QATestLab
quality is a rule

Mobile App Testing:

The Ultimate Guide 2022

Table of **Contents**

Table of Contents

Click the section to jump ahead

Introduction

To attract and keep end-users, high quality of mobile apps is essential in today's extremely competitive market. The way to ensure that the app works flawlessly and completely satisfies the requirements of product owners and users and complies to the industry standards is through software testing. Quality assurance (QA) mainly helps to detect bugs, issues with functionality, problems with security, evaluate cross-device compatibility, and assess user-friendliness so that all these issues can be solved before the product release. Thus, software testing is an indispensable part of ensuring the popularity and competitiveness of a mobile application.

But checking the quality of a mobile app is not the same as checking a desktop application – it has a number of specifics and its own challenges.

This whitepaper strives to provide a comprehensive guide on how to conduct mobile app testing, as well as an extensive information on all the sides and aspects of mobile app testing.

In addition to the general guidelines on testing mobile applications, our paper includes:

- a detailed checklist for every testing type
- the specifics of testing mobile apps for different operating systems (Android, iOS) and non-standard interfaces (for example, Xiaomi's MIUI)
- information on the automation of mobile app testing (including the challenges of test automation for mobile games)

Table of Contents

Click the section to jump ahead

1 Types of Mobile Apps: **Native, Web, and Hybrid**

There are several types of mobile applications. Each has their own distinct features that can significantly affect testing. So, it is important to determine the app's type before checking it and to know how every mobile app type differs from the others. This chapter provides the information necessary to do it.

Native mobile apps are developed for mobiles and tablets running on a particular operating system (Android, iOS, and others). This type of mobile applications can have the richest functionality because of the access to all the features of the device.

Mobile web apps are websites designed to work in mobile browsers. They don't depend on a particular mobile operating system (OS).

Hybrid apps combine the features of native mobile apps and mobile web apps. Like native apps, they run on a mobile device, not inside a mobile browser. Like mobile web apps, they are written with HTML5, JavaScript, CSS, and other web technologies.

Table of Contents

Click the section to jump ahead

2 **Types of Environments** for Mobile App Testing

One of the first things to do when testing a mobile app is to choose the type of an environment in which it will be tested. There are four options: real devices, emulators, simulators, and cloud-based tools. Usually, a combination of several environments brings the best testing results. In this chapter, we will look at each of the testing environments in more detail.

REAL DEVICES, or physical mobiles and tablets, allow the highest reliability and accuracy of mobile apps' testing. A number of tests, such as testing an app's performance in real environments, reaction to low battery, and interruption testing, can only be done on real devices. But the huge and ever-growing number of devices in the market makes it impossible to test a mobile app on physical versions of all of them. When choosing on what real devices to test, take into account the app's target audience, the devices that are currently most popular, and that the devices with different OS, screen resolutions, and characteristics should be present. Real devices are usually used during the later stages of testing.

EMULATORS are virtual testing tools that imitate the hardware and software environments of real devices on computers. Emulators are written in assembly (machine-level) language. They are a great choice when testing the way the application interacts with the target environment's hardware and software. For instance, if it is necessary to check how the different memory allocations on the device affect the app's performance. An example of a popular emulator is Android Emulator.

SIMULATORS are virtual testing tools that imitate only the software, but not the hardware, environment of real devices. They are written in high-level programming languages. Simulators are faster than emulators, but they are less reliable, less suitable for debugging, and cannot be used for tests having to do with hardware. Simulators are used, for instance, when it is necessary to check how the application's interface works under different screen resolutions. An example of a popular simulator is iOS Simulator.

CLOUD-based testing tools allow a tester to access real devices virtually, through a cloud. This solution has many of the advantages of real devices, even though some tests that require the presence of the physical device (for example, checking how the app responds to the user utilizing it while walking) still cannot be performed. One more possible drawback of cloud-based testing solutions is the dependence on the Internet connection. Examples of cloud providers are BrowserStack and Sauce Labs.

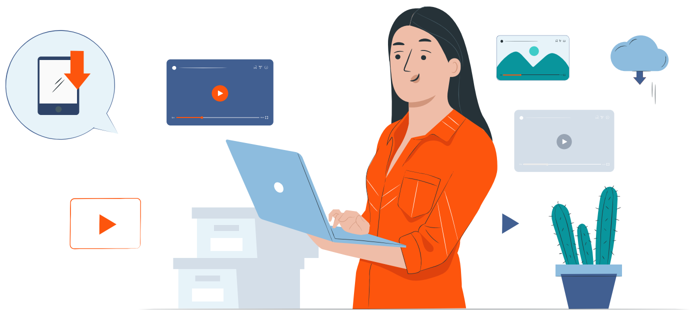
Table of Contents

Click the section to jump ahead

3 **Types** of Mobile App Testing

In this chapter, we will provide a checklist and some tips on how to perform all the important and widely used types of mobile app testing and will explain the meaning and essence of each type. We hope that this information will help the readers find the answers to all the questions they may have about different testing types and will be useful for their testing process.

Different types of testing are required to ensure that all the various aspects of a mobile app are of high quality and working properly. Each type specializes in its own area: functionality, usability, performance, security, and many more.



3.1

Installation testing checks whether the processes of installing, updating, and uninstalling an app work properly.

TEST ACTIVITIES:

- Make sure the installation, update, and uninstallation processes are quick, smooth, and intuitive.
- Ensure that, if an update is canceled midstream, the old version of the app is still working properly.
- Make sure that the app is deleted completely after uninstallation, and no junk files remain in the system.

Table of Contents

Click the section to jump ahead

3.2

Functional testing checks if all the functionality of an app works according to the requirements.

TEST ACTIVITIES:

- Check if all the fields are working properly and can be easily identified as mandatory or non-mandatory.
- Make sure that there are no issues in the navigation and scrolling.
- Confirm that the social network options are working correctly.
- Ensure that the sound and video effects are in line with the requirements.
- If payment transactions are needed, make sure that all the necessary payment methods are supported: Paypal, Visa, Mastercard, etc.
- Check whether the error messages display correctly when there is an error or the user did something wrong.
- Ensure that multitasking in the app is possible when needed, and there are no issues with it.
- Confirm that the work of the app does not have negative effects on the performance of other apps installed on the device.

Table of Contents

Click the section to jump ahead

3.3

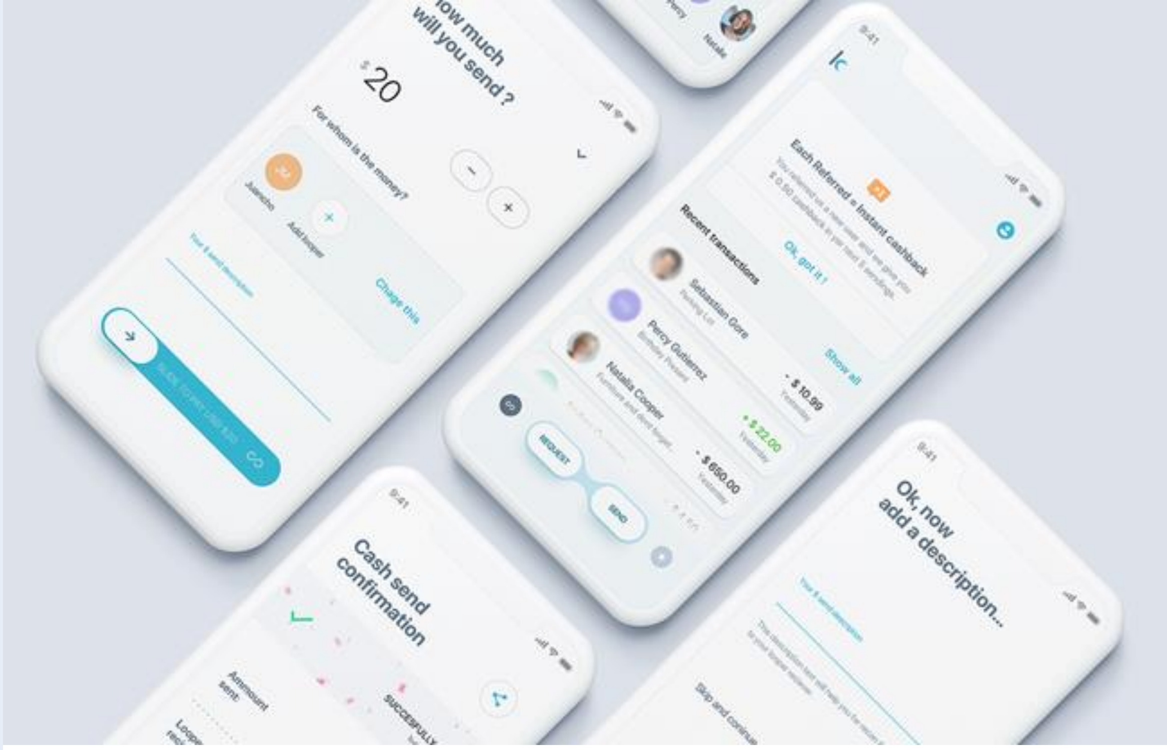
Compatibility / Integration testing checks if an application works correctly in different environments and on different devices.

TEST ACTIVITIES:

- Make sure that the app properly works on different types of devices (mobiles, tablets, etc.) with different screen resolutions and configurations so that all parts of the app interface are visible and functional.
- Check if the application can work correctly in different OS environments (Android, iOS, and others) as well as their different versions.
- If the mobile app is accessed through a browser, ensure the proper work of the application in different mobile browsers.
- Confirm that the app works properly when using different network standards (2G, 3G, 4G, 5G, and WiFi).

Table of Contents

Click the section to jump ahead



3.4

Graphic User Interface (GUI) testing checks if an app's interface works properly and meets the requirements.

TEST ACTIVITIES:

- Make sure that all the buttons, icons, menu options, toolbars, links, and other functional interface elements are displaying and working correctly.
- Ensure the proper display, quality, formatting, and alignment of images and text in the interface.
- Confirm that all interface elements display and work correctly in portrait and landscape modes.
- Check if the interface elements display properly on retina and non-retina screens.

Table of Contents

Click the section to jump ahead

3.5

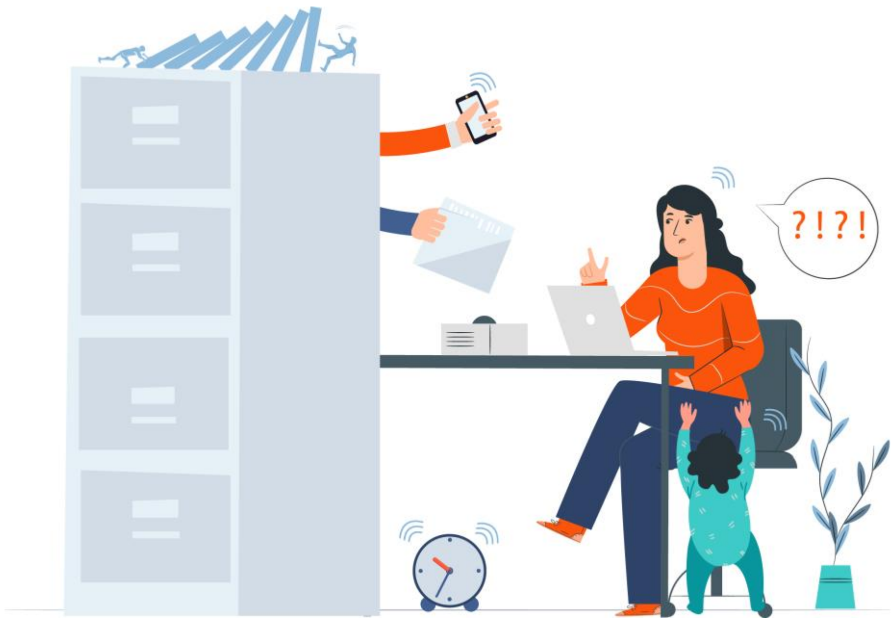
Usability testing checks if an app is user-friendly.

TEST ACTIVITIES:

- Ensure that all the app's functions are easy to use and easy to find.
- Make sure that the app's navigation is clear, smooth, and intuitive.
- Check if the app's design and layout follow the industry standards.
- Confirm that the app's design is attractive, pleasant, and user-friendly.
- Check if the app's buttons are big enough to be convenient for a user with any size of fingers.
- Make sure that the text in the app is clear, visible, and big enough to be convenient for users with different sharpness of vision.
- Ensure that a clear and easy-to-understand user manual is available for the users who don't know how to use the app.

Table of Contents

Click the section to jump ahead



3.6

Interruption testing checks how an app responds to various interruptions that obstruct its operation.

TEST ACTIVITIES:

- Check how the app reacts to incoming phone calls and messages.
- Test how the app responds to system notifications.
- Check how the app reacts to low battery and battery removal.
- Test how the app responds to charger insertion, charging, and charger removal.
- Check how the app reacts to an alarm.
- Test how the app responds to a sudden device shutdown.
- Check how the app reacts to loss of Internet connection.
- If the app needs GPS, test how it responds to the loss of GPS signal.

Table of Contents

Click the section to jump ahead

3.7

Security testing checks if an app has any security vulnerabilities and if data breaches are possible.

TEST ACTIVITIES:

- Confirm that the personal, sensitive, and financial information of the app users is properly protected.
- Check if the app can withstand a brute force attack.
- Make sure that the app is protected against malicious injections.
- Confirm that the app demands proper authentication before allowing access to any sensitive information.
- Ensure that the app's session expiration time is appropriate.

TIPS:

- Execute penetration and vulnerability testing first.
- While testing, simulate real-life conditions.
- The earlier security testing is started, the better.
- Anticipate many unpredictable situations and exceptions in security testing.
- Document meticulously because there may be a need to provide the information about the app's security to a controlling authority.

Table of Contents

Click the section to jump ahead

3.8

Performance testing checks how an app performs under various (including extreme) conditions.

TEST ACTIVITIES:

- Make sure that the app functions properly under various load conditions.
- Check how the app works at peak usage.
- Test how the app responds to short usage spikes.
- Make sure the app performs satisfactorily during long periods of normal to mildly heavy load.
- Identify the bottlenecks in the app's performance.
- Ensure that the app's response time is per the requirements.
- Check if the app's resource usage, battery consumption, CPU usage, and memory leakage are reasonable and match the requirements.
- Test how the app responds to switching the network between 2G, 3G, 4G, 5G, and WiFi.
- Make sure that the app works fine in the background.
- Check if the app's startup time is reasonable (ideally, it should be no more than 2 seconds).
- Ensure that the app works fine when the user carries the device and moves around.

3.9

Regression testing checks whether the recent changes and latest bug fixes in an app led to new issues or negatively affected the existing and already tested functionality in some way.

TIP:

Regression testing is mostly automated.
A tester automatically reruns the tests that were performed before the change.

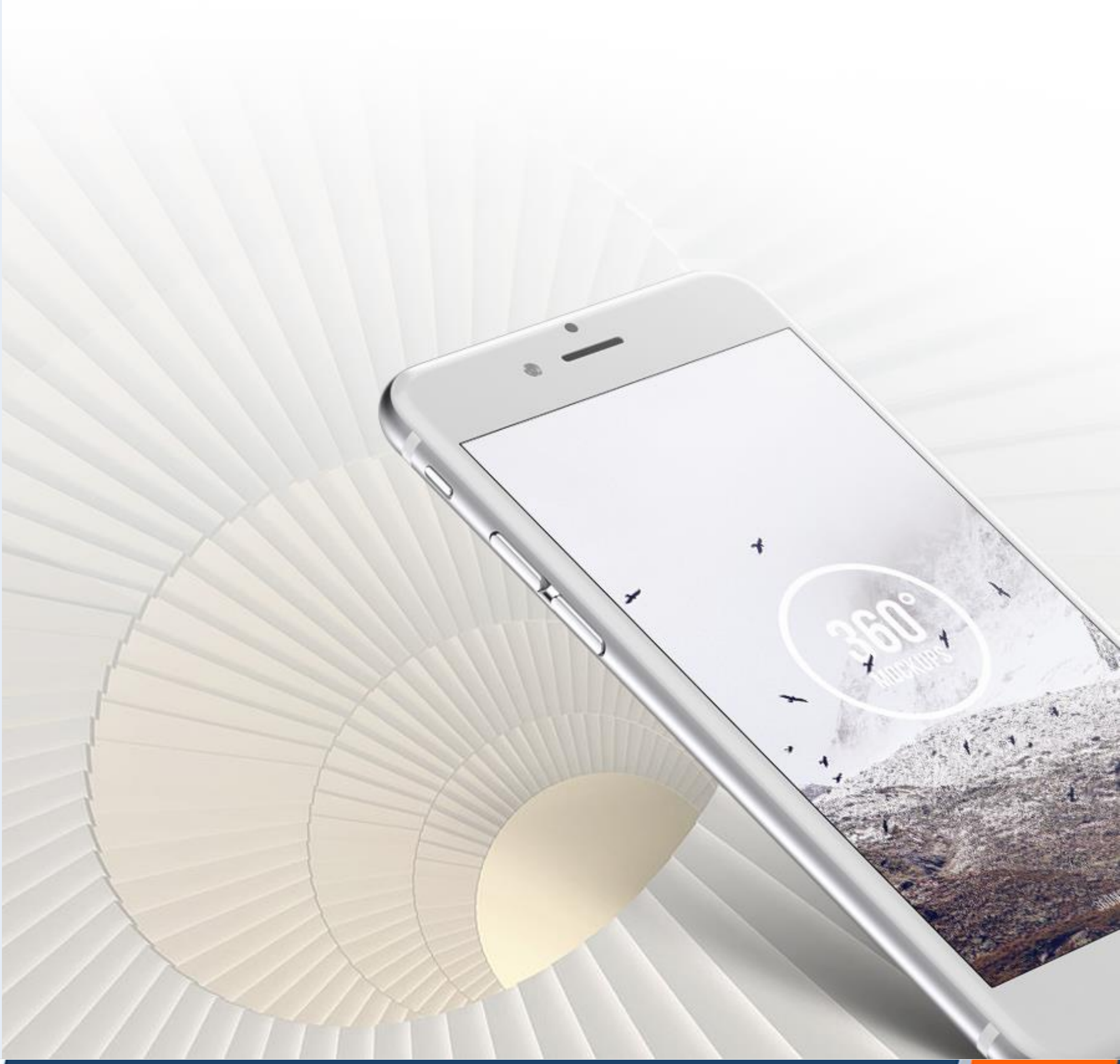


Table of Contents

Click the section to jump ahead

3.10

Localization testing checks if an app is properly adapted for users belonging to specific countries and cultures.

TEST ACTIVITIES:

- Check if all the text in the app is properly translated into the targeted language.
- Confirm that the date formats are the ones used in the targeted country.
- Make sure that all the prices are in the appropriate local currencies.
- Check if the number delimiters are the ones used in the targeted country.
- If any units of measure, length, weight, distance, temperature, etc. are used in the app, confirm that they are the ones used in the targeted country.

3.11

Beta testing is the testing performed by real users on physical devices to get the opinion of real people about the app before the release.

TEST ACTIVITIES:

- Choose the demographic of the beta testers carefully.
- Consider the testing costs and available funds.
- Decide on the number of beta testers, taking the first 2 tips into account.
- Decide on the duration of the beta testing.

3.12

Recovery testing checks if the app can withstand and recover from various failures.

TEST ACTIVITIES:

- Test if the app can successfully recover from various crash scenarios.
- Make sure that the data in the app can be recovered after a loss of connection.
- Check if the app can successfully recover in the event of a system failure.
- Test how the app responds to a transaction failure or recovers if a transaction was interrupted.

3.13

Unit testing checks if every unit of the code written for an app works correctly.

TIP:

Unit testing is usually automated. It is mostly performed by the app developers.

Table of Contents

Click the section to jump ahead

4 Testing Mobile Apps for **Different Operating Systems and Interfaces**

Not only are there countless different devices when it comes to mobile app testing, but also, on a more fundamental level, different operating systems (OSs) and user interfaces (UIs) that also affect testing a lot.

In this chapter, we will describe in detail the specifics of testing mobile apps for Android and iOS. We will also provide unique information based on the experience of our QA engineers about the challenges of testing mobile apps on MIUI, a heavily modified Android-based user interface by Xiaomi.

Android and iOS, the absolute monopolists among mobile OSs, naturally have many differences and distinct features. So, mobile apps created for these OSs have their own specifics, which greatly influence not only these apps development, but also testing.

Moreover, as Android is open-source, there are a number of highly customized versions of UIs based on this OS. As these interfaces significantly differ from pure Android, they cause their own challenges for mobile app testing. One of the most modified versions of UIs based on Android is MIUI created by Xiaomi for its devices.

4.1 Android

- Android is an open-source mobile OS created by Google and mostly based on Linux OS. Java is the main programming language used for Android apps. As it is open source, many different manufacturers use Android for their devices and occasionally tweak the Android version to fit their needs. This creates additional difficulties during testing as the app should work correctly on as many Android devices and variations as possible.
- Android is installed on all kinds of devices with different layouts, sizes, screen resolutions, user interfaces, brands, hardware specifications, and other characteristics. While testing an Android app, a tester should ensure that the app properly works on all the devices likely to be used by the end users. So choosing on what devices to test becomes an important task. In case of Android apps, using not only real devices, but also emulators and/or Cloud-based testing tools for testing becomes a necessity.
- OS version updates on Android happen much slower and less consistently than on iOS because, in case of Android, they are up to various manufacturers. So, many different older versions of Android are still in use by countless users all over the world. This further complicates testing as an Android app should properly work on many versions of Android.
- Because of Android's popularity, number of users, and the variety of devices it is used on, Android apps are often targeted by hackers. So it is important to pay a particular attention to security testing while testing an Android app.
- Testing an Android app requires more time and resources than testing an iOS app.

4.2 iOS

- iOS is a closed-source mobile OS created by Apple exclusively for Apple devices. It uses the XNU kernel. iOS apps can be written in the older Objective-C programming language, but now they are mostly written in Swift. iOS being a closed system used on a limited number of devices makes testing iOS apps easier than testing Android apps.

- There are much fewer Apple devices than Android devices, and the characteristics of Apple devices are similar. This considerably simplifies testing.

- Apple promotes using the latest version of iOS on all Apple devices, and the OS update process is unified. So testers only need to check the functioning of iOS apps on 2 iOS versions at any time: the latest version and the previous one.

- Testing an iOS app requires less time and resources than testing an Android app.

- The iOS App Store, the only channel of distribution of iOS apps, has very strict guidelines as to what kind of apps it accepts. Its review process of apps is also much longer than in the Google Play Store. This makes very thorough testing of an iOS app particularly important so that it is accepted by the iOS App Store from the first attempt.

4.3 MIUI

- MIUI is a custom version of Android created by Xiaomi for its devices. Technically not a different operating system but a specific user interface based on Android, or an Android skin, MIUI is modified heavily enough to become its own thing and create its own challenges for mobile app testers.

- The most common issue of mobile apps occurring specifically on MIUI is that app notifications and push-notifications don't display. It happens because, by default, MIUI does not allow third-party apps notifications and prohibits third-party apps from working in the background to save the device battery and speed up performance. To solve the issue, one needs to go to Settings on the MIUI device and allow Notifications as well as Autostart for the app in question and to add the app to exclusions in the Battery & Performance section.
  A tester should keep this in mind during QA activities and should recommend the app development company to add the information about the way to solve the issue on MIUI devices to the app's manual or FAQ.

- By default, third-party mobile apps don't autostart on MIUI devices. If starting automatically and working in the background is essential for an app's proper performance, this app should be added to exclusions in the Autostart, Memory, and Battery & Performance sections of a MIUI device.

Table of Contents

Click the section to jump ahead

- GPS may sometimes not work correctly for mobile apps on MIUI devices due to MIUI's strict default Privacy settings. This issue can be solved by going to Settings -> Privacy -> Location on a MIUI device and allowing location access there as well as setting Location Mode to High Accuracy. Another step is to go to the Battery & Performance section and add the app that needs to use GPS to exclusions so that it can work in the background.

- When an app crashes on devices that work on some versions of MIUI, a pop-up notification about the crash may not appear because MIUI blocks third-party pop-up notifications by default.
  These notifications should be specifically enabled for the app in the Settings for the proper error message to appear.

MIUI V12.0.1.0.QJWMIXM | Stable 629M
Update available
Control center
New: Swipe down from the upper left corner to open Notification shade and from the upper right corner to open Control center
Dark mode
Optimization: Brightness and color improvements for wallpapers in Dark mode
Phone

5 Mobile Test **Automation**

In this chapter, we will describe the specifics of test automation for mobile app testing. We will provide an extensive list of the types of testing that should be automated, review a number of the most popular and useful test automation tools for mobile app testing, and explain the challenges of test automation when testing mobile games.

Test automation is essential for mobile apps testing because it allows to cover a sufficient number of devices in reasonable time during cross-device compatibility testing, which is especially important when checking mobile apps.

And then, of course, there are the usual potential benefits of test automation – saving time, increasing efficiency, and saving money.

5.1 **When** to Automate Testing

Test automation is the **only choice** for:

PERFORMANCE TESTING**:**
Load testing, in particular, cannot be done manually.

Test automation is the **best choice** for:

REGRESSION TESTING:
Rerunning the existing tests after each system update is repetitive, tedious, and time-consuming, so it should better be automated.

UNIT TESTING:
As an example of a small repetitive testing task with a predictable outcome, unit testing lands itself perfectly to automation.

SMOKE TESTING:
A simple and quick test to determine if the system should be tested further, smoke testing can be automated similarly to unit testing.

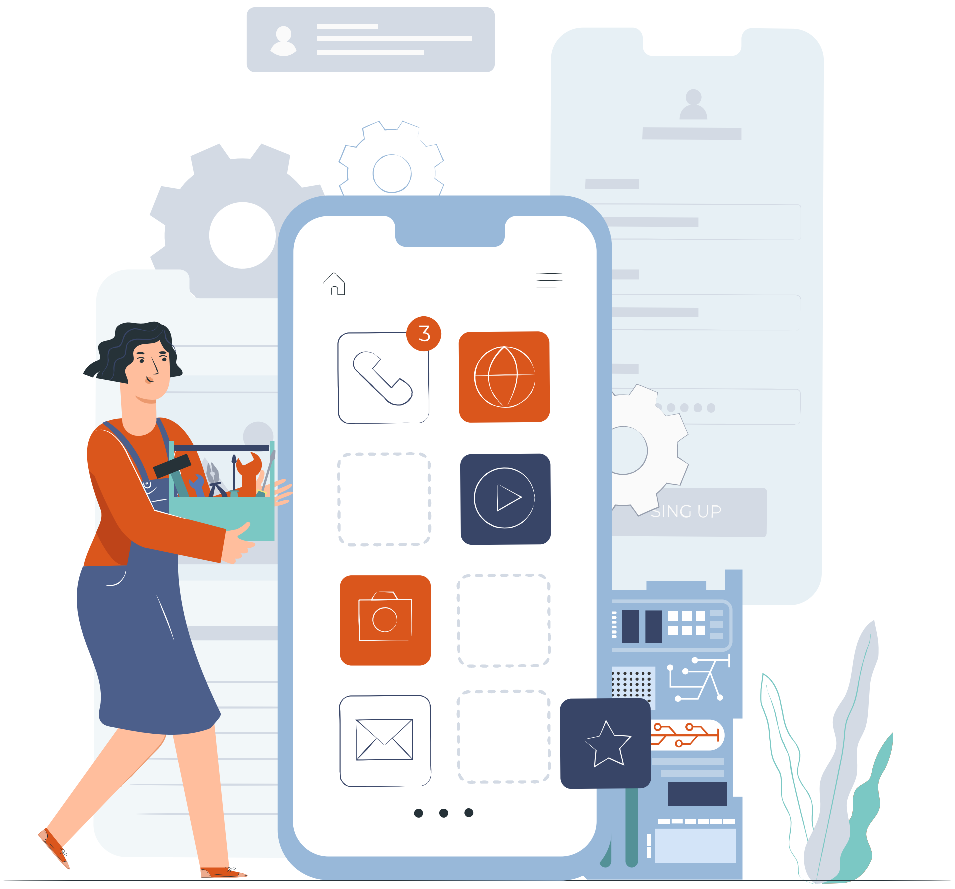
STATIC REPETITIVE TESTING:
Repetitive, tedious, and unchanging tasks that need to be performed on a constant basis should be automated to save time and minimize a chance of a human error.

DATA-DRIVEN TESTING:
Tests that require handling large volumes of data or complex calculations should better be automated for more accurate results and to avoid human errors.

CROSS-DEVICE TESTING:
Testing if an app works on a sufficient number of varying devices with different characteristics should better be partially automated to save time and efforts.

Table of Contents

Click the section to jump ahead



5.2 Mobile Test Automation **Tools**

In this chapter, we will provide relevant and up-to-date information on some of the most useful and popular mobile test automation tools that exist today.

We will review them, focusing on their advantages and disadvantages. This chapter addresses the following test automation tools that can be used when testing mobile apps: Appium, Cucumber, Cypress, Ranorex, TestProject, and Robot Framework.

Our goal is to inform our readers and make the process of choosing the right mobile test automation tool easier for them.

Table of Contents

Click the section to jump ahead

5.2.1 Appium

appium
Mobile Apps Automation Tool

Appium is the main tool for mobile test automation, just like Selenium is for web test automation. This is not an accident as Appium is based on Selenium WebDriver. So, it has very similar functionality. Appium can be used for test automation of all types of mobile apps, both Android and iOS. This tool also supports most programming languages.

Advantages	Disadvantages
• Wide functionality like in Selenium • Works for native, web, and hybrid mobile apps • Automates tests for both Android and iOS apps • Supports almost all programming languages • Free • Open-source • Has a huge active community always ready to help • Does not manipulate the app when running it through the server	• Limited work for apps on Android older than the 4.1 version • No recognition of user interface elements for apps built on Unity and Unreal

Table of Contents

Click the section to jump ahead

5.2.2 Cucumber

Cucumber is a widely used automation tool for mobile apps and other software. It allows to write test cases in an intuitive language that uses the Given, When, Then parameters. After the tests are written, Cucumber translates them into the specified programming language.

Advantages	Disadvantages
• Very easy to understand, requires minimal to no programming skills for writing test cases • Free • Does not require logging when writing tests • Supports many programming languages • Allows to reuse code because test case architecture is simple	• The setup of test framework is more complicated • The integration with other test framework components is more complex

5.2.3 Cypress

Cypress is a relatively new test automation tool, which is particularly useful for end-to-end testing. But JavaScript is the only programming language it supports.

Advantages	Disadvantages
• Open-source • Has good documentation • Provides a time machine feature, which gives the test runner a possibility to roll back to particular steps of test execution sequence • Has an inbuilt automatic waiting mechanism • Allows to write unit tests among other types of test cases	• Supports only one programming language – JavaScript • Large package size

5.2.4 Ranorex

Ranorex allows to test different kinds of apps on emulators, simulators, and real devices. This test automation tool provides the opportunity to use almost any programming language for testing, as well as to test with no knowledge of programming languages. Ranorex also enables improved data-driven testing.

Advantages	Disadvantages
• Easy to install • Allows to combine white-box, grey-box, and black-box testing • Can track UI elements • Includes an elements repository that allows to store screenshots • Has the built-in options of logging / running / reporting • Allows to re-use code • Provides a powerful play-record feature • Supports the majority of programming languages and testing without the knowledge of programming languages • Good customer support	• Not free • Weak community • The object identification option has long waiting time • No gesture support or drag-and-drop options • Scripts cannot be exported into different programming languages (VBScript, Java, etc.)

5.2.5
TestProject

TestProject is a cloud-based cross-platform end-to-end test automation tool. For mobile apps, it supports testing on iOS and Android operating systems, and on real devices, as well as emulators and simulators.

Advantages	Disadvantages
• Free • Has a good record and playback capability • Allows integration with CI/CD tools • Provides the possibility to record manual tests, write tests manually, or combine both methods • Offers efficient community support	• Currently only supports Java, C#, and Python, but more languages will be added in the future • Has no option to see the error output and make corrections through the UI, which complicates debugging • The interface can have hidden button elements, which complicates finding features

Table of Contents

Click the section to jump ahead

5.2.6 Robot Framework

Robot Framework is an open-source test automation framework that specializes in acceptance testing. Robot Framework is particularly useful for testing the apps that require testing multiple interfaces and technologies.

Advantages	Disadvantages
• Free • Allows to write tests without the knowledge of a programming language • Enables Keyword Driven Testing • Supports CI integration • Has various internal libraries that can be expanded by users • Creates detailed and convenient logs of test • Provides strong community support	• Supports only Java and Python • Does not allow running parallel tests

Table of Contents

Click the section to jump ahead

5.4
Automated Testing of Mobile **Games**

- The main difficulty of mobile game test automation is that games run on the OpenGL and ActiveX graphics containers, which provide direct screen access, bypassing the operating system level services. Because of this, it is difficult to access elements of the game inside the container through regular test automation tools. Currently, the most widespread way to access mobile game elements is to recognize the location of the element on the screen using image recognition. Image libraries such as the OpenCV library along with test automation tools and plugins are used for image recognition.

- Another challenge of test automation for mobile games related to image recognition is to find good image references and maintain a references database. The problem is that different tests on different devices require reference images with different characteristics (size of the image, how it is cropped, etc.). Also, when anything in the mobile game's interface changes, the image references should be adjusted. As a result, producing and maintaining a database of image references takes a lot of time and effort.

- One more difficulty is that the image recognition library needs time to process images, which may be too long for dynamic test cases.

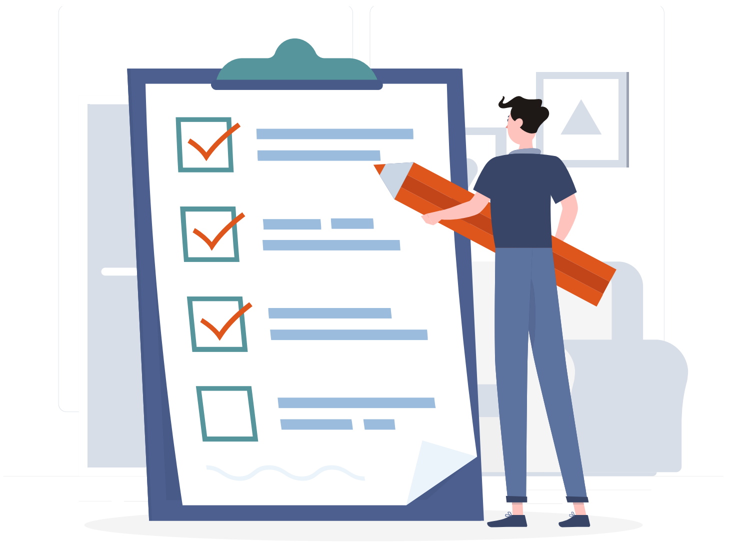
- The mobile game interface is faster than the interface of a regular mobile app, so the synchronization between the test scripts and the game may be challenging.

- It is more difficult to interpret the results of the automated tests for mobile games than for regular mobile apps because games have the randomization aspects and physics that regular apps don't have.

- While testing of any app should not be 100% automated, this is especially true for games testing because evaluating the fun factor of the game cannot be automated.

Table of Contents

Click the section to jump ahead



Conclusion

Mobile app testing significantly differs from web testing. The number and variety of mobile devices and tablets a mobile app should be compatible with create additional challenges. Mobile test automation can solve some of them, but it also has its specifics compared to web test automation.

We hope that this guide communicated and clarified everything the readers wanted to know about mobile app testing, from the general information to non-standard cases and specific challenges. The checklist and explanations we provided will help to evaluate the quality of a mobile app from all sides to make sure it is ready for release.

In case of any further questions about mobile app testing or any need of assistance with software testing of a mobile app, please contact QATestLab. Our experts will be happy to help.

Table of Contents

Click the section to jump ahead

About QATestLab

QATestLab is a leading software testing provider with more than 15 years of experience. It helps companies and product owners assess the quality of their software to ensure great customer experience and profitability.

With more than 250 QA engineers skilled in mobile app and web end-to-end testing, QATestLab can cover all the testing needs. We test, automate, consult, and provide a test lab of 350+ real devices.

QATestLab helps to verify the quality of mobile apps by:

- Checking the performance of the existing and added functionality
- Assessing the compatibility of the app with various devices and third-party systems
- Testing if the app is user-friendly
- Detecting security vulnerabilities
- Performing and assisting with mobile test automation

Contact us to start the collaboration and benefit from our QA expertise.

E-mail: contact@qa-testlab.com

Phone: +357 22 27-04-66

Facebook: QATestLab

Linkedin: QATestLab

Twitter: QATestLab

Website: qatestlab.com