order to find that balance between manual and automated

testing—both for software and mobile applications—consider the following key concepts:

- Testing new functionality? Use manual testing. If an app contains new functionality, it should be tested manually. With new functionality, testers won't know what type of automated test script to write prior to performing a manual test.
- Only testing once? Make it manual. Obviously, if only one function of one mobile app is being tested, it doesn't make sense to spend the time and energy to create an automated testing script. Although manual testing may take longer than running an automated test, script creation takes time. If there isn't a case for re-use, there is really no need to tie up automation experts with script creation for a single test case.
- What about regression tests? They are good candidates for automated testing. Each time a developer releases new code, a regression test is needed. Because regression tests are repeated, automated testing is a perfect fit. While there is some time needed on the front end to develop test scripts, testers will save time because they won't have to start from scratch with the test each time.
- What about automated tests? Automation is essential for complex tests. Complex tests have multiple components that need to be tested at once. After testers familiarize themselves with the app's func-

tionality, an automated test script can help them run through a variety of tests quickly to speed the testing process.

The Human Touch vs. Test Automation

If manual testing aims to show that an app is correct and ready in the first place, what's the use of a script that shows it still works? In a word: regression. Scripts that run to perfection can be an invaluable and efficient means to see if recent changes to the app have introduced bugs. The human touch helps answer, "Does this new thing work?" Automation is able to efficiently answer the question, "Does it still work? Especially after we 'fixed' it?"

Trying to find new bugs with scripts is a misapplication of automation and argues for the necessity of manual testing. The "human touch" finds new bugs in new code by trying both the expected and the unexpected and must assess suitability for both users and the business.

Trying to perform regression testing manually is a misapplication of manual testing and argues for the necessity of automation. Manual regression is inefficient, error-prone, and consumes massive amounts of resources over and over with inconsistent and often unreliable results.

3

From Selenium to Appium: Mobile Web to Mobile App Testing

6

Visual Regression Testing: A Critical Part of a Mobile Testing Strategy

7

3 Strategies for an Efficient Mobile App Testing Project

9

Need A Hand? Benefits of Real-Time Manual Testing

15

Using Computer Vision to Reduce Test Automation Blind Spots

18

Should I Use a Simulator or Real Device When Dev Testing Mobile Apps?

23

Additional Resources

Using Computer Vision to Reduce Test Automation Blind Spots

By Maxim Chernyak

Artificial intelligence and machine learning are trending in the IT industry, as they perform complicated tasks while excluding the human factor.

The adoption of AI and ML in the QA sphere is growing, too. Developing automated tests, analyzing reports, and automatically fixing tests after changes in UI can highly streamline testing and save time.

Computer vision (CV) is a part of computer science closely related to AI and ML. It replicates the human eye using deep learning technology and can determine objects in pictures, which helps machines orient in space and perform repetitive detection tasks.

Some test automation issues cannot be resolved using an ordinary toolkit. This is where CV can serve the right purpose. Let's look at the cases where CV can enrich the test automation process.

Mobile Automation

The most common tool for mobile automation is



Appium. A single Appium framework is suitable for regular mobile software products, but it fails to detect the elements in content-based apps, such as mobile games. While human eyes can detect all buttons and fields, Appium can only recognize the black screen.

Introducing CV to Appium mobile automation solves this problem. You can train your CV algorithm to

recognize the elements of the input data via a screenshot, get their coordinates, and use Appium to interact with them, and you are good to go.

Desktop Automation

Testing desktop applications can be challenging regarding their automation. Selenium WebDriver works only with web apps, whereas for desktop ones, you

COPYRIGHT 2020 **15**

3

From Selenium to Appium: Mobile Web to Mobile App Testing

6

Visual Regression Testing: A Critical Part of a Mobile Testing Strategy

7

3 Strategies for an Efficient Mobile App Testing Project

9

Need A Hand? Benefits of Real-Time Manual Testing

15

Using Computer Vision to Reduce Test Automation Blind Spots

18

Should I Use a Simulator or Real Device When Dev Testing Mobile Apps?

23

Additional Resources

If the application is complex, it can be time-consuming to test all the components on the pages manually and detect all the issues and updates.

can use AutoIT for Windows or AppiumForMac for OSX software products.

However, if your application has been developed to cater to different systems, writing multiple scripts with multiple technologies is time-consuming. To overcome this challenge, SikuliX can automate anything you see on your desktop computer running Windows, MacOS, or some Linux or Unix. Powered by OpenCV library, this tool allows you to click on the elements based on how they look on the screenshot and compare the actual result with the desired outcome.

Automation of a Non-Standard Device

Usually, the UI test automation scope is limited to web, mobile, and desktop applications. These options have powerful drivers and tools to identify elements and operate on them.

However, there are some non-standard platforms we may also want to automate, such as game consoles, additional screens for devices, and internet of things technologies. Whenever you are unable to retrieve the state and location of the elements with the help of the standard toolkit, CV can create the element list based on the screenshot or photo of the display provided. Investigate how you can interact with the areas of the screen using coordinates and how this can resolve the testing challenge.

UI Change Detection

Some applications' UIs rely on a set of predefined components known as a UI constructor, which simplifies the development process, making the design more consistent. At the same time, this approach adds a dependency: You need to track the common components' versions and make sure your app uses the most relevant ones. If the application is complex, it can be

time-consuming to test all the components on the pages manually and detect all the issues and updates.

This is where CV can help. Introduce the image checks to your automation process by uploading the sample page UI version, and compare the actual page look to this template, having your tests fail if the UI does not match. This will help you track changes in the design and with content updates.

These checks need not be performed during every test execution; a true/false flag can configure whether this test run considers the UI change as a bug or not.

Computer Vision Tools for Automation

OpenCV is an open source library intended to process images that can be used in multiple languages, including Java, Python, and Ruby.

3

From Selenium to Appium: Mobile Web to Mobile App Testing

6

Visual Regression Testing: A Critical Part of a Mobile Testing Strategy

7

3 Strategies for an Efficient Mobile App Testing Project

9

Need A Hand? Benefits of Real-Time Manual Testing

15

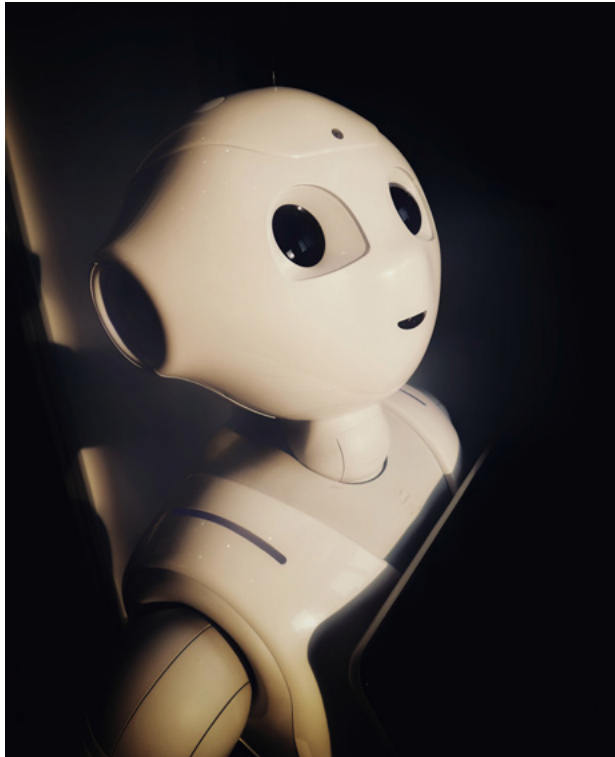
Using Computer Vision to Reduce Test Automation Blind Spots

18

Should I Use a Simulator or Real Device When Dev Testing Mobile Apps?

23

Additional Resources



This library identifies the elements on the image and interprets them as an element tree suitable for test automation. After the areas of interaction are detected, engineers can get their coordinates and operate on them using Appium or Selenium WebDriver.

SikuliX is a tool for Mac, Windows, and Linux designed to search for a particular element on the screen based on an image. It is developed using OpenCV algorithms and supports multiple languages. It also has Sikuli IDE, which is easy to use and does not require in-depth automation knowledge.

This tool can help automate desktop applications and flash web applications or mobile games when paired with Selenium WebDriver or Appium. SikuliX searches for the area on the screen and operates on the element found.

Tesseract OCR is a program for text recognition and is developed for Mac, Windows, and Linux. With the help of neural networks, Tesseract OCR recognizes the text on the image and can be extended to recognize custom fonts and text sizes. Introducing Tesseract OCR to web automation makes validating outcomes possible when the result of an operation is displayed on an image instead of being, for example, a text field. It can also be embedded into SikuliX to validate the expected result when using image-based UI automation.

Is Computer Vision a Unified Solution?

It is quite complicated to introduce CV to test automation projects. CV needs additional research, data sets to train the algorithms, multiple tool dependencies,

specialized testing skills, and strong monitoring of the results at the first stage. Besides, CV algorithms are not 100% accurate, especially when the image quality is low.

CV is a new tool in software development and test automation, and there are some cases where you should think twice before applying this technology to your testing project:

- If you have web and mobile applications with accessible elements and can write stable locators for them, implementing CV may be too challenging, fruitless, and expensive compared to the regular web and mobile UI automation
- If your application has a frequently changing UI and the components of your application look different at the end of each sprint, CV may be the wrong choice—you will see thousands of failed tests, spend lots of time uploading new versions of the design to the scripts, and will not get the desired outcome
- If image quality is low, you will get inaccurate results

Implementing CV requires additional effort, customization, and engineer expertise. However, it can be helpful when you need to work with image-based content, get text from image elements, and detect changes in your UI. CV can be a good way to streamline your test automation project and resolve specific QA tasks.

3

From Selenium to Appium: Mobile Web to Mobile App Testing

6

Visual Regression Testing: A Critical Part of a Mobile Testing Strategy

7

3 Strategies for an Efficient Mobile App Testing Project

9

Need A Hand? Benefits of Real-Time Manual Testing

15

Using Computer Vision to Reduce Test Automation Blind Spots

18

Should I Use a Simulator or Real Device When Dev Testing Mobile Apps?

23

Additional Resources

Should I Use a Simulator or Real Device When Dev Testing Mobile Apps?

By Mobile Labs

There never seems to be a shortage of components and features to test in enterprise mobility. From testing code in the development phase, to testing functionality, to testing device components and operating systems, mobile developers, testers and quality assurance teams have their work cut out for them. In response, many enterprise mobility professionals rely on simulators or emulators to expedite the testing process.

While testing on a real device, in users' hands, is the best way to get the comprehensive picture of how an app or mobile website will behave in the real world, many teams may not have the resources in place to completely rely on testing on real devices. This is where simulators and emulators are effective because they can enable busy mobility professionals to test certain elements quickly, even if they do not have access to the actual device.

Simulators and Emulators: What Is the Difference?

Although these terms are often used interchangeably, there is a marked difference between a simulator and

an emulator. Below are the traditional definitions of simulators and emulators normally found in the technology world.

A *simulator* is a program that enables a computer to execute programs written for a different operating system. While an *emulator* is hardware or software that allows one computer system (known as the host) to behave like another computer system (referred to as the guest). In this host and guest arrangement, an emulator allows the host system to run software or use peripheral devices, such as printers or USB devices, that are designed for the guest system.

But, for enterprise mobility professionals, the definitions of simulators and emulators are different based on whether you are dealing with Apple or Android.

Apple uses the term simulator to describe their program that enables enterprise mobility professionals to run and test mobile apps on a computer without



having to use a real device. Android uses the term emulator to describe their version of this program. Therefore, in this blog the term simulator will be used when discussing Apple, while emulator will be employed when discussing Android.

Why Use Simulators or Emulators?

In a perfect world, with unlimited resources, testing

COPYRIGHT 2020 **18**

3

From Selenium to Appium: Mobile Web to Mobile App Testing

6

Visual Regression Testing: A Critical Part of a Mobile Testing Strategy

7

3 Strategies for an Efficient Mobile App Testing Project

9

Need A Hand? Benefits of Real-Time Manual Testing

15

Using Computer Vision to Reduce Test Automation Blind Spots

18

Should I Use a Simulator or Real Device When Dev Testing Mobile Apps?

23

Additional Resources

For busy enterprise mobility teams, simulators and emulators make it easy for developers and testers to quickly mock up an application.

on real devices is preferable. Although Apple Simulator and Android Emulator are powerful, robust tools, they do not completely replicate the full experience of using an actual device because they do not take into account hardware elements such as the battery, CPU, networking, etc.

Below are a few reasons why enterprise mobility teams may choose to use an Apple Simulator and/or an Android Emulator as a substitute to testing on a real device.

#1: Not every organization can afford multiple real devices

With so many different devices on the market today, organizations sometimes cannot afford to buy and

manage multiple versions of costly devices for developers and testers. Without enough real devices to go around, it is more cost-effective, not to mention efficient, for team members to use simulators and emulators for testing.

#2: Ability to mock up an application

For busy enterprise mobility teams, simulators and emulators make it easy for developers and testers to quickly mock up an application. In the mock up, quick tests can be conducted, such as testing different screen sizes to verify layouts, without having to have a real device with each screen size for testing.

#3: Easy debugging/testing before code is committed to source and deployed

For mobile developers, simulators and emulators enable teams to quickly verify the application state when certain UI events are triggered. Running these tests on a simulator or an emulator enables teams to make any adjustments to code before committing it to the source code or to actual deployment.

#4: Ability to run unit tests

By using a simulator or an emulator, testers can run unit tests in a build system without having to use a real device.

#5: Browser simulation with Google Chrome for mobile web

For developers and testers working with mobile websites, using a simulator enables the use of Google Chrome for browser simulation. With this simulation, testers can see how a page renders by screen size and can quickly make code changes and refresh the page to check the rendering. As an added benefit, the browser simulation with Google Chrome still enables teams to access the Chrome developer tools.

Comparison: Apple Simulator v. Android Emulator

For mobility teams working with both Apple and Android, there are some key factors to consider when using Apple Simulator or Android Emulator.

First, when working with iOS it is important to note that Apple's code is compiled differently. Therefore, an app built for a simulator will not run on a real device as the architectures for the simulator are built for the Macintosh chips and the real devices are built for ARM chips.

3

From Selenium to Appium: Mobile Web to Mobile App Testing

6

Visual Regression Testing: A Critical Part of a Mobile Testing Strategy

7

3 Strategies for an Efficient Mobile App Testing Project

9

Need A Hand? Benefits of Real-Time Manual Testing

15

Using Computer Vision to Reduce Test Automation Blind Spots

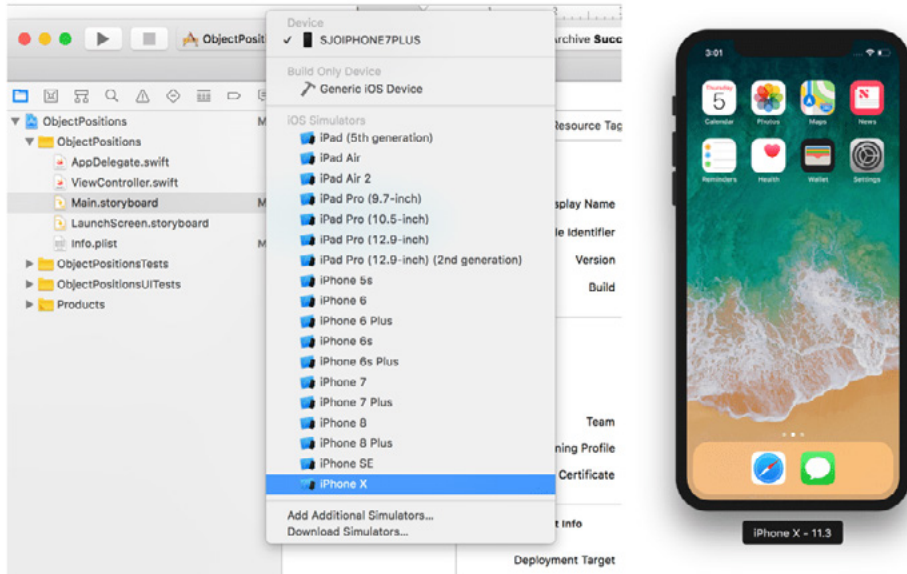
18

Should I Use a Simulator or Real Device When Dev Testing Mobile Apps?

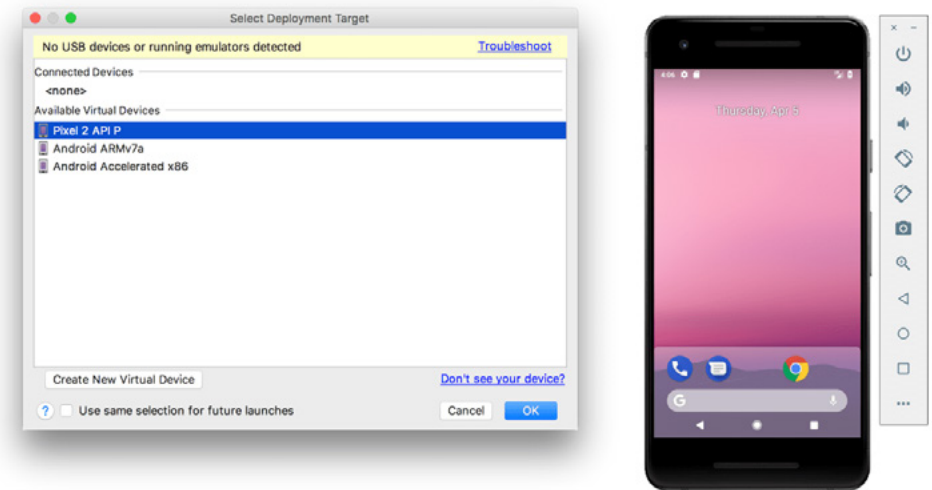
23

Additional Resources

Developers and testers should run the app from Xcode and choose a simulator from the dropdown menu located by the run button, as seen below.



For developing and testing for an Android device, developers and testers should run the app from Android Studio by hitting the run button. Next, choose the configured Android Virtual Device in the "Select Deployment Target Window," as seen below.



By using a simulator or an emulator, testers can run unit tests in a build system without having to use a real device.

3

From Selenium to Appium: Mobile Web to Mobile App Testing

6

Visual Regression Testing: A Critical Part of a Mobile Testing Strategy

7

3 Strategies for an Efficient Mobile App Testing Project

9

Need A Hand? Benefits of Real-Time Manual Testing

15

Using Computer Vision to Reduce Test Automation Blind Spots

18

Should I Use a Simulator or Real Device When Dev Testing Mobile Apps?

23

Additional Resources

Real Device Testing = Real Results

Although Apple Simulator and Android Emulator can help developers and testers quickly test certain elements of mobile apps and mobile websites before deployment, the majority of important tasks and tests are best suited to be tested on a real mobile device.

Consider this excerpt from Apple's Simulator Help Overview (emphasis added):

"Simulator is a great tool for rapid prototyping and development of your app allowing you to see the results of changes quickly, debug errors, and run tests. It is also important to test your app on physical devices as there are hardware and API differences between a simulated device and a physical one. In addition to those differences, Simulator is an app running on a Mac and has access to the computer's resources, including the CPU, memory, and network connection. These resources are likely to be very different in capacity and speed than those found on a mobile device requiring tests of performance, memory usage, and networking speed to be run on physical devices."

Source: <https://developer.android.com/studio/run/device.html>

Next, consider this quote from the Android Studio User Guide (emphasis added):

"When building an Android app, it's important that you always test your app on a real device before releasing it to users."

Source: <https://developer.android.com/studio/run/device.html>

Despite having built a powerful simulator (Apple) and an emulator (Android), both vendors still advocate for testing on a real device. Clearly, while simulators and emulators have their role in the mobile development and testing process, they should not be considered a replacement for testing on a real device.

For real results and to get the most accurate insight into how an app or mobile website will function in the real world, enterprise mobility teams should test on real devices for the following reasons:

#1: Testing on a real device gives a true end-user experience

As Apple noted in its Simulator Help Overview, there will be differences between a computer's CPU, memory and network connections and those of a mobile device. Because a mobile app or mobile website uses the device's CPU, memory and networking, running tests on a simulator or an emulator will not accurately predict the effect of these device elements on the mobile experience.

#2: Influence of manufacturer on Android devices

Because Android runs on several devices from different manufacturers, there can be different modifications by each vendor that Android Emulator may not be able to effectively test. In this situation, enterprise mobility teams should use real Android devices, running a real OS across different manufacturers for the best results.

Android Operating Systems are also different between manufacturers. These different Operating Systems are not loaded into an Android Virtual Device, making testing just with an emulator virtually impossible.

#3: Use the real device hardware features for testing

The best way to effectively test hardware features such as the device's camera, GPS, battery, etc., is on the actual device itself. A simulator or emulator cannot predict the behavior or replicate these features accurately.

#4: Certain tests are best suited for real devices

Most tests are best suited for testing on real devices as simulators and/or emulators cannot give accurate results. Examples of tests that are best suited to a real device fall into the following categories:

- Tests requiring hardware features (touch sensors, GPS, camera, etc.)

3

From Selenium to Appium: Mobile Web to Mobile App Testing

6

Visual Regression Testing: A Critical Part of a Mobile Testing Strategy

7

3 Strategies for an Efficient Mobile App Testing Project

9

Need A Hand? Benefits of Real-Time Manual Testing

15

Using Computer Vision to Reduce Test Automation Blind Spots

18

Should I Use a Simulator or Real Device When Dev Testing Mobile Apps?

23



Additional Resources

- Tests requiring different OS versions
 - Especially important for Android due to multiple device manufacturers
- Functional tests
 - All regression tests should mimic real world conditions on real devices
- User acceptance tests
 - Real devices enable teams to make sure that what is delivered is what is expected by the end-user
 - Testing colors of the app on real device screens with different brightness settings gives the best results

Even though testing on real devices can be daunting based on the volume of devices and different operating systems available, ultimately enterprise mobility teams need to provide the best and most innovative mobile experiences to users. Real device testing allows mobile developers and testers to produce high-quality app experiences that are required to keep up with digital transformation. It is an investment worth making for quality results.

If you still aren't sure whether a specific test or activity is best suited to be tested on a real device vs. a simulator or emulator, use this [checklist from Mobile Labs](#) as

a guide to determine what tests should be conducted using a real device or when a simulator or emulator can be an appropriate substitute.

Should I Use a Simulator/Emulator or a Real Device When Developing/Testing Mobile Applications?		
	 SIMULATOR/EMULATOR	 REAL DEVICE
I am doing a mock-up of my application	✓	
I need to debug my application	✓	✓
I want to test my application code	✓	
I need to test end-to-end flows of my application		✓
I need to verify the networking aspects of my app		✓
I want to verify my app performance (CPU/Memory)		✓
I want to know what happens when my app is interrupted by notifications or phone calls		✓
I want to verify the camera functionality within my app		✓
I need to verify app graphics and transitions	✓	✓
I need to verify the app from an end user perspective, such as testing screen brightness, resolution, etc.		✓
I need to make sure my app runs on various devices and OS versions		✓
I want to run regression tests against my app		✓
I want to test the battery life, or how fast the battery drains, based on usage of my app		✓
I need to test geolocation and screen/app orientation (accelerometer) functionality		✓

Additional Resources

MORE INFORMATION FOR SOFTWARE PROFESSIONALS



**ROI
CALCULATOR**

What's the ROI of Parallel Testing Your Mobile Apps?



INFOGRAPHICS

**Accelerating Mobile App Transformation
10 Tips to Build a Successful Mobile Testing Lab**



SURVEY REPORT

Automated Testing: How To Accelerate Mobile App Transformation



WEBCAST

5 Things to Consider When Implementing a Mobile Device Cloud

CONTACT MOBILE LABS

info@mobilelabsinc.com | +1 (404) 214-5804 | www.mobilelabsinc.com

COPYRIGHT 2020 **23**

3

From Selenium to Appium: Mobile Web to Mobile App Testing

6

Visual Regression Testing: A Critical Part of a Mobile Testing Strategy

7

3 Strategies for an Efficient Mobile App Testing Project

9

Need A Hand? Benefits of Real-Time Manual Testing

15

Using Computer Vision to Reduce Test Automation Blind Spots

18

Should I Use a Simulator or Real Device When Dev Testing Mobile Apps?

23

Additional Resources