Mobile Application Testing

by Optimus Information

Guide to managing quality of a mobile application.

Optimus Information is a mobile application and website development firm located in Vancouver, Canada. Visit our website at www.optimusinfo.com to get your mobile project started today!

optimus
information

eBook

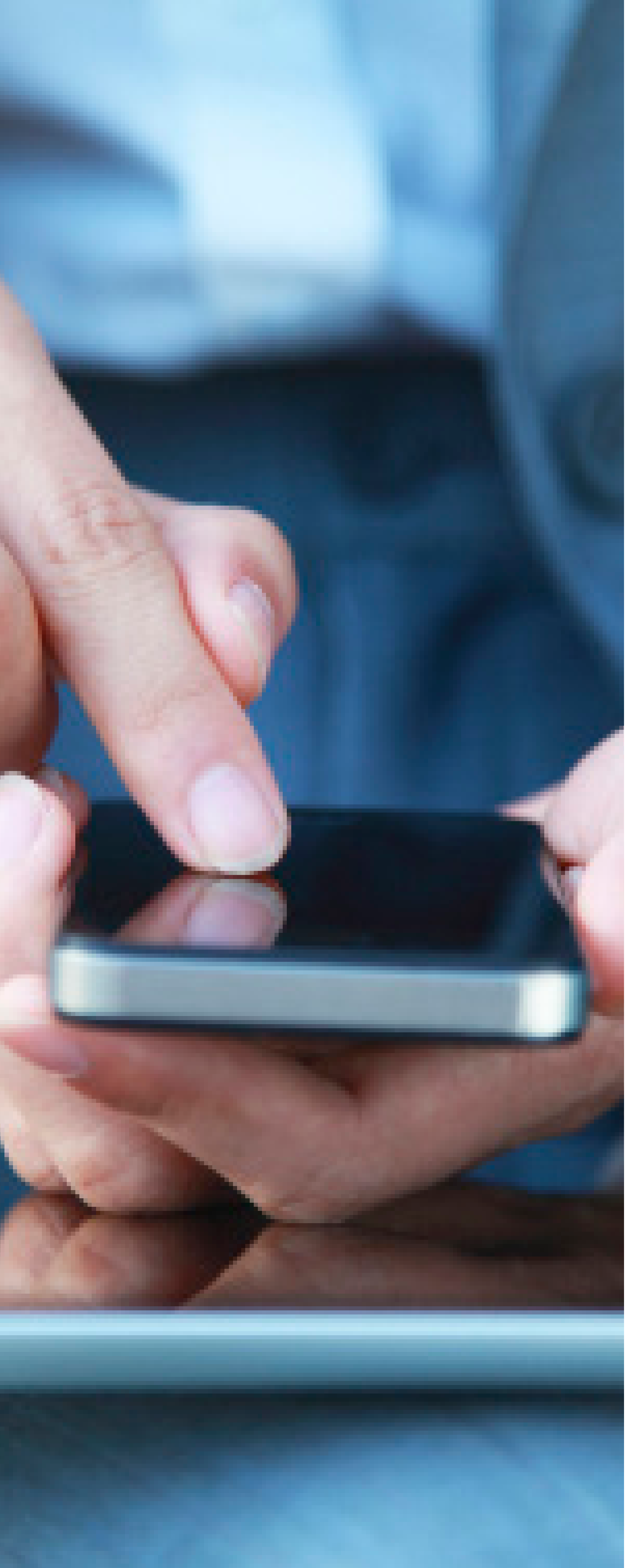


Table Of Content

Introduction

This guide will provide you with suggestions and strategies for testing a mobile application.

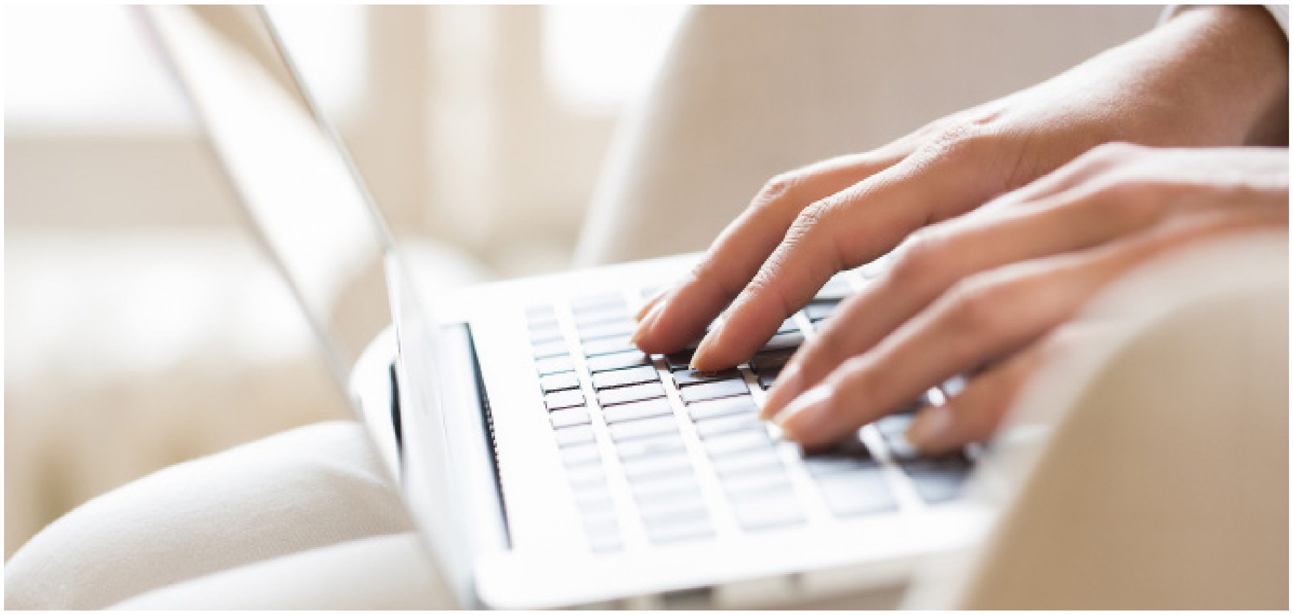
It will help you plan your QA resources and equip your QA team with the knowledge necessary for testing mobile applications.

Who should read this guide?

This guide is primarily for QA managers at independent software vendors tasked with creating large or complex mobile apps.

It should also provide value to QA teams and anyone testing a large or complex mobile app.

Key Challenges



PLATFORMS - VARIETY, DIVERSITY, LIMITATIONS

Mobile devices differ in terms of operating system, hardware capability and form factor. This variety, which is great for consumers, leads to complexity for testers. Many times, mobile application testers see the same application behave differently on different devices.

With thousands of different hardware/OS combinations, testing all the functionality of an application on every device is impossible. The best approach is to prioritize platform coverage, either based on specific customer needs or based on market share.

USER EXPECTATIONS

Expectations about mobile application performance and reliability are different than for other types of software. Users have access to thousands of applications that are either cheap or free.

If they install a new application and experience what they perceive to be a serious problem within the first few minutes, they will not hesitate to uninstall the app (and potentially leave a negative review in the app marketplace). Mobile apps are also used in short bursts, compared to traditional or desktop apps.

Here are some other expectations:

- Fast download time
- Fast start-up time
- Good response times for all common user interactions
- Visual appeal
- Small application footprint (because of limited storage capacity on mobile devices)

Native Apps vs. Mobile Web Apps

The mobile web is growing and mobile web applications may be more suitable for some enterprise uses than native apps. It is worth learning the similarities and differences in testing native apps and testing the mobile web.

Similarities

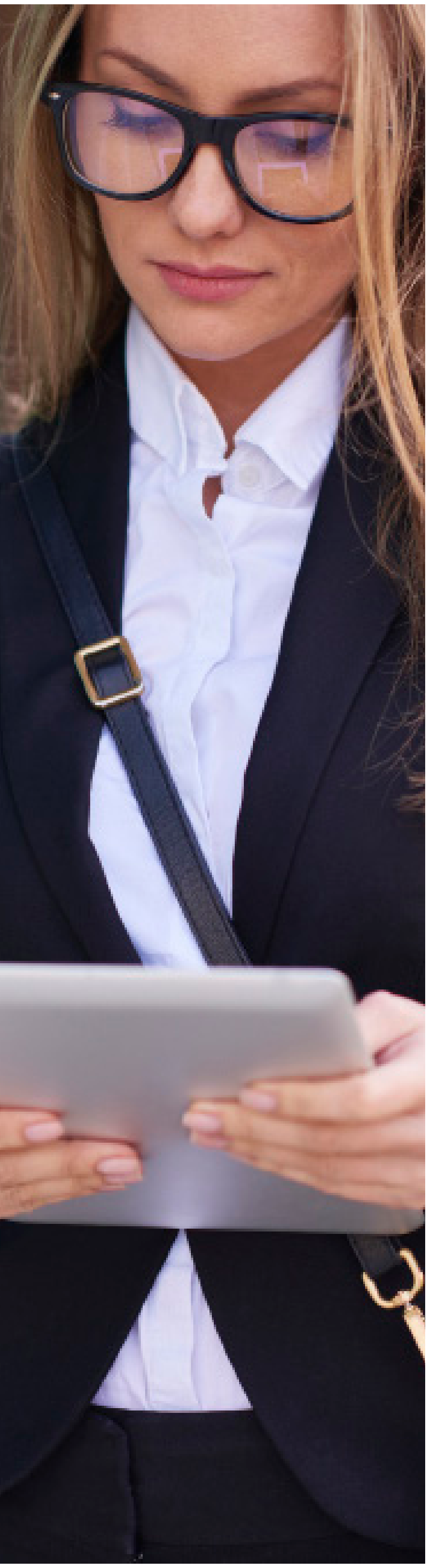
• Testing different screen sizes and form factors is necessary

• Testing on real devices is a must.

• Test automation tools are available (but most test automation tools for automating native apps testing do not support mobile web app test automation and vice versa).

Differences

Native Apps	Mobile Web Apps
Testing app installation, uninstallation and update is required.	No app installation, uninstallation and update testing.
Testing offline functionality.is required.	Testing offline functionality of apps using HTML5 local storage.
Minimal or no browser testing needed.	Extensive browser testing for supported browser and OS combinations.
Gesture functionality such as swipe and tilt, needs to be tested.	Less gesture testing as fewer gestures are available to web apps.
Apps need to be tested against the design guidelines and requirements set by different mobile platforms.	



Test Devices

With the incredible diversity of devices available, it is impossible to test an application against all of the devices it will be used on. Therefore, we have to answer two questions.

1. How should we select our test devices so that we are confident of the application quality in different devices?

2. Are there any alternatives to real devices?

To answer the first question, we should consider the following:

1. What target platforms and devices does this application target for? Is it iOS only? Is it tablets only?

2. What are the popular devices in the market by OS platforms? By models? By form factors?

3. Some people recommend bounds testing. Define an upper and lower limit for hardware constraints, and test each of those extremes. If possible, test a couple popular devices that fall within these constraints. Bounds testing is best for tighter budgets; it assures that there are no outliers in your matrix, but it isn't as comprehensive as it could be. [1]

To get the latest market share, you can refer to these sources:

Mobile Browser Usage

- StatCounter: http://gs.statcounter.com/
- Net Market Share: http://marketshare.hitslink.com/browser-market-share.aspx?qprid=0&qpcustomd=1
- Wikipedia: http://en.wikipedia.org/wiki/Usage_share_of_web_browsers

Phone and Platform Popularity

- Android Developers Dashboards (http://developer.android.com/about/dashboards/index.html) collects the data of the active devices that have accessed Google Play in a recent 14-day period or 7-day period, and displays the distributions of SDK versions installed, and the distributions of the screen sizes and densities.
- AppBrain (http://www.appbrain.com/stats/top-android-phones) shows the statistics of Android phones -most used phones and most common SDK installed by the AppBrain users
- StatCounter (http://gs.statcounter.com/) offers statistics on the popularity of different operating systems.

The cost of acquiring all of the test devices is expensive, but there are alternatives out there. Let's explore some of the options.

Simulators/ Emulators

Simulators and emulators are used generally in the early stage of development.

Pros	Cons
They are free. Every vendor offers them.	Testing with emulators is far different from using the application in the real world, so it does not guarantee the application will work on real devices.
Users can choose the sizes and OS versions, or even-models to test with.	Unable to test the application with hardware capabilities.
Allows developers and testers to verify certain functionality that is not specific to any device, carrier or operating system.	Unable to test network interoperability - networkrelated events (e.g. calls and text messages, etc.) and-different network technologies (e.g. 3G, LTE, etc.).
Convenient to test with simulators or emulators on laptops.	Application performance on the emulators does not reflect the actual performance.

Real Devices

Testing on real devices is a must because the feel of the application are hard to be mimicked on emulators or remote devices. In addition, you are not able to test the hardware capabilities and network without actual devices.

Some facilities have a device library, for example WaveFront in Vancouver. Some vendors offer loan or rental programs to development partners. You can check out some of the vendor programs:

- Samsung: http://www.samsungmobileb2b.com/partner/program_overview.do?board_seq=39
- Nokia: http://www.developer.nokia.com/Developer_Programs/
- Blackberry: https://partners.blackberry.com/web/guest/new-program-overview

Native Apps	Mobile Web Apps
Testing on real handsets gives reliable and accurate results	It is costly to acquire real devices (continuously because new devices come out frequently) and maintain them
Interoperability testing is possible because testing is performed on a live network	Real devices may be lost or stolen
Provides true user experience	Resetting devices between tests can be timeconsuming or requires expensive hardware.
Easier to expose performance defects with real handsets in a realistic environment.	

Remote Device Services

Some third-party companies provide services which allow developers and testers to test the application on various devices and networks remotely over the Internet.

The most popular service providers:

- Keynote DeviceAnywhere (http://www.keynotedeviceanywhere.com/)
- Perfecto Mobile (http://www.perfectomobile.com/)
- SOASTA TouchTest Mobile Lab (http://www.soasta.com/products/touchtest-mobile-labs/)

Some vendors provide free access to their remote devices labs, for example:

- Samsung (http://developer.samsung.com/remotetestlab/rtlDeviceList.action)
- Nokia (http://www.developer.nokia.com/Devices/Remote_device_access/)

Pros	Cons
Provides a huge selection of devices (different models) on various network carriers	Latency occurs (depending on how far the client location is from the device location)
Allows testers to record video and capture images while testing	Unable to test applications that require peripheral devices
Offers an automation module for writing test scripts May support test cases management	The automation module is expensive
Some provide integration with testing systems like HP	Touch and gestures are not easy to accomplish
Offers different packages so that users can choose-according to their needs	

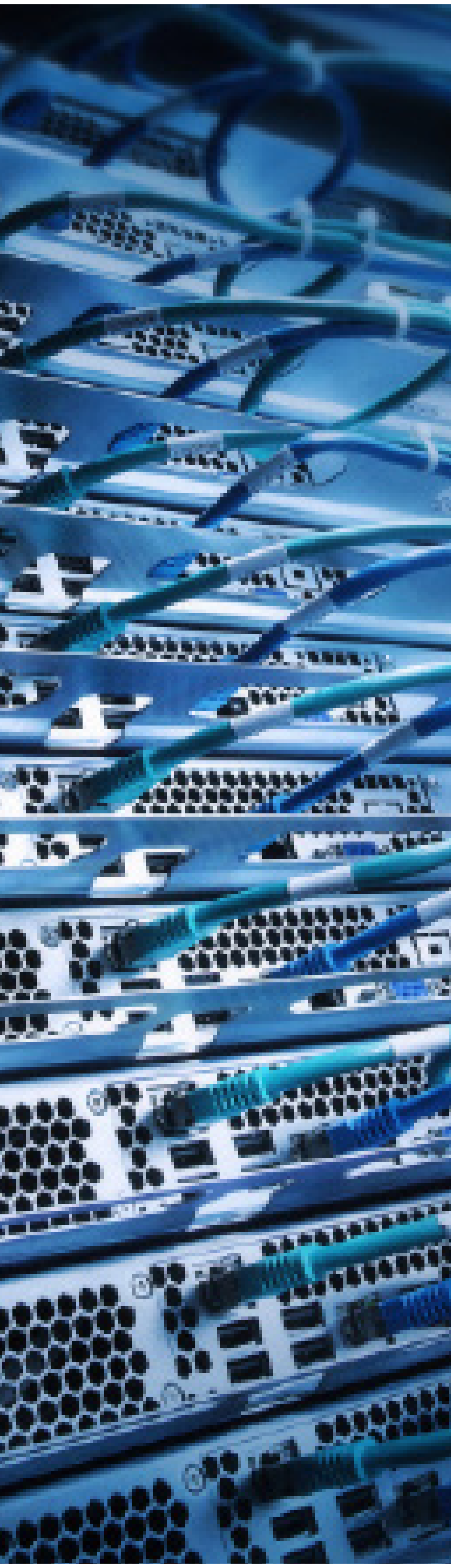
Crowdsourcing

Crowdsourcing is to outsource testing the application to an undefined public group. This approach is generally used as beta-testing.

Crowdsourcing services communities:

- Mob4Hire (http://www.mob4hire.com/)
- 99Tests (http://99tests.com/)

Pros	Cons
Get access to a broader range of devices on various-networks	The QA reports may not be credible and reliable
Obtain feedback and reports from thousands of people (instead of just ten people on the QA team)	Takes time and efforts to manage a large group of people
Lower price than hiring a professional	Collecting and analyzing the results may consume lots of resources
Good for testing localization	



Test Areas

FUNCTIONAL TESTING

Functional testing ensures the application is performing as specified in the requirements. The idea is thesame as the functional testing for desktop applications. Here are some areas that testers should specifically pay attention to when testing mobile apps

- Gestures functionality: Is the app responding as expected upon performing a gesture? For example, swipe, tilt, and highlight text.
- Phone orientation: Does your app crash when you rotate the phone?
- Social media integration: Are there problems with integrating with social media apps, such as Facebook,
- Twitter?
- Push notifications.
- Is installation requesting too many or too few permissions?
- Network connectivity: How does the app respond when there is no internet?
- Third-party services: How does your app handle non-responsive third-party services or APIs?
- Failure handling: Does your app handle failures gracefully?

User Experience

The No.1 rule is to follow the design guidelines specified for the particular OS platform. Each OS platform is designed to have its unique look and feel, so the application should match the OS design.

OS Platform Design Guidelines

iOS:

http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html

Android: http://developer.android.com/design/index.html

Windows Phone: http://dev.windowsphone.com/en-us/design

Other than the platform design guidelines, testing user experience on mobile applications is not very different from testing user experience on desktop applications. Some user experience guidelines that are especially important to mobile applications include

- Simplicity
- Context
- Good navigation
- Appropriate font sizes and button sizes
- Portrait and landscape orientation considerations

These articles provide good insights of improving mobile user experience:

- http://mobile.smashingmagazine.com/2012/07/12/elements-mobile-user-experience/
- http://mobilewebbestpractices.com/user-experience/
- http://www.ionagroup.com/2011/05/23/understanding-the-differences-between-designing-for-mobileand-desktop-media-while-learning-the-best-practices-for-designing-iphone-applications/
- http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/UEBestPractices/UEBestPractices.html#//apple_ref/doc/uid/TP40006556-CH20-SW1

  (A lot of designers recommended these guidelines for iOS)

Performance and Load Testing

Performance is another critical factor that contributes to the success of a mobile app. The following areas should be considered:

- Start-up time: How long does it take your app to launch and load the first screen?
- Response time: Does your app respond to user input instantly?
- Network: How does the app perform on various network types, like Wi-Fi and LTE? How about in different network strengths?
- Memory consumption: Memory is limited on mobile devices. How much memory does your app consume at peak? How does your app perform when there are lots of apps running in background?
- Battery consumption: How much battery does your app consume? Is this level of consumption acceptable in a context where a number of apps are running in the background as well?

If your app is built with a server/client model, you may want to consider load testing. Testing the server performance is not very different from how we usually do it, but for mobile apps, you have to simulate different network types and strengths during your load testing.

Security Testing

With the number of mobile device users soaring, mobile malware is on the rise too. Therefore, you may be interested in identifying the security vulnerabilities and addressing the risks, especially if your app has to process financial payments or confidential information.
Beware of the following areas:

- What are the changes in the device file system after installation and uninstallation?
- Is sensitive information encrypted when it is sent over the network?
- If sensitive information is stored on the device, where is it stored?
- Is it stored in a common storage shared among other apps?
- Is the stored sensitive data encrypted?
- If your app provides social network integration, what information is shared in the social network?
- If your app uses third-party services or APIs, are there any security risks involved?
- Is the app exposing confidential information in log files?

Compability Testing

Consider the following example [2]:

Your client would like to test the application on the top 10 Android devices. But with limited time and budget, you are not able to run all types of tests on all platforms. Therefore, you could suggest a complete test on the top 3 Android devices followed by a compatibility test on the remaining 7 devices.

Compatibility testing approaches:

- Test the critical areas or features of the application. The acceptance test suite is a good start.
- Issues found in the complete tests can be checked to see if they exist on the devices used in the compatibility test.
- Automated test scripts can ease the workload of compatibility testing. However, automated testing should not replace manual testing because UI or usability issues may not be spotted by the scripts.

Interrupt Testing

Interrupt testing is to test how the application reacts or functions when it is interrupted in different scenarios.

The following are some examples of interruptions [3]:

- Incoming and outgoing SMS and MMS
- Incoming and outgoing calls
- Incoming notifications
- Sudden or forced shutdown
- Cable insertion and removal for data transfer
- Network outage
- Switching between networks
- Media player on/off
- Device power cycle

Localization Testing

If your app is going to reach the global market, localizing your app is important so that it can adapt to specific markets or countries. However, localization testing is a challenge for the QA team since it requires QA to have knowledge about specific regions. Supporting multiple languages is a basic requirement; QA should also pay attention to the following when testing localization:

Region-specific functionality: Are there any features that are unique to a particular region?

- Input fields: Are some input fields active in certain regions and inactive in other regions? For example, some regions do not have ZIP codes, some countries use "Province" and some use "City" instead.
- Formats: Are the formats, such as date formats and phone number formats, adapted to the region?
- Legal requirements: Are there any legal requirements specific to a region?
- Network: To test the networks in a certain region, do you have a test team there? Or would you like to acquire remote device services, like Perfecto and DeviceAnywhere?
- Cultural knowledge: Do you have employees who are familiar with the target region and culture (not necessarily have to be a QA)? Even if the translation is perfect, the app may not be able to fully adapt to the region. For example, English speakers in North America are used to "Shopping Cart" whereas English speakers in Britain are used to "Shopping Basket".
- Automation tools: How does your automation tool identify objects? Does it depend on the language used?

Test Automation

CONSIDERATIONS FOR SELECTING AN AUTOMATION TOOL

Here are some aspects we can consider for selecting a test automation tool:

Supported platforms and devices	• Which OS platforms and models does the tool support? • Does it support testing with emulators and/or physical devices?
Script reusability	• Can the same test script run on different devices (i.e. resolutions, sizes, phone/ tablet) of the same OS? • How about devices of different OS versions, e.g. iOS 5.1 and iOS 6.0? • How about running the script on different devices of different OS platforms, e.g. Android and iPhone?
Functionality	• Does the tool support on-screen gestures, such as swipe, zoom and scroll? • Does it support outside application testing, such as phone battery and settings? • Does it provide call and SMS/MMS simulations?
Object identification methods	• What object identification methods does the tool offer, by the object native ID, image recognition, text recognition, or Web HTML5 (DOM)? • How easy is it to manipulate or access (e.g. look up the object properties, modify the object attributes) the objects using the tool?
Test diagnostics	• What kind of diagnostics does the tool collect and report, e.g. CPU usage, memory consumption?
Test environment integration	• Does the tool integrate into existing test environment, e.g. QTP, MSTest, TestComplete, etc.?
Others	• Does the tool support script parameterization and data-driven?

Test Ability

In many cases, much of functionality of a mobile application is actually accomplished on the server and exposed through an API. This provides options for testing and automation that don't always involve going through the UI. When testing the UI is a focus, many of the usual testability best practices should be followed, such as [4]:

- Use unique and clear identifiers for key UI elements. Make the names understandable and keep them consistent across platforms (iOS, Android, Windows Phone) and releases.
- Provide ways to query the state of the application.
- Create a client interface for testing the code underneath the GUI because it may be simpler to automate some tests without GUI operations.
- Develop tools or scripts that can complete the whole end-to-end testing (if the app under test requires receiving responses from another source, or interacting with other apps).

After Release

After your app is released, that does not mean you have nothing to do. To help you and your team plan for the next release and improve the quality of your app, you can consider the following:

- Use crash analytics tools to monitor and diagnose app crashes. Available tools include Flurry,
- BugSense, FlightPath (developed by TestFlight team), Crashlytics and Google Analytics.
- Read the comments and user feedbacks from app stores
- Check out the apps from the competitors

Cheat Sheet

If you don't have time to read and digest all the information in the above sections, just read the following top guidelines and keep them in mind:

1. Testing on real devices is required because testing with emulators is far from the reality.
2. When you select the test devices, consider the market shares by OS platforms, by models, and by form factors.
3. Test the application under different scenarios, such as Wi-Fi and LTE, running low on battery and incoming call or SMS.
4. The loading time of the application or mobile site is critical. The golden rule is 5 seconds, but research shows that 60% of mobile users will abandon the application or site if it does not load within 3 seconds [5].
5. Different OS types should have slightly different interface designs which should match the guidelines specified by the OS.

- iOS: http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html
- Android: http://developer.android.com/design/index.html
- Windows Phone: http://dev.windowsphone.com/en-us/design

6. Pay attention to the application submission requirements before submitting the application to the store. In fact, it is better to start early in the QA process.
7. The mobile landscape changes constantly, so keep yourself updated with the market and research.

Conclusion

Mobile devices are becoming more and more important, and mobile applications continue to grow in quality and quantity. QA is a valuable investment to ensure your company and your app remain competitive in the market.

The information in this guide will help you have a great start off on mobile application testing. If you have any questions, or you need further help, feel free to contact Optimus Information.

References

1. "App Testing: Which Android Devices Should You Choose?". blog.amadeusconsulting.com. Retrieved 2012-11-20
2. "Mobile App Compatibility Testing". testing4success.com. Retrieved 2012-11-20
3. "Mobile application testing, Wikipedia". wikipedia.com. Retrieved 2012-11-15.
4. "Page 167-168 in Mobile Developer's Guide to the Galaxy". scribd.com. Retrieved 2012-11-16
5. "Summary of Key Mobile Survey Findings by Compuware". compuware.com. Retrieved 2013-06-21