EPISODE 1102

[INTRODUCTION]

[00:00:00] JM: GitHub has been a social network for developers for many years. Most social networks are centered around mobile applications, but GitHub sits squarely in a developer's browser-based desktop workflow. And as a result, the design of a mobile app for GitHub is less straightforward. GitHub did acquire a popular mobile client called GitHawk, which was developed by Ryan Nystrom. And since joining GitHub, Ryan has worked on a new mobile app for GitHub along with a team of engineers, including Brian Lovin. Ryan and Brian Lovin. Ryan and Brian both join the show to discuss GitHub mobile and have they designed, architected and built the app.

There is no company quite like GitHub. It's a social network combined with a version control system that provides a critical utility to companies across the world, and all of this made for an interesting episode about a one-of-a-kind mobile product.

We are looking for writers. If you are interested in writing for Software Engineering Daily, send me an email, jeff@softwareengineeringdaily.com. We are paying for some of these opportunities. And I'm also looking for companies to invest in. If you are running an infrastructure company and you're interested in raising capital fourth or something targeted developers, send me an email, jeff@softwareengineeringdaily.com.

[SPONSOR MESSAGE]

[00:01:26] JM: Today's show is sponsored by StrongDM. Managing your remote team as they work from home can be difficult. You might be managing a gazillion SSH keys and database passwords and Kubernetes certs. So meet StrongDM. Manage and audit access to servers, databases and Kubernetes clusters no matter where your employees are. With StrongDM, you can easily extend your identity provider to manage infrastructure access. Automate onboarding, off-boarding and moving people within roles. These are annoying problems. You can grant temporary access that automatically expires to your on-call teams. Admins get full auditability into anything anyone does. When they connect, what queries they run, what commands are

typed? It's full visibility into everything. For SSH and RDP and Kubernetes, that means video replays. For databases, it's a single unified query log across all database management systems. StrongDM is used by companies like Hurst, Peloton, Betterment, Greenhouse and SoFi to manage access. It's more control and less hassle. StrongDM allows you to manage and audit remote access to infrastructure. Start your free 14-day trial today at strongdm.com/sedaily. That's strongdm.com/sedaily to start your free 14-day trial.

[INTERVIEW]

[00:02:53] JM: Ryan and Brian, welcome to the show.

[00:02:55] BL: Thanks for having us.

[00:02:56] RN: Yeah, excited to be here.

[00:02:58] JM: There is now a GutHub app for mobile, an official GitHub GitHub app for mobile. So my first question is pretty obvious. Why didn't this happen sooner?

[00:03:09] RN: You know, it actually kind of did. Back in like 2012, I believe, GitHub released and open sourced an android client. To be honest, I don't know like the whole story, but at the end it just didn't take off as much GitHub.com, like product development cycles, those did take off and it was just harder to come and keep things up-to-date. Not a whole bunch of android users. There wasn't an iOS counterpart yet. They did cut a fork out GitHub like SDKs. There was like a native SDK, octokit maybe. I think an android one, etc. And there've been other attempts at like building a mobile app. The mobile site has gotten like responsive design and whatnot. I think it was just the right place at the right time that this opportunity opened up for us.

[00:03:58] JM: Were there internal debates around actually marshaling the resources around getting a mobile app, I mean, on both Android and iOS. Were their internal debates about it?

[00:04:11] RN: As far as whether to do it or not, or whether to hire, or –

[00:04:15] JM: Yeah. Whether to do it? How to do it? Why to do it?

[00:04:18] RN: I mean, I've only been here a year, but like just scrubbing through some of the old posts and chats and conversations, it definitely seems like there's a lot of internal conversation about it. Whether to do it? When to do it? Who to do it? Et cetera.

[00:04:32] JM: Describe the workflows where a mobile app is actually useful. I think most people think of GitHub as a desktop-centric application.

[00:04:41] RN: My thinking on this has evolved a bunch, and Brian, I'm definitely going to want to hear your take on this too. When we started, I was thinking – So I came to GitHub having previously worked on an open source app called GitHawk, where I was really focused on notification management where I wanted to avoid the scenario where I get to my desktop, I open up gethub.com/notifications and I see like 300 notifications that are going to take me the next two hours of my time to sift through. Whereas there're a lot of times I'm out and about, I'm waiting for a cup of coffee, I'm taking the bus, that I can actually go through a whole bunch of notifications, maybe get rid of some noise and maybe leave like the most meaty important ones for when I'm back at my desk.

I thought that that was like the main use case. We would filter out notifications, maybe open up an issue. You leave a comment that says you're going to look into it later, and then you put your phone away and you're done. What's actually been interesting in launching this app is people want to do a lot more than just respond to stuff. To my surprise, after launching the apps, the app has been live – We're in mid-June. About three months now. The number one action that people are taking in our app is actually reviewing code. And I still struggle to like understand, because sometimes I'm reviewing code from peers and it's a small – I'm using like a Promax iPhone here. And even with this gigantic mobile screen, sometimes it's kind of hard to read. But I think that speaks to the power that this gives people to be able to be walking around, get a ping from a peer that's like, "Hey, do you mind reviewing my code really quick?" Seeing like a 20 line change and being able to leave a review. Not just approve it or ask for changes, but even comment on individual lines of code to evaluate or review somebody's changes. Yeah, I'm still kind of like blown away that that's what people are spending most of their time doing in our app.

[00:06:46] JM: Are there specific workflows that you would like to have on mobile but they're just too difficult to implement on mobile?

[00:06:55] BL: Yeah, there's a lot. I mean, GitHub is over a decade old. So there's just like tons of stuff that already exists that we want to think about how it should fit into the mobile app. Some stuff that we're starting to think about now is around like what happens if you're not super active with notifications or what if you're not like a power open source maintainer? What does the app look like for you? So some areas we're exploring are like the more social side, like how do you discover cool projects? How do you explore what's happening on GitHub and like drill down into the source code? Is there anyways that we could replace that like gut instinct to check Twitter when you're waiting in line at the coffee shop to check GitHub and actually learn something like read a tutorial or dig into some source and see some interesting patterns that people are using in a language that you're interested in.

I would say that's where we're exploring right now as far as like features and workflows that we want to unblock. It's basically the entire collaboration to merging code workflow. So can you comment on things? Can you read the discussion happening around a particular pull request or issue? Can you review the code? Approve it? Request changes? When a pull request is ready to go, can you tap that merge button and see it merge in and know that checks and CI is kicking off in the background to deploy it? If we can get that full end-to-end system working, then everything around that we can start to make more exciting and interesting for people who might not necessarily be working on a project right now.

[00:08:22] JM: I think that Slack was out of left field in how it impacted GitHub workflows. Is it relevant to this conversation? The conversation around developing a GitHub mobile application that Slack has emerged in recent years? Are there any workflows in GitHub that actually are Slack-centric? Therefore, you don't need to think about them as much in developing a GitHub mobile app.

[00:08:49] RN: It's kind of interesting that how much Slack ends up getting used at GitHub to build GitHub. There're a lot of integrations that people are using like chat ops and bots in order to maybe deploy something or check status on something or automate something. We're in the middle of discussing ourselves as a mobile team about building better bots to be able to

automatically deploy our apps, so like app stores, or beta builds just from the Slack. Basically treating Slack as like a command line.

[00:09:21] JM: So as you said 8 ago, there was a GitHub mobile app for android, and the project was eventually no longer maintained. It got released to the public. In between those eight years between the initial GitHub effort at mobile applications and today, what change? What were the most influential changes on the platform?

[00:09:49] RN: You're asking two newbies. Well, like for instance, I think like within between now and 8 years ago, I'd have to check the actual timeline. But I think like pull requests became a thing between like now and eight years ago, which is kind of a crazy thing to think about. One of the engineers on our mobile team, she's been at GitHub for like 4, 4-1/2 years, and early in her tenure at GitHub, she built like pull request reviews and like review requests, which to think of that being only like a single digit-year-old feature is like mind blowing to me, because it's such a staple of what GitHub is today.

I mean, the amount of stuff that's been launched on GitHub in the past 8 years is kind of mind blowing. I mean, some of the primitives we're used to, but also all of the explore stuff that Brian was talking about, being able to surface new things. All the new things that we're announcing like at Satellite and Universe this year with discussions with sponsors, so many community tools. Brian, you look like you're –

[00:10:50] BL: I would add actions to that list. Like actions feels like this one big piece of work that is unblocking you to do anything from GitHub and trust GitHub to be doing it in the background, like processing and automating code reviews and linting, and CI, and CD. Like all of that entire platform is enabling you to sort plug-in to all different steps of the software development lifecycle as we would say. Because that one feels kind of like a critical one for me. Actually, that is kind of the cool thing that has made the mobile app great, is you can merge a pull requests on mobile and just trust that in the background all sorts of stuff is kicking off if you have action setup for CI and CD. So we do that all the time, right? We can merge a pull request and know that work is happening in the background and I don't have to be at my computer to sort of verify that or run another script from a command line or do anything else.

[00:11:46] JM: How did the Microsoft acquisition of GitHub affect the development process for the mobile app? I realized, again, you guys are newer, but do you have any perspective for how Microsoft acquiring GitHub has affected the overall product direction of the company?

[00:12:04] RN: I think I would say that maybe – I mean, Brian mentioned actions. Actions I feel like has as a really great story with Microsoft. Microsoft is kind of pretty awesome CI and CD offering with Azure DevOps app center and more. So I'm sure that behind-the-scenes, there were some conversations about GitHub has a huge opportunity to add more value with actions.

As far as mobile goes, Microsoft's got, to be honest, like kind of a really impressive mobile apps. Outlook is one of the first ones that comes to my mind. I think Outlook is – It's just known as like email and it's been around for like a thousand years. But the mobile team, both engineering and design does like excellent, excellent work.

When we first started, Brian and I actually, when we were both in New York, we met up with the Outlook team and started talking about like how do you do things? How do you work? Let's talk about design systems. Let's talk about engineering principles. So there's maybe a little bit of influence there. But for the most part, what's actually been kind of fun working on mobile at GitHub, but honestly just working at GitHub is it does feel pretty autonomous. We are more or less able to kind of make our own product decisions, design language, our own technology solutions. We were using app center at the beginning and then decided to kind of simplify and move things to actions and just test flight or the Google beta program.

Microsoft, we partnered with them, gave feedback and talked about like what worked, what didn't work, and it was a great partnership. But it wasn't necessarily – I think from the outside, you might think, "Oh, Microsoft's taking over and they're going to tell you what to do and how to do it," and it's actually not like that at all. It's very collaborative with a ton of our own autonomy.

[00:13:54] JM: We've given the high level overview of the impetus for building a mobile client and the context. So let's start talking through the design process. What was the product design process for determining – I mean, GitHub is kind of a one-of-a-kind app. It's not like there's well-defined templates for how to build an app for a Git-driven social network, except for GitHawk,

which you had done before this. But tell me about the product design process. How did you start to lay out how this was going to work?

[00:14:28] BL: I guess you could think of it as like we're going through all of the possible workflows that happen on gitHub.com, the 10+ years of features that have been built, and we're trying to think about what actually makes sense on a phone. When you think about what makes sense on a phone, you're considering where a person might be when they're on the phone, like they might be away from their computer. You might consider what the phone actually unlocks you to do that a traditional browser wouldn't be able to do or at least not easily, like push notifications.

If you combine those things, you sort of land naturally in the space, like, "Okay. There is notification triage. There's like conversations. I want to know when work is happening when I'm away from my computer. Or if someone needs my attention very quickly, can I unblock them while I'm away from my computer?" Like trying to break this connection to have your laptop on you 24/7. So that ended up getting us to where we are now, which is really deep in like issue and pull request workflows, notifications, and then search to like actually just get around.

And those are the three tabs that we have. So home helps you sort of find your stuff on GitHub get into the conversations that are happening, like find the projects that you're working. Notifications is like helping you drill down and unblock people and just know it's going on. So that's really how we started. It's like what does the mobile phone unlock through sensors and APIs, and then what actually makes sense for people to do while they're away from their computer.

I think if we keep following that line of logic, there's still quite a lot we can do. But we have to be pretty selective, because GitHub has so much stuff. I think we run a risk of just rebuilding all of GitHub inside of mobile app and like we've cluttered the interface and it's not actually clear what it's for. Is it a companion? Is it a duplication? How does this work with the desktop app or the CLI? What am I supposed to use to do which thing?

I think we're still been pretty careful about like what workflows do we want to think about, and it's evolving. We're talking about like explore and discovery and how do you help people

discover the right projects to work on or learn a new programming language. Those are things that makes sense and seem interesting for somebody to do as a replacement for – I don't know, scrolling Twitter or something like that.

I guess it's not concrete we're evolving it, but certainly how it started was looking at complementary behaviors to what is possible on a browser.

[00:16:43] RN: I want to add to what Brian was saying too. I think something, a principle that we've had pretty early on it, acknowledging how complex of a product GitHub is, is that like we actually have to have a really strong opinion of what the mobile app should be. And that's actually been really fun for us, is to take all of the complications, all of the different products and nuances and workflows in GitHub and to just drill it down to like what should it be in the phone? And to just design, ideally, like one way to do something. A good example would be like how you leave a review on a pull request.

Behind-the-scenes, there're a bunch of different ways you can actually do that, and different teams have different workflows. And while I would love to be able to build and support every single workflow, some of them are kind of complicated and unintuitive on a small screen. And so we just set off to say, "You know what? We're just going to do this one way the way we think it's right, and we're going to put it out there." And we have to fully acknowledge that some of the time we're going to get it wrong, and we have got it wrong, and we will get it wrong again, but I think taking like a firm stance on like what we should build in, and more importantly what we shouldn't build has led us to success thus far.

[00:17:57] JM: How are you leveraging your experience creating GitHawk when you were building the GitHub mobile app?

[00:18:05] RN: I'm leveraging it pretty hard. I mean, a lot of the kind of fundamentals and discovery of building that app apply to the new app. And we've made some foundational architecture and technology choices like GraphQL. We've leaned into GraphQL really, really hard. When I was working on it pretty much by myself, I was, from the design side, really didn't want to stray away from like the system design language. I wanted it as iOS app. I want it to feel like an iOS app. I wanted to be basically like the settings app, but spruced up a little bit so that

the most novice GitHub or iOS user could pick up their phone, open the app and immediately know what to do.

We brought that philosophy to not only our new iOS app, but also our android app. And so all the designs – Almost all the designs we're using in the app feel like a native app. I mean, it is a native app, but like we want it to really feel and look the part. And there have been a bunch of other core technology choices that I think we've made. Most importantly, it's just been like our decision-making principles. We try to keep things simple. We try not to over-engineer. It's more important to like iterate, ship something really quickly, learn from our users, fix things going forward and not be afraid to like take risks.

When GitHawk was like my own little hobby project, there were a lot of stuff that I just like hacked together and shipped and found that the people hated, and then we just deleted it and we moved on. And even though we're a team of more than a dozen engineers now, I want to like continue doing that. I want to continue building stuff, trying things, taking some risks, learning from our users as we go.

[00:19:44] JM: As far as the technology choices for building mobile apps, it seems like you could have used React Native. I mean, how much complexity is there really in a mobile application and why not just simplify it and make it cross-platform? But you did not use React Native. You guys built native mobile applications. Why is that?

[00:20:10] RN: So I've been building native iOS apps since like 2010 or 2011. Before I came to GiitHub, I spent about five years working at Instagram, mostly as an engineer while I was there. And React Native was built by Facebook. I saw a lot of awesome uses of React Native. I saw it evolve. I saw it grow. I also saw some of the costs to like an engineering organization. And I think if you go all-in and you really know what you're doing, you can make a really great React Native experience. I quite frankly have never written a single line of React in my entire life. So if I as like the sole founding engineer and lead of this team decided to jump headfirst into a tech that I have no idea how to use appropriately, I would've probably set us up for failure.

I have to acknowledge the cost, I guess, of having native iOS engineers and need to be android engineers, but I knew that we were able to achieve our mission of shipping a high, high quality

iOS and android app faster using the tech I know and by going out and finding other people that know their native text accurately well.

[00:21:22] BL: I am not super knowledgeable hero on the tech stack, and I know React Native has gotten quite a bit better. I first played with it in 2018. It's moving quickly. It's getting better. But like I think it's worth calling out that it's WWDC week when we're recording this, and there's something very liberating about being able to watch a platform stream like this or like I/O and seeing new platform APIs and being able to very quickly into it what you're going to get for free. What's going to be possible and not having to think like one layer of abstraction around at like, "Okay. How is React Native going to handle that specific thing?" There're costs, like we have two platforms, two separate teams of engineers. But there's also this benefit that we get so many platform features out of the gate, and certainly earlier than React Native might be able to support them.

Is that right, Ryan? Am I articulating that correct? You're the expert here.

[00:22:11] RN: Yeah, you're spot on.

[00:22:13] JM: There are a variety of different development platforms now for clients that are on different surfaces. So you've got the desktop website. You've got mobile applications. There's GitHub desktop, and I would imagine there are – That can make the endpoint management somewhat complex, because you have to have perhaps mobile endpoints. You have to set up specific endpoints, or maybe you can just use GraphQL for everything. Tell me about how the endpoint management works when you have these different surfaces.

[00:22:46] RN: I mean, yeah, you kind of answered the question. We use GraphQL for everything. Almost 100% of the iOS and Android app use GraphQL, except for a select number of endpoints. End of the day, we're an OAuth application, just like any other app that you can make for GitHub. We use the public API. We have some kind of like feature preview things that we're using before we actually publish the APIs, and we've done that in the past and we'll continue doing it, of creating new APIs for the mobile app and then polishing them up and shipping them and making those APIs available for public use.

I also manage the desktop and command line teams here at GitHub, and those seams are also using like a mix of Rest APIs and GraphQL. We have people that work on the mobile team that actually 100% focus on supporting or adding API support to our app. And that's been huge. I think GraphicQL has more or less client engineers be their own API engineers and create and define queries to fetch the data that they need instead of having to lobby or put in a ticket with a server engineer who goes and builds an API and then publishes it and having to do that coordination. But there are still sometimes stuff that we need in the GraphQL API that either doesn't exist or doesn't totally work as we need it to. Ultimately focusing on what the user experience is. So having somebody on our team dedicated to adding that support, also most importantly looking into security authentication issues and those sort of things has been invaluable.

[00:24:25] JM: What are the things that you need out of perhaps the legacy backend that are not supported by GraphQL, or have you had to rewrite the GraphQL endpoint at all to include certain legacy infrastructure?

[00:24:39] RN: Probably the most challenging – I'm trying to think. Probably the most challenging API change that we've made thus far is with files and dealing with like Git objects. So at its core, those are – We have abstractions. So github.com is built in Ruby on Rails. We have abstractions over top of actually get core object types. And when we want to display a file and we want to syntax highlight some code, we first have to have a branch. We have to have a commit, and then we have to have a file path. And once we reach that path, we're going to have a Git object that we maybe want to interact with, load the contents of, view the history of.

In order to do those things, we end up having to add API support to basically git. That's been pretty challenging. I mean, even to like render a markdown file as like styled markdown gets pretty complicated.

[SPONSOR MESSAGE]

[00:25:45] JM: Errors, and bugs, and crashes happen all the time across my software. Most often, these crashes have to do with obscure exceptions that come from React components, failing to render on the client device. Source maps and stack traces would be useful, but in

many cases, I'm not able to identify the root cause because the error is occurring on a client device. It's not on my infrastructure. And what can I do about that? I can use Sentry. Sentry.io can quickly triage and resolve issues in real-time.

Visit sentry.io/signup and use code SEDAILY to sign up for two free months. You can simply choose the platform that you're integrating with. Sentry works for every major language and framework, from Rails, to C#, to Java, or React Native, and you just install the SDKs. You integrate it with your application. After that, errors will be caught by Sentry and you'll be alerted the moment that they occur so that you can triage, and resolve, and keep your users happy with functional applications. If you're building an application and you want to monitor your errors and your performance, try Sentry by visiting sentry.io/signup. And new users can use code SEDAILY for two free months.

Thank you for being a sponsor of Software Engineering Daily, Sentry, and you can go to sentry.io/signup and use code SEDAILY if you're curious about it.

[INTERVIEW CONTINUED]

[00:27:18] JM: So one specific element of designing the application, and this is a design question for basically any mobile application is that of notifications. So github.com, I often feel like I'm drowning in notifications. I imagine figuring out how to design around notifications for the mobile application has its own set of thorny issues. Tell me about how to get notifications right in the design.

[00:27:45] BL: It's very hard. Probably don't have it perfect, like there's just too many different ways that people use notifications. So we're trying to build a set of behaviors and like interactions that will let people at least find the things that they care about most quickly and get rid of the things that they don't care about very quickly. I mean, one obvious example is like you can just swipe things away. You can just kind of just swipe, swipe, swipe, swipe and kind of clear out your inbox. We've also worked with the github.com team to build out filters. You can jump to a specific set of notifications by repository or by the reason that a notification was triggered. Was it triggered because you are mentioned as a team reviewer? Because you are mentioned by name or because you've, at some point, clicked subscribe on a particular issue

and there's been new activity in there? It's like how can we get you into those filtered views as quickly as possible?

We're working right now on some more tools to help you clear out and inbox faster, like being able to bulk triage. Just being able to select a whole bunch of notifications at once, mark them as done and move on. And then I would add, we've been very, very careful about push notifications, because we just know there're too many people that get too much stuff. If we enable push notifications for everything out of the gate, nobody would use the app. It'd be a gigantic nightmare.

So our philosophy has been you have to opt-in to push notifications and then you're going to have to opt-in to each type of push notification that is important to you. Luckily for us right now, the only type we support is direct mentions. So it's pretty easy if you turn on notifications, you get direct mentions. But eventually you will be able to support more push types, like what happens when someone assigns you an issue or request your view on a pull request? You'll be able to opt-in to those specific sets of push notifications and really control the content that you care about.

And then the last thing that is on our mind, which we are working on, hopefully, hopefully come soon. We're trying to think about – There are lots of people who use GitHub as part of their day job. This is like a 9-to-5. They login to GitHub because they're doing work that they're being paid to do. And outside of those hours, they don't want to check GitHub. They don't want to be pinged for this kind of stuff.

So we've been thinking high-level, like what is it look like to just tell GitHub, "Hey, I work during these hours. It's safe to tell me about new development activity happening on GitHub within these hours. But outside of that, I don't really care." So we're calling this working hours. You'll build a define that an app, it will be per day. So you can say Monday through Friday, I work 9 to 5. Saturday and Sunday, I don't want to hear anything at all. And we'll silence those push notifications in the background given that schedule.

So it's not quite do not disturb. It's a little bit of the inverse. It's like these are times that we're, "Yes, you can send me things." And every time else, we'll silence it. But it doesn't impact the rest of your applications on your phone with do not disturb.

[00:30:37] JM: Nice. I mean, it is useful, because that's – GitHub is kind of one of these things that's sort of like Slack where it's like pro-sumer. It's a productivity tool. It's also consumer tool. I can imagine wanting to see push notifications from something that you're casual follower of, but not wanting to see push notifications from something work-related. So I can imagine needing that really fine-grained notification management there to create the right experience depending on how you happen to use GitHub. Are there any other design decisions that come to mind when you think about this as like is this a productivity tool? Is it for work? Is it a consumer application dealing with those blurry lines in what GitHub is?

[00:31:21] BL: I think this is an interesting question. This is something that Ryan and I are grappling with daily, which is GitHub has a lot of developers that use the platform within developers, we have a wide range of types of developers. You have your open source maintainers. We have new developers, people who are learning for the first time. We have super experienced, hardcore people, but maybe they just work for an enterprise organization. But then we also recognize that there're lots of other people involved in the software development process. We have designers. We have PMs. We have data scientists and researchers. We have executives and leadership teams and managers, like all of these different people expect different kinds of things out of GitHub. And that explains a lot of features you see like projects, right? How do I organize all of this work that's happening within a repository? That's what Ryan and I are thinking about right now, is within developers, there's this huge variety of people and ways they might work. And then outside of developers, we can see those sort of, I guess, you would say personas. That type of person, a PM, or a designer becoming a lot more common on GitHub and becoming more comfortable being involved in the software development process.

So we got to think about what tools those people need. What information they care about? How do we help them get to it most quickly? Whether it's notifications, issues, projects, like other kinds of things like that. How do we help them get to where they can be actionable as quickly as

possible?

[00:32:44] RN: And something too that I think is really important to us is that we launched the first version of this app mostly as like a productivity tool. When you got done triaging notifications, when you got done reviewing pull requests, there was nothing else really to do in the app. And that was by design. We didn't really want – That was the purpose of the app. We didn't really want you to do too much else. We're now starting to experiment a little bit with like explore or what could surfacing trending repositories or people or whatever in the app look like, but it is kind of important to us that we don't ever want this to become an endless feed, an endlessly scrolling feed of like stuff. It's not our mission to keep people's eyeballs attached to their phone consuming GitHub content. It's more important for us to enable people to discover more repositories in communities that they can collaborate to and then start collaborating or to give them more important actionable things that they're able to do, or even let them define maybe their owner app experience by customizing things, adding shortcuts or actions or whatever and not get into the realm of like endlessly scrolling lists, because that's not really what GitHub is at its core, and I think that's something that we're really excited about helping people get their work done faster from anywhere and less about time spent in the app or something.

[00:34:06] JM: There are many CICD workflows that center around GitHub. Did CICD fit into the product design of GitHub mobile?

[00:34:14] RN: It has not so much yet. If I'm being honest, I think there's a lot of opportunity and room for improvement there. I've already found myself – Even as we speak, one of our builds is breaking on iOS, and all I want us to be able to flip open my iPad and jump into the logs, maybe cancel it, rerun it, see if it's a flaky test, download artifacts and sift through that. But I really can't yet I kind of can on on .com from my phone or from my iPad, but not as easily.

I would love to eventually see a world where you could manage your runs from your phone. I think it's actually very possible, and I actually think the experience would be pretty good, whereas before I thought like an endless stream of logs might be kind of insane. I think that we could actually build like a pretty fun experience.

[00:35:05] BL: Yeah, I think we got to be careful, because I could imagine a world where like you could get a push notification about a failed build on a pull request that you just opened. And the cool thing about that is you could open a PR on your computer, walk away, go do something and get a ping on your phone if it's merging or not, or failing.

I just wanted to call like one principle that we've had, and I think we're doing pretty good here. I'd say we're pretty close to 100% coverage, is we can't support every possible workflow. So on every view, we give people share buttons that they can sort of eject out into their default browser on android or iOS. Ejecting out into Safari where, yes, the mobile websites shows you logs and you can have buttons to rerun action, stuff that we haven't built natively as long as people have the ability to have that escape patch and finish a workflow from the browser, then we're comfortable sort of buying yourself some time to really think through what is the subset of that that should exist natively.

[00:35:59] JM: With any mobile application, there is of course the issue of offline mode. So desktop application, you don't to really worry about that, because you go to the website, get 404 as if you're offline. Do you have an offline mode for the mobile application or do you just kind of ignore that case?

[00:36:20] RN: It's great timing, because we've been – Internal debate about what the future of offline for our app should be. And we haven't entirely decided whether we want to take things, but I think the harder part is that GitHub really is like a social network, and that social network has an object graph that is being mutated probably thousands, tens, hundreds of thousand times, hundreds of thousands of times per second.

Having the latest – If we are caching stuff offline, having the latest up-to-date data at all times is probably not going to be possible. Also, because in our app, we allow you to kind of endlessly link into users, and repos, and issues, and pull requests, and the rabbit hole is infinitely deep. So keeping that data updated and in sync as you're browsing the app is going to be like impossible if we're supporting fully offline.

So I think there's always going to be an online component that's going to be required. Otherwise we don't also want to mislead the user by thinking, "Oh, I'm offline. I'm opening my repository,

but maybe that hasn't been fetched or updated in a couple days. You're now on the airplane and you're thinking to yourself, "Well, wait a minute. Why are my star counts out-of-date?" or "I swear there was a pull request about this one feature and now it's not showing up." And there's ways from the design side that we can handle it, but I'm not sure that's going to be super valuable to users. Where I think this could get really interesting is if we eventually started doing stuff with file systems, with actually fetching the contents of git repositories or maybe allowing you to like fetch repo contents and view them offline. But I think in offline by default, I'm not necessarily sold that it will support that.

[00:38:09] BL: Yeah. I'm more interested in solving like the problems that arise on the edges of offline, which is like, "Oh, you're driving and you go through a tunnel and you lose connection for a few seconds." Like how do we make sure that any work that's happening on your device gets in queued or persisted in some ways? You're not losing data if network disconnects or you have spotty network. We have users all around the world on various network strengths and qualities and different types of phones. Like we should be very considerate about their experience on unreliable networks or with networks where large data payloads actually cost them money, like people who are on prepaid phones and things like that. I want to design really considerate features around the edges of that offline. But then yeah, to what Ryan said, like how do we go fully offline is a harder problem.

[00:38:57] RN: Yeah. I would love to be able to be on the airplane, open my phone and file maybe like three or four issues that I think of features that maybe we should add to the app or bugs that maybe I found and then put my phone away. And then when the plane lands and I take my phone off of airplane mode, that behind-the-scenes, those automatically get posted and filed. I think something like that would be great and something we could definitely do.

[00:39:22] JM: The development of GitHub mobile started more than a year ago, and I'm sure the team has developed over time. What has been the process of scaling up the development workforce in allocating work and synchronizing work between those different teams given that it's cross-platform?

[00:39:41] RN: It's been pretty wild. It started – So I joined GitHub May 28th. At that time, Brian and I were both living in New York.

[00:39:51] BL: 2019.

[00:39:52] RN: Yeah. Yeah. Yeah. Right.

[00:39:53] BL: Not a month ago.

[00:39:54] RN: Not this year. Thank you. It was just the two of us. I was doing development. Brian was doing design. And since then, we staffed up. I think we have 13 engineers now, seven on iOS, five on android and then one person working on the backend, as well as two managers, two designers, two product managers. I think that's the full team. To go from literally the two of us to that large of – I mean, an actual like organization in like 12 months, it's been pretty wild. Just like code, I think that it's important to never get to attach to your process or how you do things. Process I think should always be reevaluated. It should be experimented. It should be iterated on. It should be changed. It should be reverted. I'm classically terrible at project management. I can't stand it. I'm terrible at it.

What's been great is that I've been able to acknowledge that with the team and let the team kind of bring their own ideas. We frequently hold retrospectives on our own process how a particular app update went or a feature development went. And we kind of extract to things that worked really well and the things that didn't work really well.

Most recently, I've been experimenting with actually using GitHub actions to improve like our actual development process by automatically closing things, keeping issues in sync, kind of like manipulating stuff, leaving comments, adding labels, etc. And that's been pretty useful, but it's an ongoing process and probably a year from now it's going to look entirely different.

[00:41:30] JM: Had there been any deeply difficult engineering challenges that we haven't covered, or is it more of like a set of product design challenges? It does seem like more of a design-heavy set of problems or the engineering heavy set of problems.

[00:41:43] RN: Probably the most complicated engineering problem we have in our app is actual markdown rendering and support. It's gone through a couple of different iterations and

designs and re-factors, and there's still room for improvement, I think. But GitHub – One, GitHub supports markdown, which is complicated. Behind the scenes, it's taking markdown and converting it into HTML that we then display on issues on .com. But in addition to that, GitHub has what they call a GitHub flavored markdown, which allows you do a bunch of other stuff. There's a subset of HTML elements that we support that also have a subset of HTML element attributes that we support.

At first, I thought like, “Hey, we can just rewrite this spec and do all of the handling ourselves. That turned out to be super complicated.” So we experimented with rendering comments and web views and all sorts of things. That is also kind of like always evolving and being improved upon.

[00:42:39] JM: Do you think in the limit, people want to write code on their phone? How much of the development workflow. Do you think will move to the phone?

[00:42:49] RN: When we started, I thought never. Absolutely never anybody would ever want to write code. But now I want to. Now –

[00:42:58] BL: Well, that’s because Ryan lives on his iPad. I would almost flip this. It’s like less about the phone and just like more about not a desktop. Because there's just lots of devices that are outside of a laptop or a desktop that kind of makes sense. The iPad certainly feels like a more clear candidate here just because of the bigger screen and like we’re getting better external keyboards with external track pads. That kind of stuff makes this feel – It's bridging the gap a little bit more smoothly. But yeah, the phone, it seems like there's just such a small subset of things you'd actually want to do. Like, yeah, you could tweak a markdown file or like fix some typos. But writing fully fledged code, committing files, like I don’t know. Maybe we'll get there. But on the phone, that seems like more of a stretch.

[00:43:44] JM: Has building the GitHub mobile app given you any other newfound perspectives on how software engineering might change in the future?

[00:43:50] RN: It's really shown me how much I actually don't need a computer to do software development, and not software development as in writing code. Because as it stands, at least

for the nature of our work right now, building mobile apps, like I need a computer to write code. But a lot of our job as a team is not necessarily writing code. It's communicating with each other on Slack. It's writing issues. It's responding to people. It's reviewing code. It's brainstorming. It's hopping on Zoom to collaborate and discuss the new idea.

And what's been really interesting especially in this like COVID world that we're in, I found a lot of value in just like getting outside and trying to like work in different environments, being kind of cooped up in an apartment or 3+ months now. It's been really great to be able to take my phone, take my iPad, go out to the lawn away from everybody, away from my apartment, and you just keep working. Keep taking meetings, review code for like an hour. And that's actually changed my perspective on like what it means to be a software developer. And it's also a little bit liberating to not necessarily be in the office. Not be at the corporate office or my home office and still be able to build things with other people. I think it's really, really exciting.

[00:45:11] JM: What are the upcoming features for GitHub on mobile that you're focused on right now?

[00:45:17] RN: Some of the big ones – I mean, Brian talked a little bit about the notification support. I think that's pretty important to us right now. One of the big ones that we are committed to reaching at least a V1 of support by the end of the year would be GitHub enterprise support. GitHub has a ton of large customers that use on-premises installs of GitHub enterprise to build apps to build their tools to just run their software development teams. Some of these companies have tens of thousands of employees, and we really want to find a way so that those people can also just as much as the open source open source or get hub enterprise cloud users have been able to use our app to build and collaborate on software. We want to do the same for on-premises customers.

[00:46:06] JM: Well, is there anything else that you guys would like to add about where you see GitHub going or what changes you anticipate around the social network for developers?

[00:46:20] BL: I think we're seeing just more and more this liberation from the laptop and the desktop. And I think if we project out, you can just imagine the workflows we've talked about becoming better and smoother and faster around triaging and collaborating, reviewing,

discovering. I think we can just go so deep there. I'm excited also about like the massive amount of people who have yet to discover GitHub and who aren't developers yet and what role can GitHub help in like finding them projects to learn from and contribute to overall just build more software better, faster, more securely. It's like this really virtuous cycle that I think we can speed up and make it feel a lot more fluid away from our computer, because not everybody has $5,000 MacBooks. If we can put that on the phone, we unlock a lot of software development capabilities for a lot more people. It's a little high-level and vague and maybe a little fluffy answer, but that really is how we're approaching this. We can see a lot more activity happening on mobile apps over time as people transition to just not needing that keyboard all the time.

[00:47:30] JM: All right guys. Well, thank you so much for coming on the show. It's been real pleasure talking to you, and congrats on expanding GitHub to another set of services.

[00:47:38] BL: Can I ask you a question before we go?

[00:47:40] JM: Please.

[00:47:41] BL: What do you think of the mobile apps?

[00:47:43] JM: I think they're functional and I think they do what they should –

[00:47:46] BL: Dutiful.

[00:47:47] JM: That's a compliment.

[00:47:48] BL: I love it. No. That's what we want to hear.

[00:47:50] JM: I mean, I think you don't want to go overboard, and you shouldn't on the early iterations of it. Because you just want to be parsimonious with that design space.

[00:48:01] BL: What do you think we should build next?

[00:48:04] JM: Where I would try to think about going would be kind of like how Facebook broke itself into multiple apps. I think you eventually need to break up GitHub into multiple apps. I don't think you can get it all in one app.

[00:48:16] BL: What's an example? What's a split-off?

[00:48:19] JM: CICD.

[00:48:19] BL: CICD.

[00:48:21] JM: Right? I mean, am I wrong?

[00:48:23] BL: Maybe. We're thinking about – There's just a ton of stuff. How is it all going to fit in? how is it going to be organized and not – I don't want to become the overflow tab on the Facebook app, which is like a literally a list of 100 sub apps that all exist within Facebook.

[00:48:39] JM: Where is that? I want to find that right now.

[00:48:41] BL: Open Facebook and click on the hamburger button in the tab bar and just scroll away my friend. And that's hard. Feature discoverability and prominence and organization is hard. Multiple apps is interesting. I think, certainly, in the structure we can find better ways to organize and surface the right things.

[00:48:59] JM: I was wondering where messenger kids was hiding.

[00:49:02] BL: Now you go it. There you go.

[00:49:05] JM: Does that sound realistic? Have you guys talked at all about having multiple apps?

[00:49:09] BL: I think we've talked about more about how do we help people get to the things that they actually care about in the design workshops. So I don't know if we could actually say anything concrete there, but I think that's the more interesting problem, is like of all the things,

we can probably get better at figuring out what a given person cares about within their role of building software.

[00:49:29] JM: Cool. Well, thanks guys. Good talking to you.

[00:49:32] RN: Thanks a lot for having us.

[00:49:33] BL: Thanks.

[END OF INTERVIEW]

[00:49:42] JM: Today's episode of Software Engineering Daily is sponsored by Datadog, a monitoring platform for cloud scale infrastructure and applications. Datadog provides dashboarding, alerting, application performance monitoring and log management in one tightly integrated platform so you can get end-to-end visibility quickly, and it integrates seamlessly with AWS so you can start monitoring EC2, RDS, ECS and all of your other AWS services in minutes.

Visualize key metrics, set alerts to identify anomalies, and collaborate with your team to troubleshoot and fix issues fast. Try it yourself by starting a free 14-day trial today. Listeners of this podcast will also receive a free Datadog T-shirt. Go to softwareengineeringdaily.com/datadog to get that T-shirt. That's softwareengineeringdaily.com/datadog

[END]