uTest®
Where the best test

8 Tips for Android App Testing

Must-Know Tips for Achieving Android App Success

White Paper
June 2012

White Paper

8 Tips for Android App Testing

Must-Know Tips for Achieving Android App Success

Table of Contents

> **"Android was built from the ground-up to enable developers to create compelling mobile applications that take full advantage of all a handset has to offer."**
>
> - Open Handset Alliance

Introduction

Achieving Android App Success

Android app testing is complicated by the fact that it has – without debate – the most complex array of handsets, versions and carriers of any mobile platform available. And unlike more closed systems, each Android device presents its own set of challenges. But being challenging is no excuse for limited or poor testing. Despite encompassing a large number of devices, there *are* a few things that can and should be tested across the field. It's especially important to test Android applications in-the-wild and on as many devices as possible, because something that works perfectly on one device might cause a bug on another. Your end users will use a variety of phones, so your app needs to work consistently on a variety of phones. Rules are rules.

To help you achieve that cross device consistency, we've detailed eight focus areas that are essential to Android success. This whitepaper will highlight tips for testing within the Android Matrix, specific issues to test for and, finally, how to get testing done.

The Android Matrix

The Android testing matrix is the largest, most complex of all the mobile testing matrices. Android is notorious for fragmentation because its open nature allows more handset manufacturers and models, carriers, UI customizations and other factors to be introduced into the ecosystem. But being vast is no excuse for not understanding and testing across the matrix.

> **"The top 25 devices don't even encompass half of the map, and would still exclude well-known and popular (if old) models like the Samsung Nexus S. The developers would be missing out on over 50 percent of the Android market."**
>
> \- Casey Johnston
> Associate Writer, ARS Technica

The most successful apps test across as many handset and platform versions as possible. Take, for example, Hong Kong developer Animoca, who has seen upward of 70 million downloads. A recent TechCrunch article reported that the company actively tests each release on 400 different devices to ensure widespread success. Ignoring the matrix can only end in app failure – so when you're testing be sure to address these key matrix elements.

1. Handsets and Carriers

The most obvious part of the Android matrix is the sheer number of devices sporting the operating system. Complicating this ever-growing figure is the number of handset manufacturers and carriers that participate in the Android universe.

According to the official Android website, Android devices are available in 25 countries. Worldwide, 23 manufacturers produce Android phones and 63 carriers support them on their networks. Globally, there are roughly **250 officially recognized Android handsets** currently on the market (not taking into account platform versions or custom skins). Narrow the scope to the United States only and there are still around 100 different devices produced by 15 manufacturers and supported by seven carriers. Nearly 20 of these devices include a physical keyboard, while the rest are touchscreen. And don't forget the Samsung Galaxy Note, which not only has a unique screen size, but also has a stylus (unlike any other Android device). Because each device has its own specs – encompassing physical design and custom UI attributes – testing coverage should include as many devices as possible.

Bear in mind that Android apps can be accessed on even more devices, including on the cheaper "low-end" phones that are beginning to appear in many markets. These phones can only support non-data-intensive apps and require a completely separate round of testing. Many developers have not begun addressing this new category of phone, but keep an eye on the trend.

2. Screen Size and Density

In comparison to the extremely controlled nature of iOS, the combination of screen sizes and screen densities in the Android universe add an extra challenge to app testing. Helpfully, and despite there being more than 200 distinct devices, Android classifies all official devices into one of four screen sizes and one of four screen density classes.

- Screen sizes: Small, Normal, Large, Extra Large
- Screen Density: Low dpi, Medium dpi, High dpi, Extra High dpi

Since testing on all devices is nearly impossible, these classifications will give testers and developers a good idea of how the app will appear on devices within the same screen size/density category. A recent breakdown of active devices looks like this:

	Low dpi	Medium dpi	High dpi	Extra High dpi
Small Screen	2.3%		2.4%	
Normal Screen	.7%	26.2%	57.8%	.9%
Large Screen	.3%	2%		
XL Screen		7.4%		

As you can see, the most popular devices are squarely in the normal screen size range and generally sport either high screen density or medium density. This data is updated fairly frequently on the Android Developers website, so check it often as more devices hit the market.

3. Platform Versions

Android supports and tracks ten platforms/versions ranging from “Cupcake” 1.5 to “Ice Cream Sandwich” 4.0.4 (at the time of publication). Not all devices support all platform versions and new versions are not released to all handset makers at the same time. Instead, new releases are dripped out in increments, often leaving users eagerly waiting.

Despite being the most recent version, Ice Cream Sandwich is only the third most used platform and still has not been rolled out to all devices. “Gingerbread” 2.3.3 continues to dominate Android devices with 64% of use and “Froyo” 2.2 comes in second at 19%. Ice Cream Sandwich comes in third by only 1% (over “Éclair” 2.1).

By testing an app exclusively on the latest version you will isolate an extremely high number of users. Conversely, if you do not update an app to work consistently on newer releases you may lose current users as they upgrade. With so many platforms present on Android devices it’s important to test apps on at least the top three versions – if not the top four or five. To pinpoint the most popular versions visit Android Developers Platform Versions page, which tracks devices active in Android’s Google Play market.

What to Test

As with all mobile app testing, common functions should be tested on as many devices as possible to ensure consistency. However, Android’s device and platform combinations present new challenges – a feature that functions perfectly on one device may cause a bug on another. In addition to normal testing considerations, there are a number of recurring issues that commonly crop up on a variety of Android devices. These issues are prevalent enough that they should be added to test cases on as many devices as possible.

4. Common Considerations

These functions may seem like no-brainers when it comes to testing, but skipping one can spell doom for an app.

- **Registration & Login**: Testing should make sure the registration and login processes are intuitive, easy to complete and functional from start to finish. It is important to have testers on a variety of devices complete the entire registration and log-in process to ensure everything runs smoothly and no error messages occur.
- **Menu Options**: Often times, menu options can be difficult to access and decipher. Make sure that menu items like Help, About, etc. are easy to find and navigate. This is especially important to test on a variety of screen sizes and with real users, since fingers are much larger than a mouse on an emulator.
- **Keys**: Any problems related to scrolling, text selection, the back button, etc. are bound to lead to trouble, so make sure your key functionality is

clear and consistent. Also, be sure to check that the app will function consistently using both a physical keyboard and touchscreen.

- **Interruptions**: How does the app behave when the device battery is at full strength, medium strength and low strength? What about if the user gets an incoming call or text? If there is another app running in the background? These are all real life scenarios that users are going to encounter. Don't let them take down your app. (Crash logs are particularly helpful in diagnosis if other events are adversely effecting an app.)
- **Error Messages**: Your error messages should be clear, concise and actionable. "Error Red Marlin 12a26q" may make sense to the developers, but it doesn't help users know what went wrong or how to fix it. Make your error messages easily understandable and you're a step ahead of virtually all mobile applications on the market today.
- **Landscape v. Portrait:** An app's functionality and usability should not be affected when changing from portrait to landscape mode. Test to be sure buttons, fields and menu options are easily accessible and functional in both situations.
- **Settings:** Change a device's settings and repeat necessary tests to ensure an end-user's custom settings won't affect the app's performance.

Also be sure to check an app's effect on battery life. Many users expect a phone's battery to last an entire day, at minimum. And with phones performing more and more tasks, battery life is stretched ever thinner. If an app sucks more than its fair share of power it will be ditched by users. To be sure an app isn't a power hog, Sachin Date recommends both a normal use test and an idle use test.

- **Normal use test:** Start on a full battery and use the application for 6-12 hours and measure the battery level at the end of each ½ or 1 hour. You may use an automated testing tool to do this so as to keep the test running for the required time interval. This test will tell you how quickly your application drains the battery when in 'normal' use, with all the foreground and background features of the application running normally.
- **Idle run test:** Turn off the screen lock and power saver modes. Then start on a full battery and keep the application running on its main, home or dashboard screen as appropriate, and measure the battery level at ½ or 1 hour intervals. This test will measure the battery drain due to such things as intentional or unintentional automatic screen refreshes, and due to the background threads or services running in your application.

5. Common Issues

In addition to the regular testing considerations, a handful of issues pop up across the Android world more regularly than we'd like. Be sure to test for these common bugs across multiple devices pre-launch.

- **Special Characters:** If the app includes a search field or data entry form, test its special characters compatibility. Depending on the programming language, special characters can cause the field to choke. This is especially important if the app is going to be used internationally or with a native language that includes special characters (such as Spanish).
- **Long Strings:** Long strings of characters are more of a fringe use case. Nonetheless, it is important to make sure an app can handle at least moderate length strings of characters. Let the type of data field – and the assumed typical entry – dictate an acceptable length.
- **Tap and Hold:** Even if an app isn't designed to support the long-hold copy and paste function or the tap and hold move function, make sure it isn't confused by those actions either. Even if users don't perform these functions intentionally, it's very possible that they'll get distracted and accidentally hold the screen for too long. You don't want the app to freeze or crash as a result of this real-world scenario.
- **Virtual Keyboard:** The majority of Android devices are touchscreens with virtual keyboards. If a user accidently – or intentionally – raises the keyboard on their device the app screen can distort or, worse, the app itself can become unusable.

These are a handful of common bugs Android users encounter every day, so being sure your app isn't tripped up by them before launch can give you a leg up on the competition. When testing, remember to think like an end user and test how the app responds to potential real-world situations. Even if an app isn't designed to support a function doesn't mean a user won't try – and be angry if the app crashes.

6. Android Security

> **"Software can be correct without being secure. Indeed, software can meet every requirement and perform every specified action flawlessly yet still be exploited by a malicious user."**
>
> - James Whittaker
> Partner Development Manager,,
> Microsoft

It's no secret that Androids are susceptible to malware – largely because of the open nature of Google Play and the presence of unmonitored, third party app markets. Couple the consistent malware reports with users' increasing interest in privacy, and an accidental security slip up can be disastrous to your app's success. At the very least, be sure an app successfully accomplishes these six key security factors:

- **Confidentiality**: Does your app keep your private data private?
- **Integrity**: Can the data from your app be trusted and verified?
- **Authentication**: Does your app verify you are who you say you are?
- **Authorization**: Does your application properly limit user privileges?
- **Availability**: Can an attacker take the app offline?
- **Non-Repudiation**: Does your app keep records of events?

It is also helpful to have white hat security experts attempt to manipulate at least the most common security vulnerabilities, such as accessing data due to unsafe storage or transmission practices, cracking inadequate encryptions and unlocking hardcoded passwords. If an app is easily hacked it probably will be hacked.

Finally, test an app's access to device APIs (such as contacts, photos, camera or GPS). According to the tenants of the Open Handset Alliance (which Android is a member of), "An application can call upon any of the phone's core functionality such as making calls, sending text messages, or using the camera, allowing developers to create richer and more cohesive experiences for users." However, when stories about Path "secretly" accessing users' contacts sparked insight into this common practice, users started becoming more concerned with apps accessing unnecessary personal data. To avoid incurring user ire, test that an app clearly prompts users to grant access permission during download or launch. With so much competition in the app market, it is easy for users to find a replacement that they feel does not unnecessarily invade their privacy.

Getting Testing Done

There are a variety of tools available to help testers be more efficient and capture useful, actionable data. Not all tools work on every Android device, but there is a tool available for each general function on just about every device.

7. Helpful Tools

The best place to start is by downloading the **Android SDK** (for Windows, Mac and Linux), which offers a range of tools that will work on any Android device. The SDK will allow you to connect your device to a computer to take video and screenshots and pull advanced logs for more in depth reporting. While the SDK may seem intimidating, its video and screenshot capabilities are extremely useful.

If you prefer to keep testing central to a mobile device there are tools to accomplish many of the same tasks. Many newer devices have built-in **screen capture** capabilities. To find out if your device has a screenshot function, search the name of your device + screenshot. If it is a supported feature, this search will turn up the button combination that will produce a screenshot. If your device does not have a built-in screenshot capability, third party screen capture apps are available that do not require you to root your phone. A Google Play search for "screenshot tool" returns both free and paid screen capture apps. Before selecting one, be sure to carefully read the description and the reviews to ensure it's what you want and will work on your specific device. Remember, just because your Android meets the platform version specifications does not mean the app will work well on your handset, the reviews are the best place to find this information.

Finally, in addition to the SKD, there are several **log apps** that will allow you to pull device logs. Device logs track all applications running on a device at a specific moment.

They are extremely helpful in diagnosing crash causes or pinpointing applications that adversely affect the app you are testing. Logs can be used to track crashes, sluggishness, non-responsiveness, non-functioning buttons, non-fatal error messages and other issues. Attaching log information to bug reports will help developers address the correct issue. aLogcat is an extremely popular log app, but there are many to choose from. Once again, do your homework prior to choosing an app to ensure it will work on your device.

There are also a growing number of apps to help monitor **RAM and data usage**. Remember, users will quickly dump apps that slow their connection, take up too much space, are sluggish or zap their data. With these apps becoming more common, everyday users may start adopting them for personal use to monitor (and cull) their app collections. Stay ahead of the curve by testing apps for excess data and memory waste.

8. Useful Insights

Testing is an art, so everyone does it a little differently. Still, the best testers aren't afraid to learn new things and pick up new pointers. Here are a few tips to help you test and to keep you in the mind frame of the end user.

When capturing videos of an app under test, it's useful to hold each action and leave each screen displayed a little longer than you would in normal use. This makes the video easier for test managers and developers to follow and possibly pinpoint which action went wrong.

Digging through Google Play reviews reveals which issues users hate the most. Here's what a recent look tells us:

- 40% complain about installation
- 16% complain about performance
- 11% write about app crashes
- 3.5% report hangs or freezes
- 2% complain about the UI
- 1% had security or privacy issues

> **"You can launch a beautifully designed native application, but if it crashes, then it will receive a poor rating and users will go elsewhere. Anyone can read your app store rating. *There's no way to hide poor quality in the world of mobile*."**
>
> \- Michael Croghan
> Mobile Solutions Architect,
> USAToday

Bad reviews can kill an app before it even gets off the ground. Thorough testing of these six known problem areas can help you avoid poor reviews.

Emulators are helpful for early stage functional testing, but it is extremely important to test apps on real devices. A keyboard and mouse cannot adequately simulate touch screen usability. And some features, such as accelerometer response or location mapping, simply cannot be tested on a stationary emulator.

Though testing on Android presents a bigger challenge than most operating systems, it is not going away or simplifying any time soon. By knowing the challenges of Android testing, you can adequately address known issues, launch better apps and keep all your users – no matter what device or platform version they're on – happy and satisfied.

****

About uTest

uTest provides in-the-wild testing services that span the entire software development lifecycle – including functional, security, load, localization and usability testing. The company's community of 60,000+ professional testers from more than 190 countries put web, mobile and desktop applications through their paces by testing on real devices under real-world conditions.

Thousands of companies -- from startups to industry-leading brands – rely on uTest as a critical component of their testing processes for fast, reliable, and cost-effective testing results.

More info is available at **www.utest.com** or **blog.utest.com,** or you can watch a brief online demo at **www.utest.com/demo**.

uTest, Inc.
153 Cordaville Road
Southborough, MA 01772
p: 1.800.445.3914
e: info@utest.com
w: www.utest.com