Why Crash Analytics Matters

Mobile crash analytics bolsters crash response times and ultimately improves user experience.

FLURRY

Why crash analytics matters

When it comes to maximizing mobile app profitability, all of the feature tweaks in the world can't overcome fundamental software quality problems. No matter how much a developer spends on user acquisition, engagement, or monetization, none of it matters if the app keeps crashing on user devices.

When mobile apps don't work as intended, they're not only failing to drive revenue as expected, they're also churning out users at record rates. What's more, frequent crashes kill mobile developers' growth prospects. One too many crashes is just the thing to push an aggravated user into dropping a nuclear bomb on user acquisition by leaving a one-star review in the app store.

Bottom line, poor response to crashes can put mobile developer ROI at risk. Fortunately, mobile crash analytics can provide developers with an insurance policy against this risk.

Poor response to crashes can put mobile developer ROI at risk.

Just as organizations use analytics to segment users and squeeze better profitability out of functionality, they can use mobile crash analytics to bolster their crash response times and ultimately improve user experience.

Here are the capabilities they need to look for and the practices they need to hone to get the most out of mobile crash analytics tools.

Art and science of mobile crash response

Finding crashes and their root causes—and doing it at scale—is no simple task.

In order to know whether or not their apps have crash-causing bugs in them, developers first need a mechanism to trigger alerts when users experience a crash. But beyond just knowing whether or not an app has crashed, the developer also needs enough diagnostic data to figure out the root cause of the bug and to fix it.

It's a little bit like reading tea leaves—the more information about a crash instance that developers can get their hands on, the better picture they can form about the technical or code conditions that led to a crash. There's both a science and an art to this tea reading process.

The developers need to be systematic and scientific about the kind of data they plan to collect in advance to support their future response efforts. But at the same time, sifting through that data in the heat of an incident takes a fair bit of artistry and craft to elegantly solve the problem.

Now, crash response is hardly a new field for developers; however, many of the longstanding tools available in the desktop world don't work in the mobile environment.

Do crashes hurt your app's ranking in the app stores?

Frequent crashes and overall app performance issues do impact your app's rankings.

Mobile crash analytics are different because the mobile OS has really constrained what an app can natively report back to developers for both security and performance reasons.

A lot of the crash reporting mechanisms developers can build into desktop software are not possible within a mobile app. Mobile OS crash logs only provide a very limited set of information about crashes, and may not even report back every kind of crash caused by an app.

And that can make the art of tea reading a whole lot harder for mobile developers.

What Apple and Google's mobile OS crash logs do and do not report

Both Apple and Google automatically report when apps crash into their respective developer consoles. But the data is limited and often difficult to download and parse.

- ✓ Record code problems within the confines of the app's virtual machine
- ✗ Do not include information about crashes that involve separate processes
- ✗ Poorly record crash classes occurring within native code
- ✗ Often miss crashes involving termination due to performance issues that cause a phone to freeze

What mobile developers need to effectively parse crash reporting

The limitations of Apple and Google's native mobile OS crash logs are why it is so important to put a third-party mobile crash reporting framework in place.

The holy grail of debugging is having the means to completely recreate a bug in order to isolate its cause. But to achieve that takes advance planning and laying down some technical groundwork before a crash ever occurs.

In order to put effective mobile crash analytics at the fingertips of the development team, mobile developers must layer in a robust reporting framework that'll record the right data when things go wrong.

Key functionality to add to mobile crash reporting framework

Capabilities for building in breadcrumbs

Breadcrumbs are code-level event data—say the user pushed a certain button or went to a certain screen—recorded to tell developers everything they ever wanted to know about execution of code outside of what's recorded in the stack trace. Crash management frameworks should provide the opportunity to drop breadcrumbs where necessary.

Journey to crash recording mechanisms

The system should be able to take these manually added breadcrumbs and additional application analytics to record an event path to crash that's as complete as possible.

Remote configuration

Mobile developers can take advantage of remote config to change any number of elements within a user's app without pushing updates to the app—including the richness of app event data recorded for crash analytics. This can make it possible to create targeted testing groups from whom developers are collecting a higher level of crash analytics.

As developers deploy these features, they must consider carefully how they design the 'eventing' of an app and where the breadcrumbs need to go in order to enable swift crash response. The product manager should determine what they want for analyzing behavior in the app and the technical leads should determine the technical events they want recorded so that the analytics are properly instrumented when the crash occurs.

Managing mobile crash volumes

Some of the biggest challenges that mobile developers of highly popular apps face is dealing with crash response at scale. When crash volume is heavy, it can be tough to figure out where to start with remediation efforts. Ideally, developers want to first fix those crashes that the most people are experiencing, not the ones that only one or two people are experiencing. Managing mobile crash volumes depends on two important capabilities: prioritization and crash grouping.

Prioritization

As developers scan through their crash alerts, they ideally should be able to see these crashes ranked by number of users experiencing them and by version of the application in which they are occurring. This enables developers to start filtering crashes and figuring out which are most important to fix first.

After remediation, one of the most important filters to monitor is the version number of the app in which a crash fix was made to make sure the fix corrected what it was supposed to correct. If the version still shows the crash happening, then it means the team needs to keep digging for a root cause.

Grouping crashes

One of the things that a team needs to think about is how the app framework groups crashes. Group them too narrowly—seeking something like an identical line-for-line similarity—and the developer ends up with what appears to many different crashes that in actuality have the same root cause. But group crashes too broadly and the system will then potentially hide two different crashes within the same group. The point is that grouping techniques are important to consider as a team builds out its mobile crash analytics program capabilities.

Getting the most out of mobile crash analytics

Building out the right mobile crash analytics technical capabilities is just step one in getting the most out of mobile crash data. Mobile developers should also think strategically about how they'll respond to mobile crash analytics on a day-to-day basis.

Think about duties

A smart crash analytics strategy will start first by breaking down duties in advance to decide how the team will respond to crash incidents.

Some factors to consider or determine in advance will be:

- Who monitors crash alerts each day
- Who responds or assigns response and remediation work based on crash analytics findings
- What kind of situation will trigger an 'all-hands-on-deck' type of crash response

How formal the team needs to be in documenting the crash response process depends on the size of the organization. Smaller organizations may just need a verbal agreement on the breakdown of duties. Larger orgs may want it written and shared via a wiki or similar channel.

Integrate with issue tracking

Simply having mobile crash analytics available can be a big boon for crash response, but the best dev teams get the most out of it by putting that data into the tools and processes they use for bug tracking. Effective mobile developers should find a process or automation to tie crash reporting to bug tracking and to engineering comms channels, such as commonly used Slack channels.

Ask users for help

Consider developing a button to do a support dump with user permission. Loyal users will want to help however they can if that means they can get back to using your app as intended—and such an information boost could aid greatly in debugging.

Set up test groups and debugging environments

Create two separate environments or applications because debugging is a lot easier in a dev environment. Also consider setting up a proper test group to carefully monitor crashes before releasing the app widely. While this will not eliminate all crashes, because the test group is small relative to the full audience, it is still a best practice that will help identify potential crashes.

Find the root cause of the root cause—post mortems for the worst crashes

Engineers love to dive straight into the code and knock out the root causes of individual crashes. But sometimes, they need to step back and think about the big picture. In QA, mobile crash analytics is the last line of defense for quality. But it can also be the first clue for long-term process improvements if the team takes the time to do post-mortems for the worst crashes to figure out why they happened and how to keep them from popping up again.

Effective mobile crash analytics takes a lot of work and many moving parts. Your teams will get better the more they execute on these strategies. The goal is to tweak strategies as you learn what's best for your org in order to drive crash rates down to as close to zero as possible.

Important mobile crash benchmarks to track

These are good metrics to track the efficacy of your mobile crash analytics program.

Crash rate

Normalize crash volume by how much the app is being used

Cold-start time

How long does it take before user can interact with the app from when they launch it

Unresponsive rates

Instances where app becomes unresponsive, but not quite long enough to trigger the OS to kill the app

Keeping customer top-of-mind

Remember that this exercise of improving mobile crash management is all about bolstering the business viability of the app. So don't forget the customer in all this.

Customer communication is crucial for setting expectations and gaining the cooperation of users to improve the app rather than giving up through uninstalls and poor reviews.

Good communication doesn't happen magically. Mobile developers have to actively foster it, starting first by establishing and publicizing a channel—by email or directly through the app—for customers to report problems. This channel should be well monitored—otherwise it could cause more harm than good in the relationship with customers. There's no bigger sin than sending motivated and proactive users to a communication black hole.

Not every user is going to reach out directly, so developers should also think about how they can broadcast bigger issues that they're currently working on. One way to do this is to build a message of the day banner into the app that can also be used as an outbound channel for big crash issues.

The more you can let customers know you're actively working on problems, the more patient they're likely to be as you chug along toward a fix.

Conclusion: keeping mobile crash analytics in perspective

While it's crucial for modern mobile developers to leverage user analytics to drive user engagement and monetization, they've got to be careful not to miss the forest for the trees. Mobile crash analytics also plays a big part in squeezing the most ROI out of apps and maintaining strong market presence.

Ultimately, organizations need to remember that the best crash is the one they never had. When done right, mobile crash analytics can play an important role in boosting the overall quality of an application. The best in the business bring a zero-crash mentality to the table. They never give up on improving their crash rate—and they fight the temptation to simply accept a certain level of crashes. Whenever they have spare cycles, they go back and nail down lower priority problems after the big issues have been fixed.

At the end of the day this creates a premium user experience and the type of conditions developers need to maximize their profitability.

How Flurry can help

Flurry crash analytics & reporting

- Real-time crash reports
- Crash alerts
- Journey to crash
- Breadcrumbs
- Remote configuration
- Jira & Slack integration
- Mapping files upload
- Detailed stack trace

Get crash alerts in real time and identify the root cause to plan for the right corrective measures.

Visit **www.flurry.com/crash** to learn more about how Flurry can help you jump start your mobile crash analytics program.

FLURRY