The QA Engineer's Guide to Mobile App Testing

eBook

KEYSIGHT

INTRODUCTION

Mobile devices and applications dominate virtually every aspect of our lives. As a result, developing and deploying high-quality mobile apps has quickly become a differentiator for businesses to remain competitive. With billions of mobile devices and apps in use daily, software development teams are under pressure to deliver apps that function correctly and meet user expectations.

Quality assurance (QA) engineers play a significant role in mobile app development. A Google study found that 81% of users will leave a mobile site or app if it doesn't satisfy their needs. If an app launches with a critical error, uninstalls can skyrocket, and negative reviews roll in.

Alternatively, releasing a well-tested, high-quality app can generate positive reviews and increase downloads, revenue, and brand awareness.

However, testing mobile apps is challenging and often complex. QA engineers can use test automation to streamline test processes, uncover bugs before they hit production, and ensure the successful delivery of high-quality apps to users.

This eBook identifies the challenges you may encounter when mobile testing. We will address how to overcome these challenges, explain the types of testing you need, offer test cases, and show how test automation helps you deliver high-quality apps on multiple platforms and devices.

Contents

CHAPTER 1

6 Key Challenges to Overcome When Testing Mobile Applications

GO TO CHAPTER 1 >

CHAPTER 2

Best Practice for Mobile App Testing

GO TO CHAPTER 2 >

CHAPTER 3

6 Main Testing Types Every QA Engineer Should Know

GO TO CHAPTER 3 >

CHAPTER 4

Keysight Eggplant – The One-Stop Shop for Mobile App Testing

GO TO CHAPTER 4 >

CHAPTER 1

6 Key Challenges to Overcome When Testing Mobile Applications

CHAPTER 1

6 Key Challenges to Overcome When Testing Mobile Applications

Mobile apps play a significant role in various industries. Therefore, QA engineers must focus testing on several areas, including performance, functionality, and user experience. However, testing has its challenges. To understand what to expect, you need to acknowledge the following considerations before embarking on your testing journey.

1. Device fragmentation

Device fragmentation refers to the wide variety of mobile devices, operating systems, screen sizes, and hardware configurations in the market. Due to the variety of devices, it is challenging for QA engineers to ensure that their apps function as intended across different handsets.

Manually testing mobile apps on various devices is often time-consuming and costly because it requires access to physical handsets. Plus, manufacturers regularly release new devices, further complicating the testing process.

Compatibility issues result from device fragmentation and impact app functionality. For example, a function or feature that works well on an Apple iPhone may not work on a Samsung handset, damaging the user experience.

> **90%** of people say they use multiple screens for everyday activities, such as booking a hotel or shopping for electronics
> **– Google/Ipsos, "The New Multi-Screen World"**

2. Numerous browsers

QA engineers must ensure that web-based mobile applications work on all browsers and platforms. Various web browsers have their own capabilities, functions, and rendering engines, making the testing even harder.

Regular updates, various layouts, and performance differences all impact the user experience (UX), so testing every version is essential. Like the issues arising from device fragmentation, one feature may work correctly on one browser but not on another.

Manual browser testing is complex due to the number of permutations and variables you must consider. With various browser versions, screen sizes, resolutions, and operating systems, it is impossible to manually cover every required test case. Manual testing is resource-intensive and time-consuming, which increases the chance of human error and inconsistent test results.

Browser market share worldwide

65.4%	Chrome
18.71%	Safari
4.46%	Edge
3%	Firefox
2.61%	Samsung internet
2.4%	Opera

January 2023. Source: statcounter

3. Various operating systems

Different handsets and devices run on various operating systems (OS), making manual testing for mobile apps demanding. In addition, different versions of OS updates can cause compatibility problems, adding to the complexity of testing.

Apart from the challenges involving testing devices and browsers, different operating systems affect mobile features and functionality differently. When new features launch or updates are necessary, performing regression testing manually is repetitive and can often lead to missed defects.

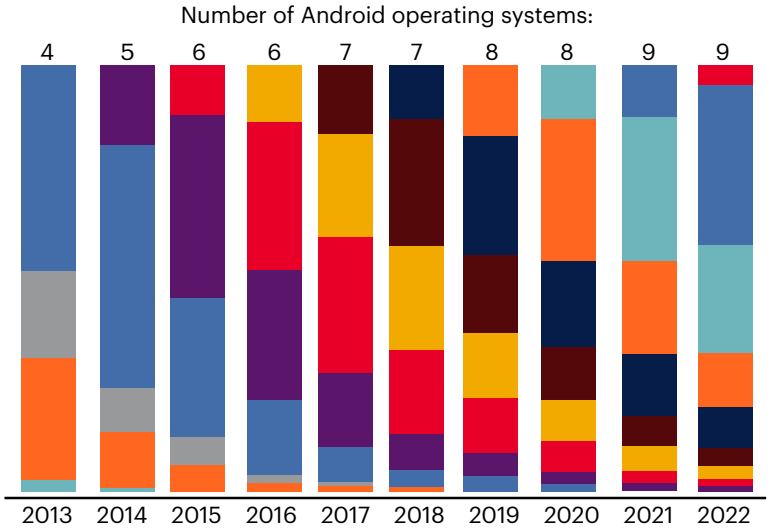


Figure 1. Android operating systems released each year from 2013 to 2022.
https://www.businessofapps.com/data/android-statistics/

4. Multiple layers of technology

Multiple layers of technology contribute to the overall performance of mobile apps and their capacity to deliver a positive UX. The source code is only one component of an app, so failing to test other layers, such as backend systems, application programming interfaces (APIs), and the user interface (UI), can result in a poor UX.

Mobile apps may have slow load times or need more data storage if backend system integrations remain untested. Similarly, if you fail to test API requests, errors can sneak in and crash your apps when they make calls to third-party services. And if you are testing different technology layers in isolation, there is no guarantee that the information they relay will display correctly on your mobile app's UI. Conducting end-to-end testing and testing every technology layer concurrently is critical to ensure a consistent and accurate experience.

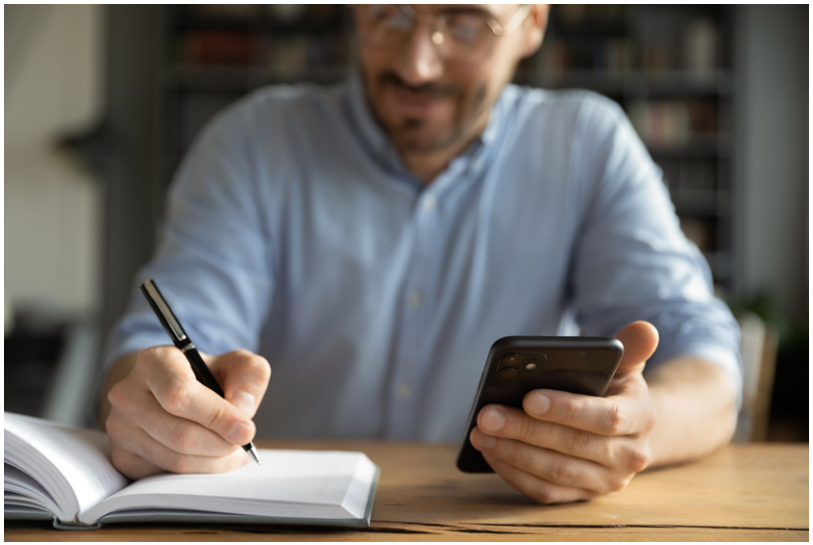
Want to Learn More?

> Solution Brief: *Modernizing the Retail Customer Experience with Full-Stack Testing*

5. Usability issues

Failing to adequately test your mobile app's UI can have significant consequences, negatively impacting user retention and installation rates. While the code in the object layer is critical to determining the app's functionality, relying solely on code verification doesn't guarantee a high-quality UI experience.

Users have certain expectations of an app's visual appeal, intuitiveness, and ease of use, which QA engineers cannot fully validate through code verification alone. For example, a pop-up window appearing at an inopportune moment, such as during a purchase, can prevent users from completing a transaction. Similarly, if the color of a login button is indistinguishable from the text "log in," it can prevent users from accessing their account information. QA engineers can only identify these issues by testing and validating the app's UI.



6. Changing user requirements

Feedback and user requirements constantly change and evolve. This presents significant challenges for QA engineers performing manual regression tests when new features, functionalities, and updates are necessary.

Identifying changes in thousands of lines of code during mobile app development is a complex and time-consuming task. In addition, mobile apps connect to multiple layers of technology that any changes will impact, making it essential to thoroughly test every configuration, platform, and system.

It's time-consuming to test every app component when user requirements change manually. Plus, the manual process is prone to human error, leading to missed defects that can negatively impact the app's performance and UX.

CHAPTER 2

Best Practices for Mobile App Testing

CHAPTER 2

Best Practices for Mobile App Testing

Optimize testing with automation

Test automation has become crucial to the mobile app software development life cycle. If you're still performing manual tests, it's time to change. Automated software testing can significantly increase the speed and efficiency of mobile app testing and offers numerous benefits. For example, you can run automated tests faster and more frequently than manual ones, helping uncover and fix defects early in development.

QA engineers can also improve the reliability and accuracy of their tests by automating repetitive and time-consuming tasks such as regression testing. As a result, automation dramatically reduces human error, helping to improve quality and the overall UX when a release hits production.

Want to Learn More?

> Blog – Avoiding "App-ocalypse": Why Mobile App Testing is a Must for Retail

> First impressions count, with 48% of users admitting app experience helps determine brand credibility **- Google**

Additionally, deploying test automation makes sense since user requirements change so frequently. Mobile apps are constantly evolving due to rapid feedback loops. Automated tests are more robust and easier to update and run than manual ones. Furthermore, app usage can increase, which requires test execution at scale. A test automation platform can run multiple tests concurrently and, if well-executed, on separate OS platforms to increase test coverage further.

Validate visual elements

Conducting image-based testing enables QA engineers to validate the visual appearance of mobile apps and uncover any defects that functional testing may miss. Verifying the quality of the object layer only goes so far. For example, a developer may write flawless code, but without validating the UI, problems can arise that damage the UX.

And with image-based testing, QA engineers can identify whether mobile apps adhere to an organization's brand guidelines, such as using the correct color, fonts, or even logo size.

Using automated software testing solutions that employ optical image and character recognition, QA engineers can quickly identify usability issues. Ensuring visual elements display correctly and function as intended increases the likelihood of apps remaining on a user's phone.

Take advantage of model-based testing

Model-based testing, or digital twins, can improve the efficiency and accuracy of software testing because it can create a visual representation of any mobile application. Using digital twins, QA engineers can create a model that accounts for every user journey and interaction. By combining this type of testing with artificial intelligence (AI) and machine learning (ML), it's possible to implement intelligent exploratory testing and dramatically increase test coverage beyond directed linear test cases.

QA engineers can also reduce test maintenance with model-based testing. For example, when features change to meet new user requirements, digital twins can automatically adapt and reflect the change without impacting the rest of the model. Tests are much more robust than manual scripting, which often break in the same conditions. Models also save time because engineers don't need to spend hours trawling through thousands of lines of code to identify the change that broke their scripts.

Want to Learn More?

> Solution Guide: *Enterprise Application Testing with Cloud-Based Device Farms*

Test any technology

With various operating systems, browsers, and devices, employing a technology-agnostic testing solution is helpful. This enables QA engineering teams to test mobile apps across various mobile handsets and web browsers.

The technology-agnostic approach also eliminates the need to invest in various testing tools and the required knowledge for using them. It is also more cost-effective and future-proofs testing because QA teams can test apps on any mobile device as technology evolves.

Device farms in the cloud

Cloud-based mobile device farms offer QA engineers a more efficient testing environment versus physical device labs. Based in the cloud, device farms make it much easier to scale up or down the number of devices requiring testing without additional hardware or investment.

Cloud-based device farms are a cost-effective way to access a wide range of handsets. QA engineers can quickly test different operating systems and various screen sizes and resolutions without the headache of purchasing and maintaining thousands of devices. Testing on different handset variations enables QA engineers to predict and monitor how an app performs across multiple devices and scenarios.

A single solution for end-to-end testing

Mobile apps often come integrated with APIs and multiple backend systems and databases. Unfortunately, end-to-end testing usually occurs in isolation; separate teams conduct these tests. Some QA teams test only a specific API call or request from a database to verify data accuracy. Others may only validate how the UI layer visually represents the API call. However, if a defect occurs at any layer of an app, the entire data journey will fail, creating an error-filled UX.

It's wise for QA teams to use a single solution that combines multiple technology layers and test them together. A unified solution enables engineers to orchestrate end-to-end testing by creating scenarios that verify API requests from a database through the object layer and validate the visual response at the UI of the mobile app.

CHAPTER 3

6 Types of Testing Every QA Engineer Should Know

CHAPTER 3

6 Types of Testing Every QA Engineer Should Know

With the increasing demand for mobile apps and faster releases, QA engineers must have a comprehensive understanding of several types of testing. Here are the six main testing types that every QA engineer should know to ensure that all apps are high-performing and deliver what users expect.

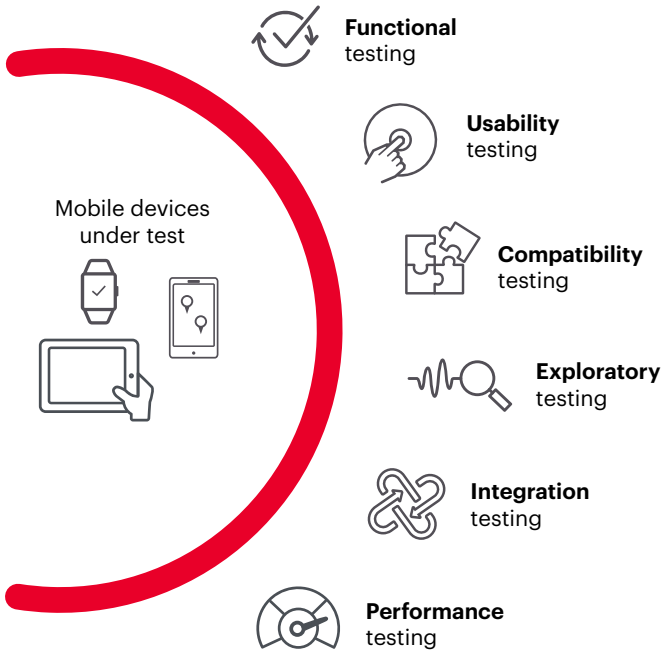


Figure 2. The six main types of testing QA engineers need to know.

1. Functional testing

Functional testing is a critical component of software testing that evaluates the app's core functionality and ensures that it works as expected. During functional testing, QA engineers verify that each app feature operates correctly, including its interactions with other components.

For example, in a mobile app, functional testing involves testing user scenarios, such as creating an account, adding items to a cart, making a payment, and receiving notifications. By thoroughly testing these functionalities, QA engineers can identify and resolve defects or compatibility issues before the app reaches production. Functional testing ensures users have a seamless and enjoyable experience, critical to the app's success.

2. Usability testing

Testing the user-friendliness of a mobile application is another vital step for QA engineers. Usability testing helps them understand how users interact with an app and identifies any customer experience-related bugs. Testing the UI is different from functional testing; it analyzes the display of visual elements and app usage. Failing to perform usability testing can significantly impact the UX and increase the likelihood of uninstalls.

For example, for an eCommerce mobile app, usability testing involves navigation within the app, button placement, assessing whether pop-ups prevent users from adding items to their carts, and incorrect product images.

Testing usability allows QA engineers to address and fix any issue before an app launches into the market and ensure the experience meets user expectations.

> A good UI design helps to increase application conversion rate by 200% and UX design by 400% **- Forrester**

3. Compatibility testing

Due to the number of operating systems, browsers, and devices available to users, QA engineers must verify an app's ability to run across every scenario as intended. This type of testing is essential to ensure a consistent UX regardless of the technology.

Here are examples of compatibility testing:

- **Screen size:** verify optimization of the app's UI and functionality for various screen sizes, rotations, and resolutions on different mobile devices.
- **Operating systems:** ensure compatibility and performance with iOS, Android, and Windows operating systems.
- **Browser compatibility:** test app functionality on the most popular browsers, such as Chrome, Firefox, and Safari.
- **Network performance:** test and monitor performance under different network conditions, including Wi-Fi, 3G, 4G, and 5G.

4. Exploratory testing

Users interact with mobile applications differently; manual tests cannot anticipate and test every user journey. Automated exploratory testing and the integration of ML-powered algorithms can address this challenge by comprehensively testing all possible user journeys.

ML-powered testing enables QA engineers to simulate a vast range of user interactions and identify potential defects that would be difficult to uncover through manual testing. This results in increased test coverage and a more thorough evaluation of the application's functionality.

Additionally, intelligent exploratory testing enables QA engineers to identify defects early to prevent flaws from reaching production and negatively impacting the UX.

5. Integration testing

Integration testing is critical for testing any mobile application because every app interacts with various technologies, components, and systems. These include web elements, APIs, and backend systems that can impact an app's UI.

Testing the end-to-end data flow between technology layers is crucial for high-quality user experiences. Example test cases could include:

- **UI to backend system communication:** QA engineers need to test data that enters at the UI level, accurately relays to a backend system, and processes correctly. This may include ensuring that data can be saved, retrieved, and displayed accurately at the UI.
- **API functionality:** Test API calls to ensure the correct data can be accessed from a backend system.
- **Validate objects with the UI and APIs:** Confirm that data is correctly interacting with the object layer and triggers the correct UI display and API calls.
- **End-to-end data flow:** Test the data flow through every layer of app technology. This includes verifying data accuracy in the backend system, API calls, and validating the visual representation at the UI.

6. Performance testing

QAs must conduct mobile performance testing to assess the app's ability to meet performance requirements under different conditions and usage scenarios. Users expect a mobile app to be fast and responsive, regardless of outside influences.

Performance testing helps identify bottlenecks or limitations impacting the UX, such as heavy usage, slow network conditions, or limited processing power.

Scenarios for performance testing can involve the following types of testing:

- **Load testing:** managing many concurrent users and testing behavior and functionality under peak load.
- **Stress testing:** identifying performance limitations when testing an app to its limits, such as dealing with a large volume of data or many users.
- **Response time:** ensuring your app is fast and responsive, regardless of network conditions to maintain a faultless user experience.
- **Network latency:** testing the mobile app under slow or unstable network connections to monitor performance issues related to network latency and bandwidth latency.

61% of users expect mobile apps to open within 4 seconds **- TechBeacon**

CHAPTER 4

Keysight Eggplant – The One-Stop Shop for Mobile App Testing

CHAPTER 4

Keysight Eggplant – The One-Stop Shop for Mobile App Testing

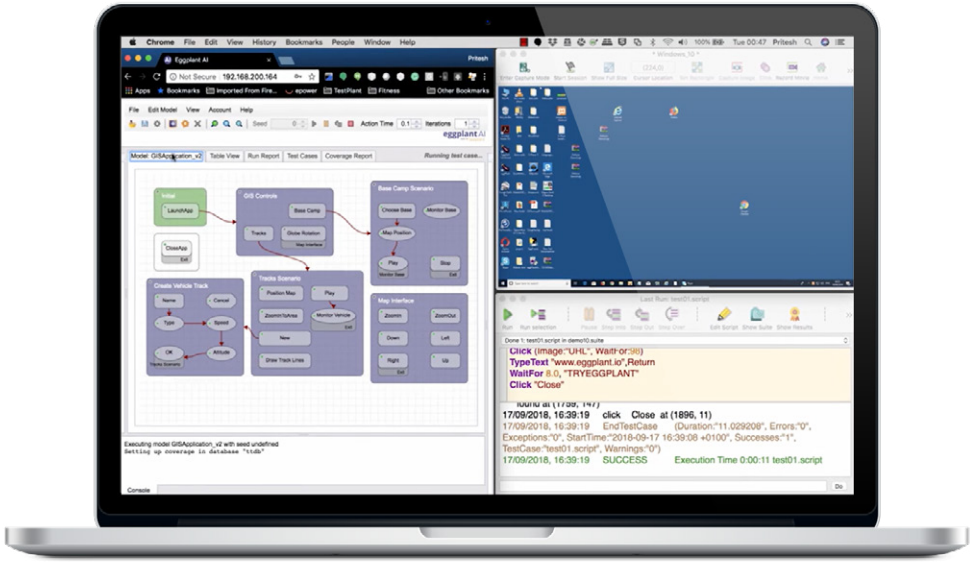
By having the ability to interact with a range of technologies, different UIs, and system integrations, the capabilities of Keysight Eggplant far exceed manual testing and other tools that solely rely on verifying the codebase. Here are the reasons why Eggplant is superior:

Model-based approach

Eggplant enables you to auto-generate user journeys spanning different devices, browsers, and operating systems. This model-based approach provides complete coverage for all user journeys and eliminates the need for manual testing. Eggplant reduces test maintenance and accelerates releases using the same test snippets across all devices, browsers, and operating systems.

Uses cloud-based device farms

Eggplant's flexibility supports easy integrations with cloud-based device farms. QA engineers can quickly scale and increase test coverage while reducing the cost and operational burden of maintaining physical devices. The integrations enable you to remain up-to-date with the latest mobile technology to ensure every test is relevant by testing a wide range of devices from anywhere in the world.

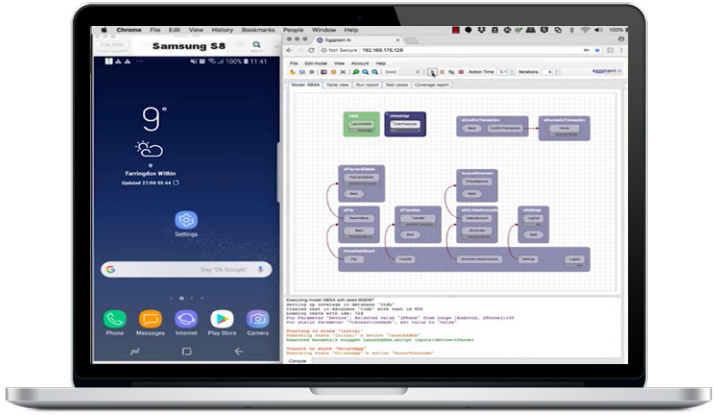


Simplify end-to-end testing across every technology layer

Eggplant is a solution that combines multiple technology layers and tests them together. QA teams need only deal with a single code layer to test the backend, through the object layer, to the front-end UI to ensure faster software delivery.

AI-assisted automation

Eggplant's AI-assisted automation generates essential and critical user flows throughout a mobile app application to ensure maximum test coverage. You can also auto-generate future test cases based on real user journeys. AI and ML enable auto-healing whenever there are changes, so every future test remains unaffected and robust.



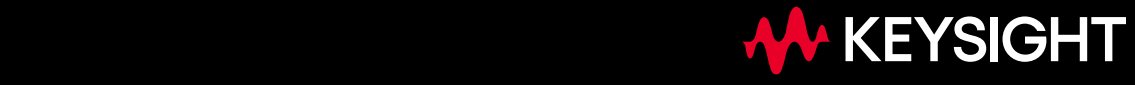
Test from the user's perspective

By interacting with a mobile app just as a human would, Eggplant ensures that every action a user can make functions as expected by testing what you think your users will do and everything they can do. Relying on code verification only is no longer necessary. You can validate the UI with Eggplant's intelligent computer vision and image-based testing that eliminates false positives.

No code / low-code approach

Eggplant's no-code / low-code approach bridges the gap between IT and the business, enabling anyone to create testing models regardless of technical ability. People across your organization can click through a model to design a user journey, and the test flow auto-generates, making software testing available to all.

Accelerate your software releases by future-proofing your testing and delivering high-quality mobile apps to users. Contact Keysight today for a free trial or in-depth demo.



Keysight enables innovators to push the boundaries of engineering by quickly solving design, emulation, and test challenges to create the best product experiences. Start your innovation journey at www.keysight.com.