The Essential Guide to iOS App Testing

Tips for Testing and Launching High-Quality iPhone & iPad Apps

WHITEPAPER

APPLAUSE
WINNING APPS GET IT

Introduction

Let's be honest. Saying you're "creating a mobile app" is just about as generic as saying you need to breathe gas. The reality is you need to breathe a very specific gas if you want to keep breathing. The same goes for the mobile app ecosystem. Creating a winning app for iOS is very different from creating a winning Android app. Whether you're developing for iOS, Android or another platform, you will uncover unique design and system advantages and disadvantages. These quirks, features, interactions and user expectations will dictate your course every time. While there are plenty of general tips and tricks for launching great mobile apps, this whitepaper will take a more in-depth look at what you need to know to produce a winning iOS app.

> *"If making it open would give us not the quality of Apple products all working together like they do, I would say keep it closed."*
>
> -Steve Wozniak
>
> Co-Founder, Apple

The iOS Matrix

When compared to other mobile operating systems, the "closed" nature of Apple has some advantages when it comes to testing. Having one

manufacturer control all models of iOS devices drastically limits fragmentation, unlike the Android matrix. The major Apple devices associated with app use are the iPhone, iPad and iPod Touch. However, as the iOS ecosystem begins to age, fragmentation is creeping in:

- *Screen: There are currently seven unique screens to develop for with iOS: Non-Retina iPhone/iPod; non-Retina iPad; non-Retina iPad Mini; Retina iPhone/iPod; the taller Retina iPhone/iPod 5+; Retina iPad and Retina iPad Mini*
- *UI: Since Apple controls all iOS devices there is a single UI to plan for. However, with the release of iOS 7, older devices (without Apple-designed processors) are being left behind. iOS users are famously fast adopters, so developers will have to make a business decision whether or not to support iOS 6 going forward.*
- *Processors: With the introduction of their own designed processors, Apple has brought better battery life and graphics to their devices. There are currently several generations of Apple processors – each more capable than the next. Even devices that are less than two years old may feel slow with apps optimized for faster processors. Apple now only sells devices with A5 chips and newer.*

Each device has gone through multiple generations, but since Apple stops pushing updates to older devices, you do not necessarily have to worry about supporting them all. This limits the number of hardware/software version combinations you need to consider. Here's a breakdown of the devices and a snapshot of which support Apple's more recent system updates.

- *iPhone: There are seven generations of iPhones, however, only the 4, 4S, 5, 5S and 5C handsets still receive software updates (earlier versions have been discontinued).*

Since the release of the iPhone 5, developers now have to optimize for two different phone screen sizes – 3.5 inch and 4 inch.

- *iPad: The iPad is already in its fifth generation and all but the first generation iPad are still supported.*

Of all the iOS devices, the iPad family has the largest variation. The iPad Mini has a 7.9 inch display compared to the iPad's 9.4 inches.

- *iPod Touch: For development and testing purposes, the iPod Touch is largely the same as the iPhone. While the WiFi capable iPod Touch has gone through five iterations, only the most recent version is still supported.*

With the introduction of the Retina displays (iPad 3+, iPad Mini 2, iPhone 4/4S/5+, and iPod 5) it is essential to include Retina-enabled graphics when developing your app. Users are no longer tolerant of fuzzy graphics and if you neglect this step your app will look dated.

For the best results, your iOS app needs to be tested on each screen size/resolution combination available. This is particularly crucial when you are creating an iPad optimized app. Users don't consider the iPad and iPad Mini separate devices so your app needs to scale beautify and work appropriately on both screen sizes. (This should also be taken into consideration if you plan to support both newer iPhones and their older, smaller counterparts.)

Tablets also give you more real estate to play with, so don't be afraid to add new features and fun padding to make the app fill the screen and best meet a tablet user's needs. While smartphone apps are often used on-the-go, tablets are most commonly used at home, so more in-depth features and better browsing capabilities can be added to these apps – just don't go overboard.

Another matrix consideration is that iOS devices are offered with several different levels of storage – including 8GB, 16GB, 32GB, 64GB and 128GB. Not all devices have the same storage options, so when optimizing your apps make sure you know which are available. As a rule of thumb, keep the payload of your app's package as small as possible. Users have a lot of media these days and if they have a 16 gig iPhone your app could be the thing that goes when space gets tight.

iOS Focus Areas

There are two sides to every coin (unless you're cheating), so with the positive aspects of testing on iOS (that comparatively limited matrix we just discussed), you also run into a few areas that are a bit more difficult than if you were working with a different OS.

A unique feature of Apple products is the device specific UDID ("unique identifier"). If you want to test your app on live devices without going through the app store, you need the UDID of each device involved in the test.

The most famous difficultly associated with iOS apps is Apple's notorious vetting process. Luckily, there are a few things you can do to give your app a better shot at clearing that hurdle without incident.

Common App Store Rejection Reasons

Apple often gives developers one or more reasons for why an app was rejected. Here are a few rejection reasons explicitly stated by Apple, or reported by developers:

- *Limited Audience: The key to these rejections seems to be trying to limit the sale/download of the app. Apps that are targeted toward a limited audience but available to all iOS users seem to fair better.*
- *Repetitive Apps: Apps that are not "useful or [do not] provide some form of lasting entertainment." In other words, an app idea that is already overdone.*
- *Unrefined/Unprepared Apps: An app that looks like it was poorly created, not tested or rushed through production. The App Store prides itself on offering quality apps.*
- *Violence: Apps depicting realistic violence or realistic weapons used in an irresponsible way.*
- *Improper API/Background Services Use: Trying to beat the system by using one of the iOS APIs or background features in a way other than its intended use.*

(e.g. Running a silent audio clip in the background to enable the app to remain running.)

- *Poaching Keywords: You cannot use the names of pre-existing apps for the keywords for your new app.*
- *Limited Features/Little Usefulness: Joke, novelty and brand showcase apps can be caught by this rule if they do not offer users any actual usefulness.*

Avoiding an iOS faux pas from the beginning will save time and give your app a better shot at making it to the big show. Keep in mind, reasons for rejection are constantly evolving and new reasons often crop up in rejection letters to developers. A more detailed list of reasons for rejections can be found in the App Store Review Guidelines.

Permissions

Apple has strict rules on what apps can and cannot do. Here are a few key things you should avoid.

- *UDID: Remember that unique device identifier? Some developers have apps access this ID for tracking and advertising purposes. In early 2012, because of privacy concerns, Apple began rejecting apps that accessed the UDID without alerting users. Apple has since released an advertising identifier that can be used in its place.*
- *Collecting Personal Data: Apple will reject apps that collect personal user data without first notifying and obtaining a user's permission. Notification is often incorporated into the app installation process, with a user's continued installation considered consent.*
- *Address Book Access: Users were outraged when it was revealed that some apps were accessing user address books without permission. Since it came to light that this was a common practice for apps, Apple has disallowed apps from accessing, transmitting and storing contact information without user permission.*

- *Excessive Permissions: Before news broke about apps accessing data without permission, it wasn't uncommon for apps to access far more data and applications than strictly necessary. Those days are gone and developers now need to be transparent about what data and applications their apps will be able to access. Additionally, apps must now ask permission to use location, the microphone and to access the camera roll.*

Memory

Memory plays a big role when it comes to iOS devices. While Apple doesn't explicitly state the RAM availability, older devices are limited by their capabilities and may bog down under memory-intensive apps.

Be sure to test the memory usage of your app and test for memory leaks. Keep in mind that an app may perform differently when there is limited memory available compared to when there is ample free memory. How does your app perform when there are several apps running simultaneously?

VoiceOver

VoiceOver is Apple's assistive feature that allows users to interact with their devices via spoken commands. VoiceOver will also describe an app's UI to users. Apple provides a UI Accessibility programming interface to help developers ensure all apps are VoiceOver compatible. Your app must work with Apple assistive features or it will not be accepted into the App Store.

UIKit

Apple provides a fairly flexible UIKit for creating apps around standard interactions. With Xcode 5, Apple brings the Interface Builder and Auto Layout features to a whole new level enabling developers to design their app once and flex to different iOS devices. UIKit also provides provisions for Dynamic Type to change according to user settings.

Just because these features exist doesn't mean you should rely on them entirely. Instead of assuming everything is working correctly, it's imperative to test any app using Interface Builder, Auto Layout or Dynamic Type on multiple devices for verification.

> *"The app approval process is in place to ensure that applications are reliable, perform as expected, and are free of explicit and offensive material. We review every app on the App Store based on a set of technical, content, and design criteria."*
>
> -Apple App Store

Helpful Tools and Tips

Here are a few resources that might make iOS development and testing a bit easier.

- *Review the App Store Review Guidelines which is available to registered Apple developers. The guidelines "provide rules and examples across a range of development topics, including user interface design, functionality, content, and the use of specific technologies." Even if your app is pristine and passes every test you throw at it, if it doesn't fit within these guidelines it was all for nothing.*
- *The iOS Human Interface Guidelines and iOS 7 UI Transition Guide will help ensure your app's UI meets expectations. Topics include: Platform Characteristics, Human Interface Principles, App Design Strategies, User Experience Guidelines and Custom Icon and Image Creation Guidelines. The Transition Guide will help you navigate the major UI changes of iOS 7.*
- *Crash reports are logged automatically on Apple devices. These reports can be accessed by syncing the test device to a computer. The .crash files begin with the application name and contain date and time information. In addition, <DEVICE_NAME> will appear at the end of the file name, before the extension, making it easier to keep track of each report.*

- *A useful companion to the crash report is the device console log. The console log is an iOS feature that includes information from every application on the device. This log can help pinpoint if your app is being adversely affected by other apps/ software. The console log does not last very long so be sure to access it quickly following a crash, otherwise the details may be lost. The console log can be saved by connecting the device to a computer and accessing the "Console" tab within the device configuration utility.*
- *The built-in screen shot command is a good tool for documenting bugs. Holding the home and power button simultaneously will send a snapshot of the device's screen to the iCloud (and subsequently to all your connected devices).*
- *Testing on an emulator is fine, but it's always good to test on real devices as well. Emulators differ from real devices when it comes to CPU type, screen interaction (mouse versus touch screen), memory capacity and usage, zooming functionality, network connectivity, sleep mode, accelerometer response, etc.*
- *Use the Test Navigator tool in Xcode 5 to help run unit tests on each build. That will give you the confidence that your build passed basic unit tests and wasn't crippled with obvious bugs before you send it to testers.*
- *Instruments is an application that traces and profiles iOS code. It is available as part of Xcode Tools. One of the most helpful instruments is the Leaks template which monitors memory usage of the app and detects memory leaks.*
- *Apphance is another valuable tool for iOS testing. This multi-functional tool sends automated crash reports whenever a crash happens (whether in testing or production). Apphance also makes it easy for testers to capture and send bug reports and screenshots from directly within the app. You see all of the bug reports from your app at a glance, along with detailed information about device type, screen resolution, logs, and more. Apphance also provides over-the-air app distribution that makes getting the latest version of your app into the hands (and devices) of testers quick and easy.*
- *As Apple continues to release new hardware, you need to optimize and test your app again to ensure it works – and looks – just as well on the latest technology.*

It is important to regularly review all of Apple's developer guidelines – from App Store review policies to Human Interface expectations to new UI guides – as they are continuously changing and updating. Knowing Apple's policies and being aware of changes before they take effect can save you time and effort.

About Applause

Applause is leading the app quality revolution by enabling companies to deliver digital experiences that win - from web to mobile to wearables and beyond. By combining in-the-wild testing services, software tools and analytics, Applause helps companies achieve the 360° app quality™ they need to thrive in the modern apps economy. Thousands of companies – including Google, Fox, Amazon, Box, Concur and Runkeeper – choose Applause to launch apps that delight their users. Learn more at www.applause.com.