WHITE PAPER

7 Best Practices for Optimizing Mobile App Experience

ROB KWOK, CTO and Co-Founder

Developing a mobile app that delivers a great user experience can be challenging. Even if your app performs well during testing, the complex environment of carriers, fragmentation among devices, and constantly changing operating system versions make it difficult to deliver a consistent, delightful user experience. Some companies, disappointed with a lack of business success resulting from their mobile initiatives to date, are being forced to re-evaluate their mobile strategies. A recent Apteligent study found that mobile business success is highly dependent on a set of user-centered app metrics that are tracked and trending positively. The companies that have avoided this disappointment focus on user-centered metrics and build specific steps into the app development lifecycle to achieve continuous improvement.

In this article, we've put together seven best practices collected from some of our most savvy and successful customers. These best practices encompass both tracking progress over time (week-over-week, across app versions), as well as applying repeatable processes to every sprint and release of an app. Applying these best practices repeatedly over weeks and months has enabled them to deliver 5-star mobile apps.

The most successful apps track two key metrics for crash rate: an overall crash rate and a crash rate in the most important user flows within the app.

Thinking about Mobile Experience from Your User's Perspective

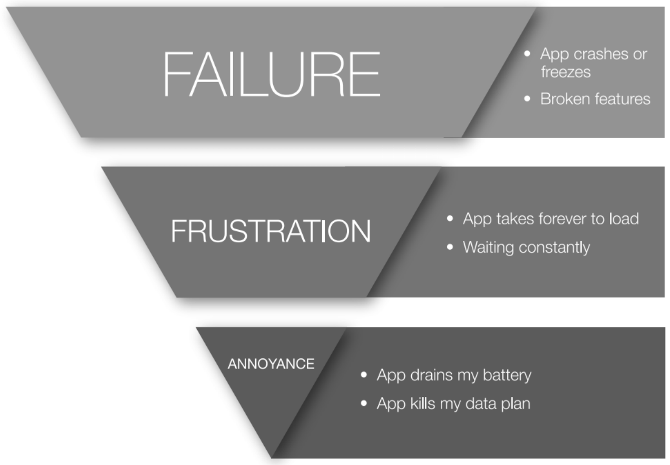


Figure 1: "The Hierarchy of Abandonment"

Delivering a great mobile app experience requires buy-in from every part of the organization to put the user first. One common refrain heard from companies that develop successful mobile apps is, "we think about customer experience from our user's perspective." We call the graphic shown in Figure 1 "The Hierarchy of Abandonment." It's a powerful framework that allows product development organizations to consider performance issues that are affecting user experience, then effectively prioritize those issues. As you read these best practices and consider your own development practices, use this framework as a new lens through which to view the performance of your app and how it impacts your users' experiences.

Avoid Failure

The best mobile teams ensure their app doesn't fail their users, first and foremost. This means the app does not freeze or crash, especially during key moments during a user's session. It also means making sure users are able to complete core actions in the app such as login or checkout.

One of the most important metrics these companies track is its crash rate: the percentage of app sessions that experience a crash. The most successful apps track two key metrics for crash rate: an **overall** crash rate and a crash rate in **the most important user flows within the app**.

There are two metrics that are critical to measure and improve: app load time and user interaction time during key flows in your app.

Best Practice #1: Fix the top ten crashes each sprint

One of the challenges with developing a mobile app can be deciding how much time to devote to developing new features vs. fixing bugs. While it is often tempting to allocate 100% of a release to shipping new features, fixing prominent bugs that are failing your users may be a better bang for the buck. The best teams allocate time every sprint to fix the ten crashes affecting the largest number of users in their app.

According to Apteligent data, the best apps have a crash rate of less than 0.25%. However, 75% of apps across iOS and Android do not meet this standard. How did apps like Groupon get to a crash rate below 0.02% and 4.5 stars in the app store? Their success was largely due to a disciplined and data-driven sprint process. Every sprint, they created a list of the ten crashes that affected the largest number of users and fixed them. With each release, they were able to continue to lower their crash and increase their ratings in the app store.

Best Practice #2: Maintain a crash rate of less than 0.25% in your app's three most critical user flows

Whether it's login, new account registration, or checkout, every app has flows from screen to screen that are critical to the success of the app. First, identify the top two or three flows in your app that are critical to improving business metrics (e.g., monthly active users, retention, or revenue). Then, for those key flows in your app, monitor any performance issues that are causing failures. Crash rate across each of these flows is an app metric that we see teams measuring and striving to improve over time. If your organization falls short of having an overall crash rate of 0.25% or less, you minimally must maintain a crash rate of 0.25% or less in your top three flows. Prioritize the top crashes that occur in these flows to ensure the most impactful bugs are being fixed.

In summary, to optimize the crash rate of your most critical flows in your app, apply the following process:
1. Identify the three most important flows for your users in your app e.g., Login New Account Registration, Checkout
2. Fix the top crashes affecting those critical flows every sprint
3. Maintain a crash rate < 0.25% in your most critical user flow

Prevent Frustration

In addition to monitoring failures due to crashes in critical flows, it's also important to make sure your app doesn't frustrate your users with long wait times or a sluggish feeling UI. Since mobile apps are typically used in short, fleeting moments while commuting to work or waiting in line at store, every interaction in the app matters. There are two metrics that are critical to measure and improve: **app load time** and **user interaction time** during key flows in your app.

Best Practice #3: Ensure your app loads in under two seconds

In addition to crash rate, most of our customers track app load time as a core performance metric of their app. App load time is a metric that is often highly correlated with user engagement in mobile apps. The longer it takes for your app to load, the less likely a user will use your app. In fact, studies have found that 50% of consumers consider app load time a major source of frustration and 25% would leave a brand due to unacceptable load times. Despite this research, almost half (46% of iOS apps and 53% of Android apps) take over two seconds to load. No wonder some organizations are re-evaluating their mobile strategy; they're not monitoring the right app metrics.

The best mobile teams monitor app load time as a leading metric of traditional key performance indicators comparing each release to the last – and each week of data to the previous week. Changes to an internal API, the addition of a third-party SDK, or a release of a new operating system can drastically affect app load times and, correspondingly, business metrics. In fact, one of our customers discovered that a 10% decline in in-app purchases within their app was directly caused by a one second increase in app load time when Apple released iOS 9.

Best Practice #4: Ensure user interactions feel responsive

Every time a user interacts with your app, they expect your app to "feel responsive." In a mobile app, there are two types of user interactions: an interaction in which a user expects the app to respond immediately and interactions in which the user is expecting a short delay because they perceive it to be working to extract or process data.

Applying the research summarized in Jakob Nielsen's article on user interface design, "Powers of 10," if the UI takes over 0.1 seconds to respond to an interaction, users will perceive the UI as delayed and not "instantaneous."[1] If the UI takes over 1.0 second, users will start to lose both their train of thought and the feeling of being in control of their interactions.

To create a responsive mobile app, measure both those user interactions in your app that should complete immediately and those where the user expects a short wait time. Start with the interactions that are the most common in your app and the ones that the most important flows depend on. Measure the time it takes from when a user interacts with the app (i.e., taps on a button) to the time when the UI gives the user a valid response (i.e., the screen is finished loading and the app is available to accept new user interactions). If any of these interactions take longer than expected, analyze the events and network calls that are slowing these down, create tickets, and fix them. A set of common user interactions and their target interaction times are shown below in Figure 2:

Userflow	User Expectation	Time
App load (time to availability of first use -interaction)	App is “working”	1.0 s
Login (not including data entry)	App is “working”	1.0 s
Registration (not including data entry)	App is “working”	1.0 s
Search	App is “working”	1.0 s
Screen transitions	Immediate	0.1 s
Browse	Immediate	0.1 s
Add to shopping cart	Immediate	0.1 s
Checkout	App is “working”	1.0 s
Locate (e.g., locate store)	App is “working”	1.0 s
Barcode scan	App is “working”	1.0 s

Figure 2: Common User Interactions and Target Interaction Times

Best Practice #5: Monitor network calls during key flows

Many interactions on mobile depend on network calls to load data or perform a query. Games often depend on network calls to load images or sound files for a new level. E-commerce apps often use network calls to search their store inventory or make a purchase. All apps that allow users to log in or create an account rely on network calls to perform these functions.

As you’re optimizing user interaction times in your app, it’s important to monitor the network calls these interactions depend on. For these network calls, monitor and improve these two metrics:

• **Latency:** the amount of time it takes for a call to complete (how long a user is waiting)

• **HTTP Error Rate:** how frequently the call fails either due to a server returning an error or the app’s inability to connect to a server

Since interaction times often depend on these network calls, make sure that the latency of calls that occur during a critical flow are less than one second. If a network call is taking longer than a second, work with the API owner to reduce the latency of that endpoint by either optimizing the processing time, using a different API, or changing the network call to send/receive less data. The error rate of these network calls should be treated similarly to the crash rate in your most important flows. The error rate of calls to these endpoints should be less than 0.25%. Set alerts for key endpoints so that if the error rate goes above 0.25%, you are notified right away.

To optimize the endpoints your app’s most critical flows depend on, apply the following process:

1. Monitor the three most important flows in your app

2. Time how long they take to complete

3. Identify network calls that each user flow depends on

4. Monitor the latency and error rate of each call

• Latency < 1 seconds

• Error Rate < 0.25%

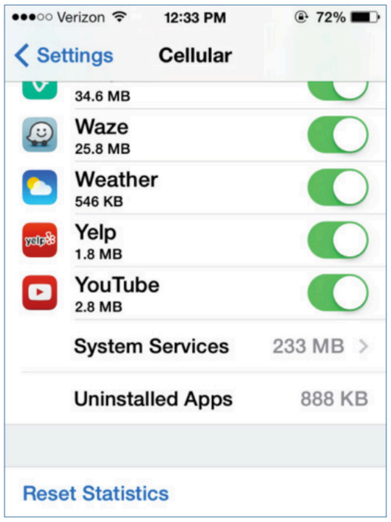
The final two metrics to monitor in your app are data usage and battery drain. While they do not immediately affect a user's in-app experience, it's important to measure these metrics to ensure users do not uninstall your app or leave a 1-star review in the app store.

Avert Annoyance

The final two metrics to monitor in your app are data usage and battery drain. While they do not immediately affect a user's in-app experience, it's important to measure these metrics to ensure users do not uninstall your app or leave a 1-star review in the app store.

Best Practice #6: Measure data and battery life usage

As consumers have become more dependent on their mobile phones, experienced overage charges with their carriers, and been stranded with a dead battery, they've become more careful about which apps to use. Both Google and Apple have responded to consumer demands to provide more transparency about what's eating up battery life and data. For example, Apple now provides utilities in their system settings that tracks the amount of data each app uses (see Figure 4) as well as battery life, and Android has a built-in battery (see Figure 5) and data usage tools as well. If a customer sees an app that's consistently draining battery or using up their data plan, they will uninstall that app. Even Facebook's app has been scrutinized for draining the battery and causing other apps to launch more slowly.



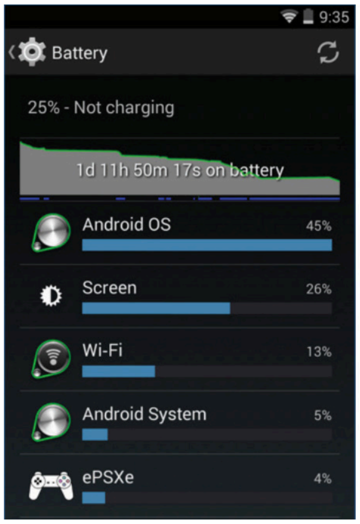
●●●○○ Verizon 12:33 PM 72%

Settings — Cellular

34.6 MB	
Waze 25.8 MB	
Weather 546 KB	
Yelp 1.8 MB	
YouTube 2.8 MB	
System Services	233 MB
Uninstalled Apps	888 KB

Reset Statistics

Figure 3: Apple's Built-in App Data Usage Tool



9:35

Battery

25% - Not charging

1d 11h 50m 17s on battery

Android OS	45%
Screen	26%
Wi-Fi	13%
Android System	5%
ePSXe	4%

Figure 4: Android Built-in Battery Tool

The most common cause of battery drain and data usage are network calls, so our customers keep an eye out for an increase in the number of network calls or the amount of data sent and received for each call. Before pushing a new release of their app to the app store, monitor the amount of data your app sends and receives. Third-party SDKs can also often be very chatty; before adding a third-party SDK to the production build of your app, do your homework and monitor how frequently calls are being made and how much data is sent with each request.

Summary: The Mobile Metrics You Should Optimize

Best Practice #7: Monitor your mobile app metrics over time and per release

In summary, we found the best apps are managed by teams that take a very data-driven approach to optimizing user experience. They measure key metrics such as app load time and crash rate, and ensure these metrics are improving over time and with each new update to their app. To prioritize what they work on each release, they look at the three most critical user flows in their app and fix the crashes or slow network calls during those flows. Finally, they safeguard against users uninstalling their app due to battery drain or data usage by monitoring how many network calls their apps send and by testing SDKs before adding them to their app.

• **Avoid Failures**

- Overall Crash Rate < 0.25%

- Crash Rate of Critical Flows (e.g., “Login”) < 0.25%

• **Prevent Frustration**

- App Load Time < 2 seconds

- Critical Flows < 2 seconds

- Flow-Dependent APIs: < 1 second latency, < 0.25% error rate

• **Avert Annoyance**

- Monitor Request Volume and Data Sent/Received

About Apteligent

Apteligent is the App Intelligence company trusted by the largest mobile apps in the world. Apteligent’s software provides actionable mobile app insights to improve digital business on iOS, Android, and Hybrid apps. Product managers and developers use Apteligent’s insights to diagnose app performances issues that impact user experience. The platform collects and analyzes app performance issues and connects problems to key business metrics. Mobile teams also have access to Apteligent’s big data platform, as well as industry and app benchmarks. Apteligent is based in San Francisco.

Learn more at www.apteligent.com.

FOOTNOTES

1. Nielsen, Jakob. “Powers of 10: Time Scales in User Experience”. Oct 5, 2009. Nielsen Norman Group.