EPISODE 13

[INTRODUCTION]

[00:00:00] JM: Facebook is a large multiuser application. Scaling Facebook was different than scaling a single user application such as an e-commerce store or a search engine. A social network is faced with unique infrastructure scalability challenges as well as subjective questions around user communications, privacy and content.

Pedram Keyani worked at Google before joining Facebook in 2007. In his years at Facebook, Pedram worked on infrastructure, internal tools and management. He became deeply familiar with the company culture and its operations. Pedram joins the show to talk about how Facebook has scaled and the lessons that he took away from his time there. After his time at Facebook, Pedram joined Uber where he worked as a director of engineering for four years.

Uber is another multiuser application with a very different set of constraints. At Uber, Pedram worked on several projects including Uber's push into China, which he describes an intense, competitive experience. Pedram is able to contrast the culture and scaling processes of Uber, Facebook and Google, and it made this a rare opportunity to see how three different high-performing companies build software differently.

I hope you enjoy the episode.

[SPONSOR MESSAGE]

[00:01:30] JM: When I'm building a new product, G2i is the company that I call on to help me find a developer who can build the first version of my product. G2i is a hiring platform run by engineers that matches you with React, React Native, GraphQL and mobile engineers who you can trust. Whether you are a new company building your first product, like me, or an established company that wants additional engineering help, G2i has the talent that you need to accomplish your goals.

Go to softwareengineeringdaily.com/g2i to learn more about what G2i has to offer. We've also done several shows with the people who run G2i, Gabe Greenberg, and the rest of his team. These are engineers who know about the React ecosystem, about the mobile ecosystem, about GraphQL, React Native. They know their stuff and they run a great organization.

In my personal experience, G2i has linked me up with experienced engineers that can fit my budget, and the G2i staff are friendly and easy to work with. They know how product development works. They can help you find the perfect engineer for your stack, and you can go to softwareengineeringdaily.com/g2i to learn more about G2i.

Thank you to G2i for being a great supporter of Software Engineering Daily both as listeners and also as people who have contributed code that have helped me out in my projects. So if you want to get some additional help for your engineering projects, go to softwareengineeringdaily.com/g2i.

[INTERVIEW]

[00:03:21] JM: Pedram Keyani, welcome to Software Engineering Daily.

[00:03:24] PK: Thanks for having me.

[00:03:26] JM: Today we're going to talk about your time at Facebook as well as your time before Facebook at Google and your time after Facebook at Uber, but let's start with Google. Before you joined Facebook you did work at Google, and when you were there, you worked on Orkut, which was an early social network developed at Google. What were your early lessons about social networking that you learned at Orkut?

[00:03:49] PK: For me, Google is the first job I had outside of college, and it was cool because this was the first time I was writing code that people were actually using, and Orkut was used heavily within Brazil and India and some other countries as well. Seeing simple features like groups that connected people who were geographically distributed around the world actually discuss and debate and had meaningful interactions with each was really eye-opening. I mean, it became really addictive to me that I could actually build things that people were using. So that's on the social side.

On the technical side, the thing that I learned was that when I joined Orkut, they were actually built on top of MySQL server, which is Microsoft Technologies, and a bunch of Microsoft technologies. SQL server is kind of like the kitchen sink of databases, and we were having a hard time getting it to scale. There're only so many tweaks and things that we can do to actually make it work more efficiently for us before we decided that we have to actually rewrite the whole backend in C++ on top of Bigtable which is Google Technology.

So, for me, learning kind of early on that the technology decisions you make really tie you down, and if you use something that you can't get inside of that's close source, that's expensive, then it's really going to limit your ability to scale. That's like the big technical take away from me on Orkut.

[00:05:12] JM: Were there any broader scalability lessons about infrastructure from around that time, 2005 to 2007, that you learned when you were at Google?

[00:05:25] PK: I think the big scalability issues, Google was built for scale. Actually, we had to reverse engineer how was our product going to work on top of things like Bigtable, which allow you to scan in one direction, but not the others.

For example, your friends list on Orkut, the way that we made it work so you can do pagination forward and backwards is that we had two instances of Bigtable. One that paginated in one order and one that paginated in the other order and we're able to jump back and forth. I mean, we really had to – At least for me, I realized that you can't just have this theoretical problem that you want to solve and you can just write some local code. You actually have to really think about how all the backend systems will work together.

I think the more interesting and challenging scalability issue we faced at Google, at least on Orkut, was that as we went from this obscure project within Google to something that was starting to get real attention with billions of page views, there were these rumors in hushed voices talking about the idea that we would get some kind of Founders Award, which is a grant that Larry and Sergey gave.

So we went from seven engineers to I think 15, 20 engineers and a distributed site in Brazil and a bunch of PMs and business folks and legal who wanted to be a part of that team, and we really quickly went from a very fast, little team to bog down team with a lot of people who had to make decisions. So, technically, the scalability issues were not what we dealt with. It was more on how teams emerge and the motivation for people to be on a team.

[00:06:58] JM: It's funny, I had a show pretty recently with Matt Klein from Lyft who built the Envoy proxy, and he's been engineer at Lyft for a while. Before that, he was at Twitter, and before that he was at AWS. And he's written a lot about this idea of human scalability problems that in a fast-moving company, often times the most acute scalability challenges are not technical in nature. They're human in nature. Can you describe a little bit more what kinds of human scalability challenges you've seen throughout your career?

[00:07:33] PK: Yeah. I think the biggest scalability issue is when you're part of a very small mission- focused team, everyone is focused on surviving. Everyone's focused on, "How do I make my product work? How do I make this thing successful?" You're not concerned about how big of a slice of the pie you have. You just want that thing to bake properly.

As teams get bigger, as responsibility gets more diffused, as people start to perceive that there is a lack or a scarcity of rewards, they sort of behave differently. I've seen – And we'll get this letter. I've seen at Facebook, this is handled really well, because there's a constant reminder from the leadership all the way through the leadership chain that once you're going to build it, it's going to be more impactful than like the next performance review for yourself. Tying it back to how do people think about their own evaluation and how they think about their career and what are they going to get out of this experience.

I mean, I think that plenty of great ideas have died along the way, because people lost sight of the fact that the mission is more important than the individual team members, and if you get the mission right, everyone will be rewarded in terms of credit, in terms of promotions, in terms of the knowledge that they gain.

[00:08:55] JM: You joined Facebook in 2007. What was your initial job at Facebook?

[00:08:59] PK: Yeah. So I got hired as a general-purpose engineer. I had experienced back at Google working on some anti-spam and anti-porn tools. So that I think there was a hope that I would gravitate towards that work, but there is no mandate that, "Oh, you have to do this." But I quickly dug into that area and started building all kinds of things and it kind of suited my nature because I'm naturally paranoid and I worry about the edge cases a lot.

[00:09:23] JM: Can you say more about that, about your nature? What causes you to be to be paranoid? There's that classic book by Andy Grove, *Only the Paranoid Survive*. What are the virtues and detriments of paranoia in the tech industry?

[00:09:36] PK: I think, for me, it was just thinking about how would people abuse a system as it went from this kind of niche college network to more wide-scale. When I joined, and I was going through the codebase, I saw some things that people has written around rate limiters and I was like, "Oh! I wonder, in what case would you want to actually bypass certain kind of human limits on posting on other people's walls and stuff like that?"

I actually start to dig in and find out when these things were being triggered. Oftentimes it was someone was trying to promote their band or promote some product that they had in a way that was like really annoying to other people. So I just now started to think all the time like, "Okay. Well, how would someone want to abuse the system, abuse their friend graph." Because early I was just a friend graph, and I kept thinking that if this thing becomes successful, it's not just going to be your close friends and family. It's going to be people that you're a part of community with or random people that you want to be friends with.

[00:10:38] JM: What were Facebook's early problems with site integrity?

[00:10:42] PK: So the site integrity team was formed by this guy, Jeff Rothschild, who saw that there was kind of a loose affiliation of people working on these problems. Rudimentary, rate limiting was our initial way to solve a lot of these problems. But some of things that we had to like really sort of account for, again, is it went from a very exclusive college network to a more widespread network, is how do you build frameworks that allow product builders to build their product without actually having to think about this? Because every product builder has to think about every single abuse scenario, it’s just going to be oppressively hard for people to innovate and build things.

So Facebook as a service was initially very nice and clean. There was just a profile, and then we added things like newsfeeds and groups and events. As the surface area grew, we needed some kind of framework that allowed product builders to build their product with just adding a few lines of code and then not having to worry about anything else. That's really what me and my team focused on.

[00:11:48] JM: There are some canonical problems, integrity, type of solutions, like building a spam detector bears some resemblance to building site integrity tools for a social network. But I'm also sure there are ways in which the social network qualities of Facebook were completely new and there were completely new attack vectors the you had to solve for. What was new about building tools for detecting problematic behavior across a social network?

[00:12:21] PK: Yeah. There’re a lot of things there where, again, in email spam, for example, you know the contents, you know the structure of the email. At Facebook, we had messages, we had wall posts, we had events, we had comments in photos, we had photo album titles and descriptions. There was just hundreds, and then eventually thousands of different places that you could have text in the site. Then you can actually add URLs in any of that text. So we add protection in all kinds of different places, again, in a way that was invisible to our developers and also invisible to our users.

[00:12:57] JM: So, what kinds of APIs do you want to offer to your developers to give them the tools to build their own domain-specific site integrity features.

[00:13:09] PK: You mean internal developers?

[00:13:12] JM: If I understand you correctly, you wanted to build tools that made it easier for Facebook's internal developers to add integrity to their own internal Facebook applications. What kinds of tools did you want to build for those internal developers?

[00:13:30] PK: I can explain this through the first real thing that I built, which was a system called Blackhole, and the purpose of Blackhole was to be able to enter text into it. So I added a simple API where you can basically submit text and it will pull out domains, URLs, email addresses, phone numbers, IP addresses, normalize them and then quickly check them against a list to see if there’s malicious content in there or something that might be fishy or something like that.

So I built that and I started to go through our product code and find the places where I needed to plug this in, and I saw kind of a handful of different anti-spam things that people put in. Again, we had a rate limiting system. Some people put things where they built like really rudimentary profanity checks and things like that. So our code was littered with integrity related code all over the place. I clean that up.

I built a very simple framework called Sentry that allowed you to just basically enter the contents of something that was going be submitted and then configure it through a simple site var, internal site var and say, “What checks do you want to have run against this content.” Whether you want to check the text for URLs, or just domains, or profanity or whatever it is.

So it was a simple API. It was a pre-check before submitting, and then check on display time, because there are some things that we couldn't immediately catch at submit time that we later deemed bad and we needed to be able to not render and then cleanup.

[00:15:06] JM: One of the more complex problems in modern site integrity is that of finding bots who do not belong in your site, and differentiating from a bot and a human is actually really hard. It's funny, because in computer science classes you learn about the Turing test and you think about this conversation between a person and a robot and trying to detect whether it's a robot or not.

The domain that we actually have to encounter, the Turing test in real life, is much more rich. We have fully-featured profiles, and these profiles can be trained on other profiles from around the internet. It’s a really hard and multidimensional problem and you add to the fact that some users like to make kind of pseudo-accounts, like Finsta phenomenon, where you make a fake

Instagram to attend events or do things that you want to be pseudonymous. Do we have the tools to differentiate fake accounts from real accounts in today's internet?

[00:16:08] PK: When you talk about today's internet, the class of problems around bots and fake agents is much more complex than when I initially was on the team. I mean, we were concerned about people creating several fake accounts so that they could play Farmville and give each other tractors and things like that, and that was a very different kind of problem.

The notion of a fake or real accountant, it goes along a spectrum. There are people who have accounts that are true reflections of themselves or as true of a digital reflection as you can have yourself. Then people fudge that line a little bit. They tell half-truths about their age or other things like that, and then you get the world where people create additional accounts maybe to do things like play Farmville or they want to have a fake account so they can go to groups that they might not necessarily otherwise want anyone else to know that they're part of, to discuss sensitive topics.

Then you go to accounts where one individual may create hundreds of accounts, and those accounts are intended to lure other people to become friends with that account so that they will click through and potentially go to some kind of pornographic side, or buy certain kinds of products that they thought was being promoted by a real person.

Then you get into the – I think when we talk about the Turing test, there's obvious places where you can fake the Turing test when you're a real human who's taken over the account of another human. So we had this whole class of 419 scams, which is people would have someone else's account and then now they would exploit that network of trust and message that person's friends. Say, "Hey, I'm stuck in London. I lost my wallet. Western Union me $500," and that's not a scalable attack, because, again, you have to be an individual operator to get someone's credentials and start messaging their friends. It was a very lucrative attack depending on where you live, the $500 a meaningful amount of money in any given day.

One of the things that we used to combat that, we built something called Social Capture, and what it was when we detected certain patterns like this, like repeated messaging of friends that you might not talk to frequently from a geography which you don't typically login from, we would block the account from being able to take any further action until they were able to –

After being shown a picture of so many human faces, identify their friends in those faces. That's something that if I were to show you a grid of 4 x 4 of combination of random people in your friends, you'd be able to pick out your friends. But if I try to do that and impersonate you, I wouldn't be able to do that. That was a really – For us, we took something that was a unique property of a social network and turn into a great superpower.

[SPONSOR MESSAGE]

[00:19:01] JM: Every software engineer writes integration, whether we're integrating Stripe, or Slack, or Google, or Facebook, we write code to leverage the APIs and tools of the software world. As an application gets bigger, more and more of these services exist in your app. You have Twilio, and HubSpot, and Zendesk and Salesforce. You begin to want integrations between these different services, and the amount of integration code you have to write grows and grows.

Zapier can simplify the integration process between your apps and services. Zapier is an online automation tool for connecting two or more apps. For example, I can use Zapier to integrate stripe with Google Sheets, and every time a user signs up and pays for a subscription with the Software Engineering Daily mobile apps, their Stripe email address can be put into a spreadsheet and Zapier can make that Google Sheet easily import those email addresses into our MailChimp newsletter, Software Weekly. Then Zapier can make sure that every reply to the MailChimp newsletter sends a message to our Slack, and if that newsletter subscriber is also in our Slack channel, we could send them a message and start a more real-time conversation with them.

If you're looking for a single service that centralizes all these integrations into simple workflows called Zaps, Zapier is the easiest way to automate your work. Find out how Zapier can help your software integrations by going to zapier.com/sedaily to try Zapier for 14 days. That's Z-A-P-I-E-R.com/sedaily. There's probably a way that Zapier could make your software run more smoothly, and if you are just a technical person, you probably have enough spreadsheets, and Gmail accounts, and social media management that Zapier could save you some time personally even if you don't have a business. So check out zapier.com/sedaily right now through November and learn how your API integrations could be managed more easily. Try Zapier for free.

Thank you to Zapier for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:21:30] JM: I had a show recently where I interviewed Kent back who worked to Facebook, and I asked him about why Facebook doesn't use any public cloud technologies, and he said basically it’s because public cloud technology just wouldn't scale to Facebook's demands. It made me more curious about Facebook's internal tools and infrastructure. It’s just some stuff that we haven't really covered in detail.

Can you just give me your perspective on Facebook's internal infrastructure? What it was like to spin up a service and deploy a service in some of the tooling around releasing and scaling?

[00:22:11] PK: We had a dedicated team that did all of our release management and they allowed us to iterate incredibly quickly. I think when you talk about using cloud technologies and building on someone else's infrastructure, I mean, it can’t compare. If you’re a small player, it's great, because it gives you the ability to set up a service very quickly and iterate on your product ideas. But at scale, it’s just incredibly expensive to – We were running, for example, machine learning pipelines and retraining models thousands of times a day because we deal with a very adversarial system. We train a model to learn a pattern of bad behavior.

As soon as we learned that model, we give signal to the bad actors that we figured them out. So they changed their behavior very quickly. So it's kind of an arms race. To run thousands of very large training sets to create models every day on someone else's cloud, it would charge you an arm and a leg. So you have to build your own custom workflows for that.

You want to be able to manage your own data pipelines and how often you release it. Because from an attack going from nothing to affecting thousands, hundreds of thousands, millions of people can take just a few minutes depending on the type of attack. So we couldn't – Aside from the cost of paying someone else for this, we couldn't take that kind of a hit to our ability to move quickly.

[00:23:41] JM: People sometimes assume that Facebook engineering was a carbon copy of Google. They assume that the Google culture was successful and Facebook kind of just copied that. But in reality, the two companies operate much differently in terms of their engineering practices and in terms of their culture. How would you contrast Facebook engineering from Google?

[00:24:08] PK: Oh, yeah. They're night and day different companies from an engineering standpoint. Facebook is an incredibly bottoms up. As a manager, I couldn't tell people what to work on at Facebook, which was different than Google. But at Google, I spent a lot of time writing design docs and going through lots and lots of reviews of a design doc before I could actually write any code. Whereas at Facebook, you start working on the product pretty quickly and you document the parts of it that matter for other teams to be able to collaborate and integrate with your service. But they did things in completely different orders.

For example, at Google, to be able to write in a language or in particular programming language, you had to have a readability review. You got to read a very extensive set of documents and conform to a certain style guideline and you had to have your code reviewed and approved by people who'd been on that list, and it wasn't until you were able to deliver kind of something that matched those standards before you were deemed qualified to write code in that language.

Whereas Facebook, that was – I actually proposed that to a group of people when I first got to Facebook. I'm like, "Hey, Google has this awesome thing. It ensures that our code will be consistent." And people were like, "Our code is definitely going to be more consistent, because we work so closely with each other and we give people really fierce feedback when things are going off the rails."

[00:25:31] JM: So you mentioned something a little bit earlier about the idea that the Facebook culture, it includes this idea that the average engineer, it's conveyed to them how much impact their work is going to have. Can you go a little bit deeper on that and explain in more detail what you meant?

[00:25:51] PK: Yeah. In the early days, we would talk about the fact that there is over a million people using the service for every engineer that we had. So you had a clear sense that you have an outsized impact. You could push code when you needed to. I mean, when I got there, diffs were handing out – Being sent out by email and you would R copy your code to a set of servers. Being that connected to the push cycle, being that connected to a feature going out, you definitely feel like you're having an impact on the product experience directly. I think that's a big part of it.

The way that we rewrite our self-reviews and the way our peer reviews happen also, which was all about the impact. How do you get there is obviously important. You can’t be a jerk as you're building your code, as you’re collaborating with people. But did the things that you worked on have an impact that they change some metric that mattered to your team and were you able to tie those metrics to the larger impact of your organization that fed into the company?

That’s something that I felt that Facebook did incredibly well, which is people new Mark’s goals for the company. They joined the company because of those missions and we interviewed to make sure the people wanted to be part of that mission. People self-select out of it.

That mission drove what our goals were, and our senior leaders made sure that they were able to tie what their goals were doing and how that related to the bigger company goals. Again, in the organizations that I was, it was effectively done where I knew what I was responsible or I knew what my team is responsible for. At the end of the half, I could tell, “Hey, I did a good job here. We moved this metric. It mattered, because there are fewer people whose accounts have been compromised. We are able to reduce the time it takes for us to do classification, which eliminates this whole class of engagement problems that we might be having.”

I mean, I think it was very intentional how at least senior management all the way down enforced this, and as an engineer and as a manager, as a director, I always felt like I was – I knew what was going on. I knew how I was contributing to the company.

[00:28:04] JM: You mentioned the bottom up element of Facebook a little bit earlier. How did Facebook scale that ability to have engineering be bottoms up? Because it seems like at a certain point it would just get too chaotic, and I can imagine in the early days when everybody's really motivated to get the company to take it public. Everybody's got a good amount of equity. So everybody's taking it really seriously. When a company gets to its later stages, I imagine it's really hard. It becomes harder to have everybody in the company maintain a sense of ownership. Is it possible to scale that bottoms up engineering process?

[00:28:49] PK: I mean, again, I joined Facebook in 2007. It was 200-ish people total. Actually, the company was very small, and I left in 2014 after it was public. The overall ability for someone to affect change at the company was really a function of a couple of things. One is that we really reinforced this idea that there's a phrase at Facebook, code wins arguments.

If you have an idea and you're not able to win support for it through your peers or your manager, we have these things called hackathons, and you could use your spare time or during a hackathon to go and actually build something, build a prototype for I and show that, "Hey, this idea is worthwhile. It's actually meaningful in whatever the domain that you're working on.

So hackathons were an incredible way for us to reinforce the values of the company, which is people can work on any idea that they want. They need to show that's a meaningful idea. In a very short period of time, you have to turn that idea into a working prototype, which means you have to boil it down to the very basics. When you think about that, there's something really powerful about that, which is if you're telling me that I can work on anything at this company or that my ideas matter.

Well, then now it's up to me to actually really push my assumptions, and building something is the best way to push your assumptions. You're like, "Hey, this new feature is going to be awesome." "Okay, go build it." When you build it and you see like, "Oh, actually, I didn't think about this and this and that," and your internal distortion field gets evaporated, you see that certain things don't make sense.

Other things that happened in hackathons that were great, which is these were self-organized events and people would put all their ideas on different – Like a WikiDoc, and they would find people across the company who wanted to work with them or wanted people to volunteer for their project. So we were able to get people from all across the company to work together and collaborate with each other very quickly.

Again, I think that's incredibly powerful, because you have an opportunity there to potentially change the course of the company. You have an idea. You can show other people that’s meaningful and you can buy yourself some time to actually explore it more deeply.

But as it scaled, every company, as you go from tens of people, to hundreds of people, to thousands, to tens of thousands people, the company has to change. As a 100,000 person company one day, not every engineering and PM and designer is going to be able to change the course of the company, and I think they have to recognize that that’s the company that they're joining, because if they really want ultimate autonomy and ability to impact the company, then they should probably go work at a 10-person startup. But then they have to also live with the consequences of a 10-person startup, which is more than likely it's going to fail.

[00:31:44] JM: Describe Facebook on boarding. How was the onboarding process a key component of the Facebook culture?

[00:31:50] PK: Yeah. So onboarding is the first, I think, three days of a new employee’s time at Facebook, and they brought different people to come in and talk. Talk about the values. Talk about the mission. Give a history lesson of the company. So I was one of the speakers. We had other people like Chris Cox and other folks from across the company who’d come and speak.

Everything that I talked about was to reinforce the values of the company. I prefaced it with, “Hey, look. These are the values that make Facebook great in the mission that we’re solving and the problems that we’re solving. So move fast. Break things. Build trust. Focus on impact.”

These are great values when you're trying to build a set of social products where you need to iterate quickly. You need to be able to rely on your team to trust you, but you have to be able to make mistakes. I think the most important thing that I tried to teach people and I think a lot of people reinforce that and people at Facebook hold true is that innovation and failure are two sides of the same coin. You can't have one without the other.

So if you want to be an innovative company, then you have to reward when people succeed, which also have to be understanding that people are going to fail. If they fail, we get to learn from that failure. We have to embrace the fact they we’re going to fail. Fail hard, fail fast, fail often, and place a lot of emphasis on that. If we’re firing people because they didn't have a great product idea, then we’re really going against the company culture.

So really reinforcing what the values were and also make it clear that there is no hubris here, like these values work for Facebook. If you were to transplant the Facebook values that move fast and break things to a company the makes pacemakers, then you'd kill a lot of people. We really – At least when I spoke about it, I made it clear that like you may have come from a Google or an Apple or even an IBM or a company that had different company values, no one is saying that those values were bad. They work in the context of that company. But now that you’re here, understand that this is the company that – This is the way that this company works.

I think that that onboarding experience and then on also boot camp for engineers are two incredible things that allowed Facebook to maintain a very important cultural identity as it, again, went from tens, to hundreds, to thousands, and tens of thousands of people.

[00:34:16] JM: After Facebook, you worked at Uber. What are the new scalability challenges that you encountered at Uber that you had not seen at the previous companies?

[00:34:25] PK: Yeah. It was interesting, because at Facebook, I worked on site integrity, and site integrity and the growth team, we kind of had competing interests at all times. Actually, I was really good friends with the lead, the eng lead of the growth team at Facebook, always sat next to them. We made sure we did our planning together, because we wanted to work together.

Then when I went to Uber, instead of working on security or spam or anything like that, I went to run the engineering side of growth there. I mean, it was a completely different beast for me. Uber, as it scaled, was all about the local markets. There are restrictions and there are regulations and there are specific challenges on a city-by-city basis. So you can't build one software platform and just release it to the world.

Every city launch required some kind of customization, whether it'd be how we did local advertising, local messaging, integrations with background checks, or in certain countries, credit cards were not prevalent. So we had a build debit card solutions. I mean, it was very challenging just to launch a city and assume that – You couldn't launch a city and just assume that it would just perfectly work. You had to be able to adapt to every single city.

[00:35:38] JM: What was the strategy for building out those specific regional teams for growth market? Because as you are in charge of the growth team, the growth engineering team, you had to build out the these specific regional teams for growth in markets like China, and India. What's the average strategy for setting up a service in one of those markets when you are going to have these region-specific needs, like setting up debit card support or specific regulations?

[00:36:12] PK: I think it's a function of how big of a market it is. Obviously, China and India are massive markets. So you're going to want to invest a lot there. Then you also have to look at the obstacles there. Many of the services that Uber relied on, like Google Maps, doesn't work in China. So we had to integrate with local providers there.

As you experience the product – In the United States, this is a magical thing. You open this app, you push a button, a car shows up, you get to your destination and you walk out. You don't think about anything else. In other places where that's challenged by CDNs not working and not having local maps and not having a credit card on file. We realized that these markets needed engineering.

So in the case of India, we built a local engineering team from scratch, because we needed to build a lot of custom solutions in India because, for example, car ownership is not as prevalent there for a class of people who we thought could be good drivers for the service. So we had to build vehicle leasing solutions and other things for them to be able to get on the platform and scale.

We didn't build an engineering team for every single sub-region, as in like an engineering hub there, because that's just get too hard to manage. So, in a lot of cases, we decided that we're going to have engineers doing rotations, engineers and PMs doing rotations. Visiting the city teams, seeing what are the challenges that our local city teams are having.

A local hub actually has people who onboard drivers and handle rider and driver complaints and they have to speak a language. They have to understand about the local regulations, whether how vehicles are inspected. Any time something would be really far out of the norm for the regular use case at Uber, we'd have to send an engineering team to make sure that our tools work for those folks. Then we had to build in the right kinds of checks and balances to make sure that as your products area of all your tools increases, that any one tool going down could have a detrimental effect to one city and you need to be able to track that in this sea of hundreds of tools.

[SPONSOR MESSAGE]

[00:38:27] JM: GitLab Commit is GitLab's inaugural community event. GitLab is changing how people think about tools and engineering best practices, and GitLab commit in Brooklyn is a place for people to learn about the newest practices in dev ops and how tools and processes come together to improve the software development lifecycle.

GitLab Commit is the official conference for GitLab. It's coming to Brooklyn, New York, September 17th, 2019. If you can make it to Brooklyn on September 17th, mark your calendar for GitLab Commit and go to softwareengineeringdaily.com/commit. You can sign up with code COMMITSED that's, C-O-M-M-I-T-S-E-D and save 30% on conference passes.

If you're working in dev ops and you can make it to New York, it's a great opportunity to take a day away from the office. Your company will probably pay for it, and you get 30% off if you sign up with code COMMITSED.

There are great speakers from Delta Air Lines, Goldman Sachs, Northwestern Mutual, T-Mobile and more. Check it out at softwareengineeringdaily.com/commit and use code COMMITSED.

Thank you to GitLab for being a sponsor.

[INTERVIEW CONTINUED]

[00:39:56] JM: Uber, to me, those early growth years, it sounded so just hard. I remember seeing Matt Ranney who is very senior engineer at Uber give a talk. He gave a talk at KubCon about – It was like how Uber managed the growth of services from 500 to 2000 and how they managed the growth of engineer account from 200 to 2,000 or something like that in just a couple of years. He just looked exhausted. Like I saw him in person, I said, "You just look exhausted." I remember interviewing him later on, and he sounded exhausted, and I couldn't imagine anything else.

This is like a move fast and, I guess, it's sort of like the pacemaker situation where you almost have no choice but to move fast, because this service is in such high-demand, and if you mess it up, it really is, it really does have that level of sensitivity like a heart rate monitoring device or something. So, in an overwhelming engineering job like that, how do you avoid burnout?

[00:41:00] PK: Running. I used to run 36 miles every morning as a way to clear my head, but I'm kind of half joking there, but really being able to carve out time for yourself was really critical for me. I'd never been in such a high-intensity period of my career for such a long period of time. I've worked hard and long, but for bursts, like one month, two month at a time. Never four years at a time.

It's a stressful job when you're building out a service like that, and when the service is growing so quickly and the demand is so high and the sensitivity, as you said, is very high, because when your service goes down, drivers aren't making income. Riders are sitting on the side of the road. People are stranded somewhere. So we took our job really seriously.

Those incredible amount of intensity that you felt when you talk to Travis and other leaders in the company, when you met with drivers. I mean, I took an Uber every single day for 4+ years, and you talk to the drivers and you get a sense of like what the impact that's had on their life. At least, for me, that was really eye-opening that we're building a service and people are actually making their livings off of it. That was helpful for me. I always relayed that to my team, but it was stressful.

I mean, I started, my team was 20 people. Right before we exited from China, I had grown that team to about 250 people in about 2-1/2 years and managing that scale on an organizational level was like very challenging. Hiring that many qualified managers is also really hard. But I guess you have to just assume. If you're working in that kind of space, something that's growing that much, you're going to stress associated with it. You want to hire really great people.

You want to find good friends with in the company that you can actually talk to when things are getting tough and who can just help boost your energy. That's really critical, is find time on your own to make sure you're mentally healthy. So, exercise and other things like that. Find a support network within the company and then just always tie yourself back to what is the mission of the company and find people who you are inspired by and you trust within the company that can really help buoy you when the waves get really rough.

[00:43:20] JM: Do you have any specific memories of the waves becoming particularly rough and having to ride those waves out?

[00:43:27] PK: When we got into China in a big way, that was really overwhelming, because it wasn't just the growth team. The growth team was doing a lot of work there. I mean, we were rebuilding everything from the ground-up in terms of part payment integration, fraud detection, map integrations, you name it. We had to like either build it or work within the company. It's an all company effort, from 100+ engineers on the China team all the way through our infrastructure team, our project managers, our CTO. Everyone was hands-on involved in getting us into a data center there so we can handle the scale.

The amount of money that we were burning through every week to operate in China and the incredible growth rate, I felt this incredible amount of responsibility on my shoulders as did a lot of people. That was for me – I mean, it was very stressful, but it was also incredibly rewarding also to see what we had built. I think that is the hard part, to see from outside, is you see it's a stressful environment, fast-paced company, but internally having this knowledge, like, "Look what we built. Six months ago, this thing didn't exist.

These people weren't using this service. We didn't have this many drivers on our platform." Now that we've done all these hard work, it's there. That was just – I mean, it's beautiful, and I wouldn't trade any of the stress that I had during that time working there for the – I couldn't keep the memories of what we did.

[00:44:54] JM: Yeah. Well, Uber was well-rewarded for that foray into China. I mean, the outcome was quite a windfall. So, kudos to you.

Uber was using public cloud in its early days, correct?

[00:45:08] PK: I think so. I wasn't there in the super early days. So they might've been using a public cloud before I was there.

[00:45:14] JM: Okay. So when you joined, what was the infrastructure like?

[00:45:17] PK: Most things were running internally on machines that we configured ourselves inside of other people's data centers. I'm sure there weren't tools that were being run on public clouds, but a lot of the stuff that we dealt with internally was not.

[00:45:32] JM: It is pretty interesting that you've been in the tech industry for a pretty long-time, a pretty long recent time, and you managed to kind of thread the needle to avoid companies that has invested a lot in public cloud. It's unique, because most of the engineers who I talk to have spent some significant time working in a public cloud environment. Have you spent much time communicating with people who have been working a lot in public cloud? Do you have any perspective for how working in a private cloud environment differs from working on public cloud infrastructure?

[00:46:06] PK: When I was – Right before the Instagram acquisition, I was in communication with their head of engineering, Mike Krieger, because they had some spam related questions that they needed help with.

So, actually, me and a couple of people from my team went over there to start talking to them and said like, "Oh, how can we help you?" Because I had been instructed by someone within the company to go help them, and I didn't know why I was being asked to help them. But they were built on AWS. Then we had a period of radio silence, and then I found out that we acquired them. So, some people on our team helped actually onboard them onto Facebook technology and move them over from public cloud.

I mean, as I guess retired/person on sabbatical, as I'm building my own projects, I am evaluating which cloud to put my stuff on. But I haven't had direct experience building anything on it other than the companies that we've integrated with.

[00:46:59] JM: Within Uber, you worked on growth and then you worked on developer productivity. So, he the developer productivity I believe is much about developing tools to allow developers to work more specifically. What was the Uber internal tooling like and how did it compare to Facebook's internal tooling?

[00:47:22] PK: Well, that’s a really broad question. I guess I'll target it towards one thing, which is when you're a Facebook engineer and you build a product, for example, like a new version of groups or a better photo sharing experience, you can directly experience that product as a consumer.

Uber is a marketplace. So there's a driver side of things. There’s a rider side of things. There's, actually, a marketplace team. There’s pricing. There are a lot of different pieces there, and it’s actually very hard for an individual engineer to test their feature in a vacuum.

So, for a very basic need for our engineers was to be able to take a feature and then simulate a set of trips and see how that feature would behave. Whether it's a user-facing feature, like something in the mobile app, or a backend feature that you need to be able to run a series of trip events against that thing. So those are some of the kinds of tools that we built, which is to make web-based tools that allow engineers to be able to exist in their own small ecosystem against a pretty complicated surface area with a lot of subsystems.

[00:48:32] JM: Since your time and Uber, as you said, you've taken a step back to do some consulting and take a break from being in the middle of the storm in a high-growth company. In that time, reflecting away from the high-growth company world, do you have any reflections on working at these types of companies that you've just acquired? Thanks to the benefit of sometime away?

[00:48:56] PK: Yeah, lots of reflections. I think that I've been lucky to work at some very mission-focused companies, and I think that the most important thing at lease for me was being able to relate to my team how what we're doing affects the mission of the company. I've been blessed to work with a lot of really incredible leaders who have been able to create this kind of – How do you say it? Like an aura, like one of like the best leaders or best examples of leadership that I saw at Facebook was this guy, Jay Parikh. So heads infrastructure.

Jay was really good at letting people know what mattered to infrastructure and what mattered to him. So even when he wasn't in the room, people would make decisions based on like what is infrastructure supposed to do. Infrastructure is supposed to scale. It's most reliable. It's supposed to become affordable. It's supposed to be something that our product folks don't have to think about inherently when they're scaling up their services.

He did just a great job of reflecting that, and I think that that's a great model of leadership. That's one really big take away that I've seen at these companies. When people do that effectively, things just go right, even high-stakes environments.

My manager at Facebook, this guy Arturo Behar, who you had a guest, Nick Schrock on, he actually managed him as well. Arturo was incredible, because he showed me a management style that was less about telling people what to do and more about presenting them with hard problems and helping them ask the right questions and asking them the right questions so that they can construct what that future needs to look like, and then holding them accountable for delivering on it.

Those are the big things about leadership that I think I've taken away from those companies, and then the rest of it around being able to iterate is about how do you make it so that a small group to a medium-sized group of people can iterate really quickly? The real important part there is do you build the right infrastructure? Do you build the right frameworks and do you have the right tools so that people can try a lot of things out quickly? Can I get more bites out of the

Apple every single go around? That I think is going to be a hallmark of innovative companies forever, right?

If you can iterate quickly, if you can experiment quickly, if you can tolerate failure as an organization, then you have a shot. The rest of it will depend on is the idea good? Can the leadership navigate the changes at hand? Can you scale your organization? Can you maintain the right culture and change it when you need to, frankly, to make sure that you're building the right company that builds the right technologies, that solves the right problems?

As I advise smaller companies, I see them sometimes over-optimizing for like the world where they're a thousand person company and really deliberating on things that don't necessarily matter to them yet. Then some companies that are a little bit further along than they've hired and they're in the hundred, 200 person range and they don't have some of the basic things around what does a job trajectory look like for an engineer. What are the competencies that we want to evaluate our engineers against to make sure that they are in their daily execution of their aircraft? Are they delivering towards what are teams are going to make for the mission of the company?

There's a lot of there. I mean, I think having some time away from this and having some time to really think deeply about it gives me a better perspective about what these things mean in terms of building great engineering organizations and then working towards an important, but a hard mission. I don't think that – If you can give me an example of a company that's built something meaningful and done it with no stress, I'll be amazed.

[00:52:44] JM: Do you have any ideas or spaces that you're exploring things that might lead to what you build next?

[00:52:54] PK: Most of what I'm building right now is more of just personal projects for me. I have spent a lot of time looking at the Gmail API. I went from having a personal email and then a work email and I'm trying to be very diligent about how I managed both of them. Just thinking about in organizations, we give people the tools. We give them Slack, and we give them Assana, and give them other things like that. But we don't really train them about what it means to manage their communication and their time effectively.

I think that there's something there. I don't I don't know exactly what it looks like, but I think that when you take the average college kid, you throw them in a company, you give them access to email and different documents and things like that and then you say, “Go. Do your job.” They just, by happenstance, start to learn habits. Some of those habits are good and some of those habits are bad, but I suspect the thing that separates the best engineers and designers and PMs and HR people from the okay ones is how they manage their time and attention.

I think there's something to be built there. I don't know exactly what it is. If anyone listening has a great idea, I wish you well and I hope you solve that problem, because you’re probably changing the course of human history if you can do it properly.

[00:54:12] JM: Last question; what are the lessons that the rest of the software industry can take away from Facebook?

[00:54:19] PK: I think that you have to make a company culture that suits what you’re building. Again, company culture is not the perks. It’s not the free food and it’s not the things like that. It's really like how do you deal with hard decisions? How you deal with conflicts? How do you deal with failure? How do you plan when there’s ambiguity in the world? How do you organize people's time and energy and how do you make sure that you align people's responsibility with the amount of autonomy you’re going to give them? I think that's really important.

I think Facebook got that right. I think a small number of companies have really gotten that right where they built something amazing in they’ve built a workforce where amazing people want to work there for long periods of time. That's really important.

I think the other take away there is that when you knew build a company, if you centered around the engineering manager, you will quickly create an environment where engineers don't want to work. They’d be tolerated. They may be there because they're getting great salary and equity and they see the company is going to be successful. They ultimately don't want to be there for a long time.

If you create a company culture that's great and you empower your engineers and your engineering managers to work towards problems together, you can solve all kinds of amazing problems and you can do it in a way that’s sustainable, that people want to work there much longer than the industry average is – What? I don’t know. Two years, three years, something like that. I think that those are those really important things, and there're a lot of great companies that get one of those or the other. Not very many to get both of those right.

[00:55:56] JM: Pedram Keyani, thank you for coming on the show. It’s been great talking.

[00:56:00] PK: Thank you so much for having me.

[END OF INTERVIEW]

[00:56:09] JM: Podsheets is open source podcast hosting platform. We are building Podsheets with the learnings from Software Engineering Daily, and our goal is to be the best place to host and monetize your podcast.

If you’ve been thinking about starting a podcast, check out podsheets.com. We believe the best solution to podcasting will be open source, and we had a previous episode of Software Engineering Daily where we discussed the open source vision for Podsheets.

We’re in the early days of podcasting, and there's never been a better time to start a podcast. We will help you through the hurdles of starting a podcast on Podsheets. We’re already working on tools to help you with the complex process of finding advertisers for your podcast and working with the ads in your podcast. These are problems that we have encountered in Software Engineering Daily. We know them intimately, and we would love to help you get started with your podcast.

You can check out podsheets.com to get started as a podcaster today. Podcasting is as easy as blogging. If you've written a blog post, you can start a podcast. We’ll help you through the process, and you can reach us at any time by emailing help at podsheets.com. We also have multiple other ways of getting in touch on Podsheets.

Podsheets is an open source podcast hosting platform, and I hope you start a podcast, because I am still running out of content to listen to. Start a podcast on podsheets.com.

[END]