EP 13: Building quality apps with Firebase

[00:00:00]

[music]

Maria Neumayer: I think if you're the only app out there that does what you're doing, you're probably **[00:00:10]** okay, but only for a while until somebody else comes around and does it better.

Shobhit Chugh: By the way, I would love that additional job of testing the app by ordering food over the weekend. **[00:00:20]**

[laughter]

As people's apps get more complex, definitely they have had to step things up from their processes, **[00:00:30]** just how closely they're monitoring app quality.

Tamzin Taylor: Yes, because I imagine that the last thing you want is someone not getting their dinner.

Maria: Exactly. We don't want people to go hungry.

[chuckling]

[00:00:40]

Tamzin: First impressions count and over the last few years, users expectations have increased, as many companies realize that app users are highly engaged. **[00:00:50]** They're investing more into their apps usability and technical capabilities as a result. What does this mean for you? Well, if your app crashes, hangs or chews up battery **[00:01:00]** life, there's probably a competitor out there who can provide the same experience, but much, much better. Today, we're discussing how to make your app best in class, and free up the users' attention **[00:01:10]** to get the most out of your product or service, which really matters.

Dirk Primbs: Hello, and welcome to another episode of the *Apps, Games, and Insights* podcast. We're your host Dirk Primbs and Tamzin Taylor, both **[00:01:20]** working from Google. In this episode, we'll be discussing the importance of app quality and why developers should make it a priority. Joining us today is Maria Neumayer, **[00:01:30]** Staff Software Engineer at Deliveroo, and Shobhit Chugh, Product Manager for Firebase Crashlytics.

Tamzin: Welcome both.

Shobhit: Thank you. Glad to be here.

Maria: Thanks. **[00:01:40]** Good to be here.

Dirk: So cool to have you. App quality is such a huge topic. Let's start with something let's say more manageable by introducing our guests properly. **[00:01:50]** Tell us a little bit about your background and your role. Maria, do you want to start, maybe?

Maria: Sure. I'm Maria. I'm an Android Engineer at Deliveroo. I've been **[00:02:00]** working on writing Android apps for over 10 years now. Focusing on quality as well and making sure our users always have a good **[00:02:10]** experience no matter where it is.

Dirk: Excellent. Shobhit, you're with the Firebase team, right?

Shobhit: That's right. I work on Firebase Crashlytics. I have for the past **[00:02:20]** almost four years now. Firebase is a suite for developers to build mobile and web apps. Whether you need back end infrastructure that **[00:02:30]** you obviously don't want to build on your own or you need tools to understand how your users are doing, how your app quality is to remotely change the behavior **[00:02:40]** of your app without shipping a new release. It's a comprehensive set of tools to help support you.

I actually came into Google with the acquisition of fabric, which became part of **[00:02:50]** Firebase. My past background, I was an engineer several eons ago. Did some management consulting and then moved into product **[00:03:00]** management.

Tamzin: Nice. App quality is really broad, but when I think about quality, I think of all those times I've tried to use a banking app, and it crashes. **[00:03:10]** I get so angry because I had no other way of getting to my money.

Dirk: That would make me angry too.

Tamzin: Well, yes, unless you forget your PIN number, which often happens to me as well, it's a great way to save. So **[00:03:20]** that we can put some context around this for the listeners, Maria, what does app quality mean to you?

Maria: App quality is a really broad topic. At the first start, you can think of just **[00:03:30]** how stable is your application? Or is the app crashing, as you said? Can you actually use it? Then there's a lot more to it. You have to think about the actual user **[00:03:40]** experience. Can the user actually perform the task that they came in to do? You open your app and you have to try to figure out how the hell you can send money to **[00:03:50]** somebody. How does that work? Just making the experience easier for the user is really important as well when it comes to app quality.

Tamzin: Excellent. I've really just thought, I'm also **[00:04:00]** trying to pass my driver's learner theory test and their app is so awful. I'm sorry, if you're listening out there, it needs some UX help. It needs **[00:04:10]** some onboarding instructions, a simple thing like that. I hear what you're saying, Maria, a good point. Shobhit, what does quality mean to you?

Shobhit: Very similar. For me, it's a holistic view of the **[00:04:20]** customer experience. That could be ease of use, ease of navigation. Can you get to the thing you use most often quickly? **[00:04:30]** How's the UI? Are there any bugs that keep happening? Of course, if there's crashes, that's a huge problem or the app freezes, or **[00:04:40]** there's performance issues. You're on a slow network, and now you're waiting 15 minutes, maybe 20 seconds for the app to load. All those things are a part of app quality.

Dirk: To a **[00:04:50]** large extent, it's basically if the app matches the user expectations, and ideally surpasses them. It's like, I'm expecting a nice app that **[00:05:00]** works. As soon as I'm actually getting my money faster from my banking app than I anticipated, and that's in a safe and clean way, then at least Tamzin is happy already.

Tamzin: I would be happy. **[00:05:10]** Much more happy if it replicated my money. Maria, tell me just some of the listeners out there. What's a minimum bar to think about if you're thinking about app quality?

Maria: That's a good question. I think it probably **[00:05:20]** depends on the user. Some users are going to be very happy using a really bad app, but others are going to be expecting a lot more from you. I think you need to make sure **[00:05:30]** that you figure out what your user base is. If they're actually expecting a really high-quality app, what are your competitors are as well?

I think it's really **[00:05:40]** important to figure out where your baseline is, what you want your app quality to be like, and then make sure you sustain at that level so you don't fall into the **[00:05:50]** trap of your app quality slowly decreasing. Then at some point, someday, you need to go in and spend a lot of time and getting it back up because you're actually losing your users.

Shobhit: I just want to add **[00:06:00]** one thing, Maria, I think people's expectations are rising, not just based on the competitors you have, but the other apps that they experience. They now want the banking app **[00:06:10]** to be as good as any consumer app that's out there.

Maria: That's definitely true. I think if you're the only app out there that does what you're doing, you're probably okay, but only for a while until **[00:06:20]** somebody else comes around, and does it better.

Tamzin: I used to work in a company that built apps, and you'd get a whole bunch of bugs coming through, but then you'd have all these new feature requests. **[00:06:30]** We always struggle with, well, how do you get the decision-makers to prioritize fixing the bugs versus launching some shiny new feature? How do you **[00:06:40]** approach that? What are your tips for some of the listeners?

Maria: I think that is one of the most challenging things to do because I think app quality is one of those things where, as long as it's okay, and **[00:06:50]** users are mostly happy, it's not going to be a big problem, but at the point where the app quality drops, and it becomes an **[00:07:00]** emergency, then you're in a really bad position. Talk to your leadership. Make sure you inform them of the challenges and the risks of not **[00:07:10]** actually focusing on this. What the implications are that if we don't focus on this, maybe at some point, none of our users can use **[00:07:20]** the application.

I think that's mostly more important on the backend side, where if your services can’t handle the load anymore, and users actually can't use your **[00:07:30]** application anymore at some point, and nobody can use the application at that point, it's really bad. Even from an application standpoint, if you've got a bug or **[00:07:40]** something that stops a user from doing something, or your app is crashing to 10% of your users, if your user base grows, then those 10% of the users can actually **[00:07:50]** be a big number of users. Keep monitoring the problem, see how big of a problem it is.

Then as it grows, you can look at **[00:08:00]** where it's really important that you focus on it and where it may be okay. One thing we did at Deliveroo, is we set up a baseline **[00:08:10]** for what our minimum crash rate or our maximum crash rate should be. Then, once we got above that, we would **[00:08:20]** say, okay, we need to start fixing a problem. If we were below that, it's okay. We're below our rate of crashes that we deem as acceptable **[00:08:30]** and it doesn't have to be fixed right now, we can wait and then we

can fix it for the next release. If we go above that threshold, we will do something right now **[00:08:40]** and we will fix it for the user. Defining what that level is important.

Tamzin: It sounds like we're saying is that it's very much data-driven.

Maria: Definitely. **[00:08:50]**

Dirk: It very much cuts both ways. It's like in one way you frustrate your users and you may end up losing fans or users or worst case, both. **[00:09:00]** On the other hand, bad app quality may very quickly be a security hazard for your systems. May be something that not only hurting your reputation, but very clearly, **[00:09:10]** your operations. It may cost a massive amount of money if it uses additional loads on your servers and whatnot. There are plenty of reasons I would say, if you are looking **[00:09:20]** at it from a data-driven perspective to make this a really, really strong business case.

Maria: Definitely, I think it also helps not falling into the trap of **[00:09:30]** keep talking about the problem and figuring out, do we have to do something? Do we not? You stop the discussions, you know immediately just by looking at the data. Is it a problem **[00:09:40]** that needs to be fixed now or can we wait? You don't spend a lot of time on actually triaging your bugs and talk about the problem rather than actually **[00:09:50]** fixing it.

Tamzin: When we're talking about being able to show data to your management to say, look, this is why we should be setting some limits on what we think is acceptable crash rates and **[00:10:00]** here are the levels at which we want to escalate and prioritize. How does Firebase help developers do that?

Shobhit: In terms of showing the suite, very often developers use **[00:10:10]** one of our top-level metrics. They might, first of all, just look at user metrics, such as, how's my retention doing right now? Or am I getting new users? **[00:10:20]** But then there's specific metrics that are related to performance, such as what's your app startup time? Or, for example, what's your **[00:10:30]** crash-free users? Probably our best-known metric. How's that for this release versus the previous release? All of that is tracked, and you can see it right away. That's often **[00:10:40]** what developers use to show it to their management.

Tamzin: Excellent. I guess, if you were a fast-growing organization, and you just want to put on loads of customers very quickly, your emphasis might **[00:10:50]** be just making sure that the app is running and that you're still able to transact these transactions. What are some of the pitfalls that developers face when they do grow too quickly? How can they mitigate **[00:11:00]** those around quality?

Shobhit: From our perspective, there's a couple of things that are really, really useful. The first one is really knowing upfront, **[00:11:10]** what are things that you're going to look at to make decisions on? Do I ship this next release out or do I need to roll it back? Things **[00:11:20]** such as, crash free user rate, for example. That's one pitfall developers should also look out at. I think as you are growing, one of the things **[00:11:30]** that becomes a challenge is all those manual touch points in that you might have in shipping your app, that often just becomes a case **[00:11:40]** where if somebody missed something, it's a big problem.

We encourage developers to, first of all, just as you're growing, initially, to manually keep **[00:11:50]** on top of things, but then start to automate a lot of those things and make sure that you don't have to have a person in the middle who's responsible for making such **[00:12:00]** crucial decisions every day. Then one day, just in the hand of the person didn't

happen to be there and you ship out a bad release, the consequences of this are pretty bad for **[00:12:10]** your app's usage that day, and then even ongoing customer retention. We definitely encourage people to start to automate some of those processes over time.

Maria: That's definitely something I **[00:12:20]** agree with. I think, in the beginning, we had a hugely manual process of releasing our app ties. At the beginning, when I joined, we had a whole **[00:12:30]** meeting about the release, and are we going to roll out or are we not. Over time, we removed all those steps and made sure **[00:12:40]** the release process is as easy and streamlined as possible. It used to take us more than an hour or something to actually do a release. Now, **[00:12:50]** it's just somebody runs a script somewhere, and everything happens automatically.

We still have a bunch of manual steps to actually roll that out and monitor that, but we're trying **[00:13:00]** to automate that right now. Hopefully, end of month or two our process will be fully automated and it won't require an engineer to babysit **[00:13:10]** that release every week or every two weeks.

Tamzin: Maria, when you talk about some of those manual steps that you've removed, can you give some examples of what they were so that if people back at home **[00:13:20]** were listening can go, "Oh my God, we did that as well. I didn't realize you could automate it."

Maria: Some things we improved were around strings, so translations. Making sure **[00:13:30]** we can just download those translations and not having to actually go in and download the translations, check if they're okay, then push the translations, **[00:13:40]** merge them, and then actually cut to release. That's something that took a long time. Partly, it was because our tooling wasn't great **[00:13:50]** and we actually had to check those strings manually. We wrote a few lint checks, which actually would check if the format was correct. We would rely on those **[00:14:00]** rather than somebody manually having to check are all the placeholders in place or is there a space somewhere they shouldn't be?

Automating all of those steps is really useful. **[00:14:10]** Then just actually doing the release process. We had a long-form document that outlined all the steps you had to take. It was just **[00:14:20]** all exactly the same every time. It was just copy and pasting all of those commands and saying, now you check out this branch and now you push this and we just automated all **[00:14:30]** of this, created a script.

Now it's just literally somebody running this script, and everything happens automatically. It checks out the project, it **[00:14:40]** creates a tag on GitHub, it pushes it, it uploads it to the Play Store, or it uploads it on a CI and CI then runs it and publishes on Play Store. Everything **[00:14:50]** is just happening automatically and we don't have to actually do all of those manual steps anymore.

Shobhit: It's been a team from some of our most advanced customers, this automation piece. **[00:15:00]** In fact, last year at Firebase summit, we had Spotify, do a talk with us, which is now on YouTube, where they spoke about exactly how they started to automate their **[00:15:10]** release as well. Especially take out the pressure from that one person who was responsible for monitoring that release. Imagine the pressure like, **[00:15:20]** hey, what if I did something wrong and something else, all of that just gone? That's been dramatically useful for them.

Maria: I definitely remember having to do releases just **[00:15:30]** before going home because we had to wait for something to be merged first. We used to do it together with

another person, so you wouldn't miss anything. It would even take **[00:15:40]** two engineers' time.

Tamzin: Hopefully, you weren't doing it on a Friday night.

Maria: Yes.

Shobhit: [chuckles]

Maria: Exactly. Don't ship on a Friday night. [chuckles]

Tamzin: Yes, indeed. I often **[00:15:50]** recall 20 years ago working on stuff that required rollbacks over the weekend at 3:00 AM. It's not fun. Thinking about what you've just said, how you've made some optimizations **[00:16:00]** using automation. Taking a step back and thinking about the whole process of a good launch, what does that look like for you? What are the key factors that listeners would need to think about?

Maria: I **[00:16:10]** think the most important thing is to have good insights into your release. Really, knowing how it is doing. Having **[00:16:20]** the confidence of being able to ship in good quality. If you start a rollout and you're not entirely sure if a lot of users are going to **[00:16:30]** crash or something like that and you're worrying about every release, that's not a good situation to be in. Focus on increasing your confidence in the release. That can **[00:16:40]** be by having feature flags in place so you can disable things if something does go wrong, rolling out slowly.

Using the state rollout on the Play **[00:16:50]** Store, or having a beta testing group, having internal testing. Use your employees, if you've got a team using the app. Everybody's using it every day. **[00:17:00]** Make use of that and test things internally before you roll out. Test your code, write UI tests, integration tests, just increase your confidence in a release. **[00:17:10]** Then once you do roll out and there is a bug in there, you can disable it. You can hold a rollout, you can disable the feature flag, and everything's **[00:17:20]** okay. You can fix it before it affects too many users.

Shobhit: I'll add something to that, which is if things do go wrong, do you have mechanisms in place so that you detect things really **[00:17:30]** fast? Whether folks have had on-call systems set up so that somebody's paged and they can look into issues? Very often, people mention **[00:17:40]** this one Firebase Crashlytics feature called velocity alerts where they're like, hey, velocity alerts saved my release, and you really save the app because we got notified. Somebody got paged, **[00:17:50]** and we were able to react to it and do exactly what Maria was saying, which is turn off a feature flag or rollback the release or hold it, something like that.

Some of the trickiest things we've seen is when **[00:18:00]** there was a server issue, a back end issue, and that caused the app to start crashing. It was completely disjointed from the actual release **[00:18:10]** process. Those are some of the tricky situations where you need to have these things in place.

Maria: Also, watch out for your third-party dependencies. Somebody else might **[00:18:20]** have an issue and crash. Make sure you can disable those features as well in case something goes wrong because over that, you have no control.

Tamzin: A question for you, Maria, is that what are some **[00:18:30]** of your favorite tools in the play console or Firebase that he used to do these diagnostics?

Maria: We made use of the velocity alerts. It has definitely saved us a bunch of times, **[00:18:40]** we built upon Firebase, we export our data to BigQuery and then import it into our internal **[00:18:50]** tool so we can really set up the alerts exactly how we want them and trigger exactly at the point where we want them and really make sure **[00:19:00]** we know how well our releases are going. That's especially important as we roll out our releases. As you roll out the releases, you're not going to have a lot of data. **[00:19:10]** It can sometimes be a bit noisy.

Making sure that if you do roll out your release, especially if you do a staged rollout, if there is an issue **[00:19:20]** to get notified really quickly, so you can disable that release is really important. For us, that was really good to be able to set up our own alerts on **[00:19:30]** that, and really dig into the data and see what was actually going on. Actually, also, being able to figure out how **[00:19:40]** many users it was affecting. Sometimes errors get grouped together, but they're actually not the same. By digging into the data directly in BigQuery, **[00:19:50]** we could really see what was happening and how many users did it happen to and see what the impact was.

Tamzin: Yes because I imagine that the last thing you want is someone not getting their dinner.

Maria: [00:20:00] Exactly. We don't want people to go hungry.

[laughter]

Exactly.

Dirk: Do you have any mechanism to pick up on some **[00:20:10]** softer quality indicators? If the whole thing crashes, that's pretty obvious that's bad quality, right? I think even, to some extent, worse is when people **[00:20:20]** two-three times have a bad experience, that's not necessarily indicated by the app doing something crazy. Is that something you try to prepare for as well **[00:20:30]** and if so, how?

Maria: Another thing we do track is how many non-fatal issues there are. Where there's something where does a bug, but it isn't that bad to **[00:20:40]** make the app crash. Somebody is blocked from doing something, for example, we do monitor those as well. We have also an alert in place in case those go up. **[00:20:50]** If there's some error happening to a lot of users, we get notified and we can fix it. We also monitor, in general, how the orders **[00:21:00]** are doing and if our customers are happy and all that, that is more outside of the application processes directly, but it does get fed back to us.

Dirk: Also, **[00:21:10]** I imagine that for something like Deliveroo, the question of how happy have you been with our service is a little bit of combination out of, yes, the pizza was not that good **[00:21:20]** and the guy was unfriendly at my door, and the app was slower than I anticipated. You have to figure the app part out.

Maria: Yes, it can sometimes be hard looking at user **[00:21:30]** feedback in general. Users might just say this app is bad or this app doesn't work, but what actually doesn't work, what was actually the problem and **[00:21:40]** distilling the data down and seeing the themes. If a lot of users are reporting, they can't use this one button in your application, then there's **[00:21:50]** clearly a problem with that. If one user says your app is bad, you can't actually do much about that apart from asking, what's the problem and how can we **[00:22:00]** fix it?

Dirk: Is there something as a pattern or something analytical that you can spot in the data where you basically have a way to pick things up before they **[00:22:10]** actually become bad?

Maria: I think it depends on the issue. Somethings you're going to be able to spot by checking the data. If it's a performance issue, there's the Firebase **[00:22:20]** performance tracking, for example. Or if you've got your own analytics systems in place to check how many users are actually using this feature **[00:22:30]** and that suddenly drops, there's an issue. Others are going to be a lot harder to figure out where there's just less ideal flow and users are having **[00:22:40]** a hard time navigating that. If that always was the case, you might not actually know there is an issue, unless you go through the flow yourself or you **[00:22:50]** talk to your users and ask about it.

Dirk: I imagine that app quality becomes-- Well, it is an ongoing topic when you develop your app. I imagine in times where you have to roll **[00:23:00]** out a lot of features or where your users change their behavior massively, it becomes even an area of more concern because you have to have a close eye on that. Now, **[00:23:10]** for those listening later to this podcast, we are still in 2020. There's still COVID and I imagine for someone like Deliveroo, COVID brought a lot **[00:23:20]** of additional load, maybe some features, and maybe some changed user behavior. Maybe, Maria, how did COVID affect the **[00:23:30]** development of the app, how users interacted, how you had to pay attention to quality?

Maria: We had to make sure that food delivery is safe for users. One feature we **[00:23:40]** introduced was to have contactless delivery. You would just get your food in front of the door and then you open the door and you magically have food there and you wouldn't have to **[00:23:50]** interact with anybody, just to reduce the risk. There were a lot of features we couldn't really plan for, they just suddenly had to happen and. **[00:24:00]** We had to make sure that we still launched and safely. Luckily, we had everything in place to make rollouts safe and have everything around **[00:24:10]** behind feature flags to being able to disable it, roll out or releases slowly and have all the alerting in place. In the end, it was okay but if we wouldn't have that in place, **[00:24:20]** it would be very risky to roll out these really emergency changes that we had to make really quickly.

Dirk: Would you say the more complexity in **[00:24:30]** the devices that are being used and the user base the more people pay attention to this and the better, as a result, the process is behind that? **[00:24:40]** In other words, if you think back to going through changes and processes and development methods, is there a correlation between how complex the task you're tasked with **[00:24:50]** and how well you usually roll out features and start new things?

Shobhit: I think there is as people's apps get more complex, definitely they have had **[00:25:00]** to step things up from the processes, just how closely they're monitoring app quality, because if you drop the bar **[00:25:10]** on that, very quickly you'll start hitting all these issues that become absolute showstoppers for your users. Yes, there's definitely a correlation between how **[00:25:20]** complex your app has become, how quickly it is grown and just how much attention you need to pay to high quality and how many processes you need to build up to maintain that over time. **[00:25:30]**

Dirk: Would you say tolerance has gone up or down? We used to tolerate quite a lot of crashers when it came to technology, right now, it's probably less and less so. That would be my feeling but **[00:25:40]** I don't know how it is in your experiences.

Shobhit: I have a perspective on this which is that for most app categories, your competition is just a click or app install **[00:25:50]** away. It has gone down. In some occasions where I have a membership with something, I remember **[00:26:00]** having to place an order on my iPad and the app get crashing so I went into the web interface and actually did that. Same thing, maybe if you have a bank, your mobile app **[00:26:10]** might not be working but then you go online and get it to work. If that continues, even in those categories, people switch, they give up. They take their money and go elsewhere **[00:26:20]** literally.

Tamzin: Shobhit, I know that Firebase has remote config testing. Tell us about that because if the listeners today don't know that you can actually do AB tests, really **[00:26:30]** simply without having to update your app he's going to tell you how to do it.

Shobhit: There's actually two things I'm going to mention related to that. First of all, remote config which allows you to **[00:26:40]** maybe, for example, create a feature flag or create different configuration for different devices. What you can do is just simply if something goes wrong, turn **[00:26:50]** off the feature, or roll out multiple versions. Test out which one works best and then decide this is the winner or I'm going to distribute it in some way. **[00:27:00]** That's a great way to put some of these controls in place.

The other thing that often comes across is the works on my machine phenomena where **[00:27:10]** you have, I don't know how many thousands of different types of devices and all these different configurations, then people just can't reproduce the issues. **[00:27:20]** One of the things we actually had to build in Crashlytics was capturing context, things such as what are the exact actions the user took and what **[00:27:30]** was the configuration of the device which, for example, game level they were on when the issue occurred.

Now it becomes easier for folks. One of **[00:27:40]** our customers mentioned that they have this device locker. They pick that same device, they follow the same steps and they are able to reproduce much faster **[00:27:50]** but that was really hard to do without knowing, hey, out of the 3,000 devices that my users are using, which one exactly is it occurring on.

Tamzin: 100% especially back in the day before android when you **[00:28:00]** had all of these different Nokia and was it the Sharp phone, the GX10, oh, my God and all the different versions of java. I don't know what phone they were talking **[00:28:10]** about when the bug got put in but there's no way. I worked for a telecommunications company. We had a lot of phones but this is why I think where we've come to today with it, the test suite of **[00:28:20]** virtual phone testing is incredible.

Dirk: Also, sometimes it's really hard to diagnose by just trying something from what people told you, even if you really get it right, **[00:28:30]** even if you do exactly same thing, it may just still be the case that the surrounding environment is different. There's something that another app they opened just before they changed the status something that **[00:28:40]** clashes with your functionality or I don't know, some sensor goes off that's not going off while you try it. Having some advanced tracking and postmortem analysis, **[00:28:50]** functionality really helps.

Maria: That's where Crashlytics is actually really helpful because you get a lot of data with your crash reports. You get which devices that it happened on **[00:29:00]** which OS version, was in the app in the background, was it in the foreground and did it have memory left. You can then also add your own events to it so you can log the click **[00:29:10]** events, for example, or the screens that a user saw. Then you can look at what the user

actually did. Then in a lot of cases, it will help you to figure out what actually **[00:29:20]** happened or if that only happens on this one obscure device that you actually don't have access to.

Tamzin: Maria and Shobhit, if I had to restrict **[00:29:30]** you to three top things you would advise listeners to do when it comes to quality, what would they be? Shobhit?

Shobhit: For me, these three things, one have a very clear threshold **[00:29:40]** on when you, for example, stop the release and start looking at certain things or when did do you need to start investigating bugs. Once you **[00:29:50]** have that clear decision criteria setup, everything else becomes easier. You don't have to think about it, you don't have to get into long meetings to go and discuss. Second, **[00:30:00]** just stay on top of it.

Sometimes these user reviews and feedback from users are another way of triaging everything and keeping on top of **[00:30:10]** things before you get into that real bad state. Then last but not least, automation. I think manually checking into your release is cumbersome, and **[00:30:20]** not knowing when things went wrong over the weekend until you check the dashboard on Monday is also just a recipe for disaster. Make use of alerts and monitoring and automation **[00:30:30]** as much as possible.

Tamzin: Excellent. Shobhit, I want to touch on something you said there. When I talk to my partners and I see their CEOs and founders, I say, "Did you know that you can set up **[00:30:40]** an email alert for every time you get one star?" They go, "No, I didn't." All of their employees look at me with daggers, "Don't tell him that or her that."

[laughter]

[00:30:50]

Maria, over to you. Top three things?

Maria: Funnily enough, exactly the same three things. [laughs] We're aligned. If I have to **[00:31:00]** think of three other ones don't release on a Friday. Wait until next week. One thing we do is we cut a release on Thursday or a Friday, we **[00:31:10]** let employees use it over the weekend, order some food. Then on Monday, we check-in and release. Just don't go into the weekend not **[00:31:20]** knowing if your release is going well unless you have somebody on call, which you also don't really want to be the poor person having to check the release over the weekend or getting notified **[00:31:30]** or alerted.

Tamzin: I agree with you on that. One thing I've learned as well is maybe take your CEO out of the beta group.

Dirk: [chuckles] That's almost like stealth tactics here. **[00:31:40]** [chuckles]

Shobhit: Easier said than done.

Dirk: Well, maybe half an extra beta group for your CEO then.

[laughter]

Maria: Otherwise, make sure you can disable **[00:31:50]** features, or stop your release. So use stage roll outs to make sure you can stop anything before it happens to too many users. **[00:32:00]**

Shobhit: By the way, I would love that additional job of testing the app by ordering food over the weekend.

[laughter]

Maria: We used to do a test orders for every release. It **[00:32:10]** is not a good thing necessarily because you just eat too many snacks.

Dirk: No one said that you have to deliver it. It's like, you could even have test restaurants and stuff.

Shobhit: We're going to test **[00:32:20]** all the use cases. Let's make sure this works for sushi.

Maria: I did go for days where we had doughnuts, we had **[00:32:30]** cake, we had cupcakes, we had, I didn't know what, it's too much. [chuckles]

Dirk: Well, I'm getting hungry at this point. I don't know what you mean by it's too much. All right. **[00:32:40]** Shobhit, Maria, thank you for being with us today. We had a blast with you. Thanks for sharing your insights. Thanks for sharing your perspective on app quality. I think for our developers, **[00:32:50]** it's maybe the number one thing people think about when they develop software. Firebase is providing a lot of tools to make things easier on that front. Thanks for **[00:33:00]** sharing all of that.

Shobhit: Thank you for having us.

Maria: Thank you.

[music]

Tamzin: Thank you again **[00:33:10]** to Maria and Shobhit. It was great having you on the show today and sharing your insights. Thank you to our listeners for tuning in. If you have any thoughts on the topics covered on today's **[00:33:20]** episode, we'd love to hear from you. You can find us on Twitter at #GooglePlayDev. Keep an eye out and subscribe to our podcast for the next episode coming soon. **[00:33:30]** Until then, keep playing and learning.

[music]

[00:33:40]

[00:33:41] [END OF AUDIO]