Perfecto
by Perforce

THE COMPLETE GUIDE TO MOBILE AUTOMATION TESTING

Contents

Introduction to Mobile Automation Testing

Mobile app testing is notoriously complex. With so many device and OS variations, with different functionalities across them all, you have a lot to cover when testing mobile applications.

Mobile app testing is more complicated than web testing. Mobile apps usually have more users and need to work on a broader range of devices. They also have more functionalities than web apps. And of course, these all must be tested.

On top of this, the mobile market remains incredibly fragmented. The diversity of iOS and Android devices change rapidly and continuously. Keeping up with new models and new operating systems is challenging, especially for Android.

The sheer volume of tests needing to be executed can be daunting for teams. The key here to keeping up with the ever-changing mobile market is through automated testing. But it can be difficult to automate complex mobile scenarios with real user conditions.

Use this guide to develop a better mobile app testing strategy that can mitigates risk, reduce escaped defects, and release high-quality apps faster than ever.

Understanding the 4 Different Types of Mobile Apps

There are four types of mobile apps that companies develop today. And your mobile testing approach might differ based on which type of app you're building.

NATIVE (IOS/ANDROID)

Native apps are specific to iOS or Android. An iOS app is built into an IPA binary file, that can be then tested with Appium and/or XCUITest frameworks. And an Android app is built into an APK package that can be then tested using Appium and/or Espresso frameworks.

HYBRID

Hybrid apps include a native application wrapper that is independent of iOS or Android. So, a hybrid application can access all operating systems specific capabilities. A hybrid application can be installed from the Apple App Store or Google Play.

Hybrid applications are also supported by the Appium test framework.

WEB

Web apps are accessed through mobile native browsers, such as Chrome, Safari, or Firefox. These are pure web applications. So, they are supported by the Selenium test framework.

PROGRESSIVE WEB APP (PWA)

A progressive web app is an installable web link specific to iOS and Android. Instagram and Twitter are both examples of PWAs. You can create a shortcut to these apps and install them — without going to the App Store or Google Play.

The Top Types of Mobile Testing

There are many different types of testing for mobile apps, and for good reason. The mobile app space is incredibly fragmented. You need to ensure your mobile app works seamlessly across different devices, OSes, generations, and more. And that's where mobile testing comes into play.

1. FUNCTIONAL TESTING

Functional testing is the most basic testing activity. It ensures that all the features of your app function as they should. Functional testing ensures your app functions as expected across devices, OSes, and other variations.

Some examples of functional testing include unit testing and integration testing, with unit testing serving as the foundation of the test automation pyramid. As a best practice, it is recommended that you use virtual devices for unit testing, in addition to real mobile devices later in the testing cycle.

Functional testing tests:

- Entering and exiting the app.
- Memory used by the app.
- Resuming the app after an interruption.
- Responding to calls and notifications with the app running.

2. PERFORMANCE TESTING

Performance testing tests your app and its performance under different conditions. Some examples of performance testing include load testing, stress testing, spike testing, and volume testing.

Performance testing tests:

- Apps under standard traffic levels.
- Apps with stress/load levels of traffic.
- The CPU utilization of your app.
- App performance under varying network conditions. These include:
- Ideal network conditions.
- Poor connectivity.
- Changing networks.
- Switching between 2G, 3G, 4G, and 5G.

3. LOCALIZATION TESTING

It's important that your mobile app functions properly no matter where it is being used. That's where localization testing comes into play, aligning the app with local language, culture, religious sentiments, language variances, and device accessibility.

Localization testing ensures that your app will be translated properly into regional languages, and that it will meet local regulations and even legal requirements.

4. COMPATIBILITY TESTING

Compatibility testing is a critical part of testing mobile apps, given the fragmented nature of the mobile space. Ensure that your mobile app works across all types of devices, generations, OSes, and different types of hardware and software with compatibility testing.

Prioritize the most common mobile devices and operating systems in your region with the latest data in our bi-annual Test Coverage Index. You can download it for free here.

5. INSTALLATION TESTING

Installation testing is a type of mobile testing that ensures your app downloads properly from the App Store or Google Play. This type of testing verifies that app installation and updates work seamlessly, as do uninstalls of the app.

6. SECURITY TESTING

Users want to ensure that their data or sensitive information won't be compromised. You can ensure this with security testing of your mobile app, with testing such as penetration testing, vulnerability testing, and security scanning. These types of mobile testing involve finding security weaknesses in the app to determine whether or not it can be breached.

7. REAL USER CONDITION TESTING

Real user condition testing of mobile apps is critical. By applying environmental and device conditions to your app testing, it gives you a truer look of what the end user experiences once your app is released. And fortunately, real user simulation testing can be automated.

Real user condition testing tests things like:

- Packet loss.
- Network throttling.
- Screen rotation.
- Conflicting apps.
- Interruptions from incoming calls and texts.

8. ACCESSIBILITY TESTING

Accessibility testing ensures that your app is usable for all users, including those with disabilities, such as vision impairment, hearing disabilities, and other physical or cognitive conditions. Accessibility testing also meets legal requirements. Without it, you could be faced with big fines.

Automated accessibility testing includes testing things like:

- Color contrast.
- Screen magnification.
- Screen reader compatibility.
- Speech recognition capability.

9. USABILITY TESTING

Sometimes referred to as user experience testing or exploratory testing, usability testing is a highly important type of mobile testing. This type of testing sees how user-friendly and intuitive the app is.

Usability testing functions as the top of the test automation pyramid. It is one type of testing which should be manual, not automated, with business testers running through business flows that actual users of the app would use. Real mobile devices are critical in this type of testing, as real devices closer reflect what the end user will experience.

10. AUTOMATED & CONTINUOUS TESTING

With so many types of mobile testing required, automation is an enabler for teams to test faster and more efficiently. Test automation isn't easy to achieve. But it's a must to keep pace with DevOps and extend test coverage.

If you are just getting started going from manual testing to automated testing, keep in mind that it's a best practice to start small. Don't try to automate everything all at once. Instead, stabilize a small set of automated tests and go from there.

When you reach the highest possible levels of test automation, with extended test coverage and full integration in your CI/CD pipeline, you've reached the continuous testing stage. At this point, you are automating all that is possible in a cloud infrastructure at a very large scale. Continuous testing is considered the gold standard of testing.

Selecting an Automation Framework for Mobile Testing

For automating mobile app testing, you have plenty of mobile testing frameworks to pick from. Choose one that is reliable, aligns with your objectives, integrates with your toolchain and current processes, and complements the skillset of your team.

Here are a few test automation frameworks for your considerations.

APPIUM

Appium is an open source framework and is the leading framework for mobile app testing. It's open source, which means it's backed by a community. And the Appium framework supports end-to-end testing in multiple languages. You can use it on iOS and Android devices using a WebDriver.

ESPRESSO

Espresso is another open source framework for mobile testing. But this framework is Android-specific. Created by Google, it's ideal for Android UI testing.

XCUITEST

XCUITest is an open source testing framework, but only for iOS apps. It's developed and maintained by Apple and is ideal for iOS-specific UI test automation. XCUITest is a developer-friendly framework for unit and functional testing.

QUANTUM

Quantum is an open source framework spearheaded by Perfecto. It is a BDD testing framework. That means you don't need to know code in order to write test scripts. You can create test flows in plain language with Quantum, which is great for teams without coding skills.

Should You Test On Real or Virtual Devices?

Real devices, on the one hand, come with great value and benefits but also with their own set of costs. Simulators and emulators are also beneficial and can deliver unique value to both developers and testers, but also have drawbacks.

An emulator is a desktop app that completely virtualizes all aspects of a real device, including both the hardware and OS. Simulators don't mimic the hardware or OS. They allow the user to validate application flows, but not in a true production environment.

WHEN TO USE EACH

Simulators and emulators are ideal for testing early in the dev cycle. Emulators are best for testing external behavior, and simulators for internal behavior. For example, use emulators for testing calculations and transactions, and simulators for internal hardware and firmware tests.

On the other hand, real device testing is the most reliable and is best for usability and performance testing. You can also apply real user conditions for more realistic testing.

ADVANTAGES OF SIMULATORS AND EMULATORS

Simulators and emulators are both virtual devices. Both solutions virtualize mobile devices, and offer a number of advantages for testing mobile applications.

1. Variety

Simulators and emulators can virtualize a large variety of devices and operating system permutations. This capability allows users to validate very easily on multiple platforms as well on cases requiring specific device/OS combinations.

2. Price

Virtual device solutions are cheaper than real devices by a wide margin. This applies to both local and cloud-based solutions. The need to drive testing early in the development process requires teams to scale and test more.

3. Starting on a Baseline

With virtual devices, you can always start from the same device state. Achieving this with real devices may require doing a factory reset, which can take a lot of time and effort. The fact that the virtual device always starts from the same point helps.

MAIN USE CASES FOR SIMULATORS AND EMULATORS

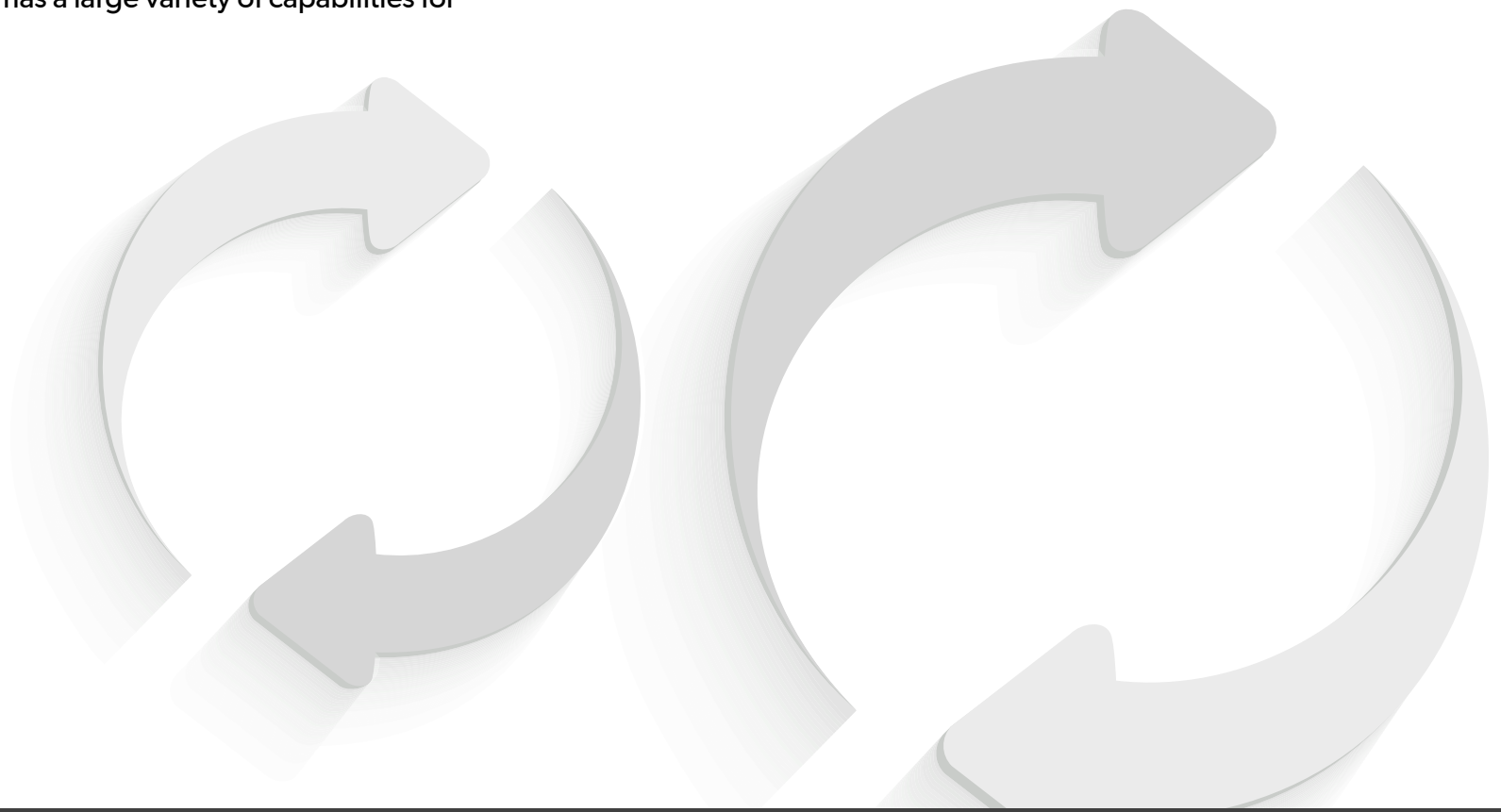
Local Development and Validation

Developers and testers use simulators and emulators on their local machines for development, app debugging, and local validations. The most common IDEs for native applications come with virtual device tools as part of the basic installation.

Xcode comes with simulator functionality while Android Studio comes with an emulator. Both became stable and mature in the last few years and each has a large variety of capabilities for advanced validations.

Continuous Integration Testing

The major use case for virtual device labs is for continuous integration (CI) testing. The increased adoption of DevOps and Agile methodologies drives teams to test more in the early stages of the development process, otherwise known as shift left testing. New test automation frameworks that are more aligned to developers' skills and tools, such as Espresso and XCUITest, help development teams to increase their test automation coverage.



4 ADVANTAGES OF USING A REAL MOBILE DEVICE

1. There's Nothing Like the Real Thing

While virtual devices are good for basic validations, to really validate that an application works, you need to run tests against real devices. While virtual devices can be very helpful to validate functional flows, there may be cases of false positives. This means that while tests may pass, in reality, there may be issues with the application.

That's because simulators and emulators test on the "happy path" — the main flow of the application rather than trying to see how the application handles negative and extreme cases.

2. Better User Interface (UI) Validations

User interface validation should be done on real devices to validate the accuracy of the UI. In addition, usability issues are very easy to find while working on a real device, unlike virtual devices.

In most cases, where there is a need to enter input from the keyboard, real devices overlay the app, unlike virtual devices where the keyboard is presented next to the device interface.

3. More Accurate Performance Testing

Real devices provide more accurate and reliable performance measurements on transaction times. In addition to the implications specific hardware has on performance, virtual devices also render the UI differently.

4. Improved Hardware & Sensor-Related Validations

Common use cases that can be virtualized are ones that require interaction with device hardware and sensors, such as the camera, accelerometer, and biometrics. In some cases, the behavior of real and virtual devices may differ.

How to Create a Mobile Test Coverage Strategy

Across Android and iOS, the mobile market is incredibly fragmented. Users expect mobile apps to work seamlessly on their devices, but keeping up with the continuous flow of new models and new operating systems is difficult for testing teams. And given the fragmented nature of the mobile market, teams have an impossible amount of permutations to test.

The key here is to strategize and prioritize the most impactful permutations. This allows you to mitigate the risk of escaped defects through an extended test coverage strategy. Test coverage is very important for mobile apps. That's because…

USERS DON'T TOLERATE BUGS

Users today are simply less tolerant of buggy performance on the apps they love.

FUNCTIONAL TESTING ISN'T ENOUGH

The agile testing manifesto advises to cover all types of testing within the software iteration cycle. To prevent various defects, you'll need functional, performance, security, accessibility, and more types of testing.

App responsiveness and quality are critical to the digital experience and digital confidence, and add additional layers to testing strategies. This includes performing tests under real user conditions.

A COMPREHENSIVE TEST COVERAGE ANALYSIS IS THE ONLY WAY TO SUCCEED

For app developers, this shift to a digital-first world requires a comprehensive test coverage analysis.

It asks for more rigor and accuracy in:

- How digital test labs are built and maintained.
- What platforms to include and how often to test on these platforms.
- What test scenarios to include per each software iteration and when in the pipeline to execute them.

All while dealing with rapid market changes.

One of the best ways to do a test coverage analysis is to consult an index. Perfecto publishes a comprehensive mobile and web test coverage index biannually. You can download it for free here.

HOW TO ENSURE TEST COVERAGE

Here's how to ensure test coverage.

1. Collect the Data

The best way to define and plan your digital test coverage is to have a mix of the most relevant metrics. This what devices and browsers people use to visit your app and website and who your competitors are. You should also keep an eye on market share data and device releases and browser updates.

In addition, your teams should gather test scenario history to guide your future software iteration test scoping.

We recommend ensuring that only high-value, robust, and cross-platform test cases are part of the pipeline and CI. This isn't a one-time audit. Rather it's a continuous monitoring of your test suites to ensure the regression suites and others are up to date.

2. Determine Device and Platform Criteria

Next, you'll need to determine device and platform criteria.

At Perfecto, we recommend that teams include one or more platforms (smartphone, tablet, desktop browser) from each of the four groups below — no matter which test coverage goal they're trying to meet.

Reference Devices

This is a key group because it includes devices such as Google's Pixel. These devices are important because they will always be the first to get beta and GA versions of the Android OS. This allows dev and test teams enough time to test their apps on the upcoming platform. These devices should be part of the test lab despite their market adoption or share.

Popular Devices

This group is a no-brainer to include in your test lab. It can consist of devices coming from both your customer data and the greater mobile market.

Legacy Devices

In this group, we find older iPads, Samsung devices, and mobile OS versions. These devices are popular in various markets and as such, require testing. However, they're often slow to receive the latest OS updates. Also, running on older hardware with less CPU and memory can be challenging for modern applications that support newer features.

Emerging Devices

It's imperative to treat your digital platform as an ongoing effort and therefore it's crucial to keep an eye on new devices, operating systems, and other trends and be prepared to test them. This includes new devices or major beta versions of iOS or Chrome. Including these devices in the mix can save R&D time later on and also position your brand as ahead of the curve.

Once you are able to gather the above insights and make an informed decision on the test lab setup, you're ready to think about the size of your lab.

The DevTest team should understand how many of the test lab devices and platforms they actually need to cover all manual, automation, performance, and other test cases. They also should understand whether these platforms are sufficient for the entire team, based on head count.

Sizing the lab correctly can positively influence your team efficiency by eliminating cases where developers or testers are waiting for a device or desktop to be released by another user. It can also help reduce your overall testing cycle time by executing more tests in parallel on more platforms.

3. Include User Conditions in Your Test Coverage

After you have the right mix of devices for your lab, you'll need to think about the real life conditions your users experience each day.

- What networks do they use? Are they on Wi-Fi?
- How many apps do they run at once?

The digital platforms that consumers use are complex. You'll want to recreate specific user environments as much as you can in your test lab. You should also consider testing for these conditions on different screen orientations (portrait and landscape).

By taking a set of tests that run against a given number of devices on a functional level and factoring in real user conditions, you enhance your test coverage. You add more depth to your testing and increase the likelihood of meeting your user expectations.

4. Define the Right Coverage Mix

Now it's time to use the data you've collected, device and platform criteria, and user conditions to determine the right test coverage mix. This will help you build a lab with mobile devices and browsers.

Use the following testing considerations:

1. Supported locations (countries).
2. Supported mobile platforms (iOS, Android).
3. Target test coverage level (Essential, Enhanced, Extended).

We also recommend building your lab with the following guidelines to achieve the optimal test coverage and the least risk:

1. Test on the top 10 different device/tablet models on various OS versions, in what we call the "Essential" group.
2. Follow up by expanding to a list of top 25 total devices, including the top 10. This second group, the "Enhanced" group, will represent an optimized list from the market's longest tail of popular, legacy, and new device/OS combinations.
3. From there, you can move to the third group for "Extended" test coverage that can be met by testing on 32 different device/OS combinations. In this group, we will either see devices that are older but still need to be tested against, or very new but not popular yet.

As part of each group mix, the following attributes should be considered:

- Device and OS popularity, including Custom OS versions (market share).
- Screen sizes, resolution, and other screen attributes such as pixel per inch (PPI).
- Device age (launch date).
- New and trending devices and platforms.
- Operating system version update rate (e.g., reference devices like Android Pixel get a higher score).
- Unique device properties important for testing purposes — chipset, CPU, memory.
- Audience demographics.

The second and third groups offer the highest test coverage and the least risk to digital teams. You should also have an ongoing process of refreshing your test lab. This is usually done on a quarterly basis, to make sure nothing major has changed in your user base or in the market that would require a new device or OS.

Deciding Which Tests to Automate

High rates of automation are the goal of continuous testing. But not all of your test cases will be automated. So, which types of test cases should you automate?

A test case should be automated if:

- The task is going to be repeated.
- It's going to save time.
- The requirements, the test, or the task are low risk, stable, and unlikely to change often.
- The test is subject to human error.
- The test is time consuming.
- The test has significant downtime between steps.
- The test is repetitive.

These qualifications allow you to set standards for testing across the team and prioritize tests. This increases your chances of releasing on time. Most importantly, this enables you to get strong ROI on your test automation.

WHICH TESTS SHOULD YOU AUTOMATE FIRST?

Unit testing should take the top priority, followed by integration testing and functional testing. The types of test cases that should be automated have a lot of complex manual scenarios, especially those that require a lot of data or environment setup.

1. Unit Testing

Unit testing is the fastest method of testing and, therefore, should be the highest priority for your automation. That is because it's easier to debug. These are highly reusable tests. They are low cost to fix, and there's a whole host of frameworks that you can use to implement this regardless of your programming language.

2. Integration Testing

Integration testing, where we're testing our interfaces or modules, should also take high priority. These tests help us ensure that everything is working as expected. When automated, integration tests will run more quickly for us and be able to give us that feedback.

3. Functional Testing

With functional UI testing, there are a whole host of tools and frameworks that you can use that will match your development code base. So, you should take an approach that values it as an upfront concern. Running those tests will help identify flaky ones. And we don't want flaky tests.

Remember, just because you can automate a test doesn't mean that you should. If you automate a whole slew of tests that are going to require a great deal of upkeep, then you are investing additional time and money that you may not have.

Instead, you should focus on adopting a risk-based approach, such that you're only automating the most valuable tests. Take great care in automating those most valuable features, and making sure that they're automated correctly for long-term sustainability.

Mobile App Testing Checklist

FUNCTIONAL TESTING	
Sending and receiving messages with the app running.	
Responding to push notifications with the app running.	
Making and receiving calls with the app running.	
Resuming the app after an interruption.	
Rejecting calls with the app running.	
Entering and exiting the app.	
Memory used by the app.	
Your app's effect on the device's battery.	
Biometrics, like fingerprint and face ID.	
Functions like camera, screen resolution, and geolocation.	

SECURITY TESTING	
Two-factor authentication.	
Fingerprint and face ID.	
Proper storage of app data and private information.	
Proper encryption of information stored locally on the device.	

PERFORMANCE TESTING	
Your app under typical levels of traffic.	
Your app with stress/load levels of traffic.	
Your app with real user conditions.	
The CPU utilization of your app.	
The time to install and uninstall the app.	
The time to launch the app.	
App performance during low battery.	
Ideal network conditions.	
Poor connectivity.	
Changing network while moving.	
Dead zones.	
Switching between 2G, 3G, 4G, and 5G.	

ACCESSIBILITY TESTING	
Text color contrast.	
Screen magnification.	
Dynamic font size.	
Color ratios between text and background.	
Screen reader compatibility.	
Readability of the application.	
Speech recognition capability.	
UI hierarchy for consistent structure.	
Labels used by assistive technologies, like VoiceOver or TalkBack.	
Hit area size for designated user interaction.	

USABILITY/UX TESTING	
Testing the app when changing the screen orientation.	
Testing the navigation of the app.	
Testing the mobile menu of the app.	
Testing visual feedback for user interactions.	

COMPATIBILITY TESTING	
Screen size.	
Screen resolution.	
Device-specific features.	
OS-specific features.	
Changes in the UI.	

RELATED RESOURCES

- Mobile & Web Test Coverage Index
- The Buyer's Guide to Web & Mobile Test Automation Tools
- Mobile App Testing Strategy Combining Virtual and Real Devices
- Tech Talk: Mobile Testing Fundamentals
- The State of Test Automation Today
- Top 11 Challenges in Automated Testing & What to Do About Them

Perfecto
by Perforce

About Perfecto

Perfecto by Perforce enables exceptional digital experiences and helps you strengthen every interaction with a quality-first approach for web and mobile apps through a cloud-based test platform. The cloud is comprised of real devices, emulators, and simulators, along with real end-user conditions, giving you the truest test environment available.

Our customers, including 50 percent of the Fortune 500 companies across banking, insurance, retail, telecommunications, and media rely on Perfecto to deliver optimal mobile app functionality and end-user experiences, ensuring their brand's reputation, establishing loyal customers, and continually attracting new users. For more information about Perfecto, visit **perfecto.io**.

TRY PERFECTO