EPISODE 434

[INTRODUCTION]

[0:00:00.6] JM: Tinder is a rapidly growing social network for meeting people and dating. In the past few years Tinder's user base has grown rapidly and the engineering team has scaled to meet the demands of increased popularity. On Tinder you are presented with a queue of suggested people that you might match with and you swipe left or right to indicate that you like or dislike them. Creating that queue of suggestions is a complex engineering problem. Many factors go into suggestions that Tinder gives you; geo-targeting, food preferences, your favorite band, your photos and the people that you have swiped on in the past.

Brian Lee is an engineering manager at Tinder and he joins the show to describe the interaction between the mobile client, the backend servers and the offline analytics and machine learning. We also talk about managing different teams and how to reorganize smoothly as a company grows.

If you like this episode we've done many other shows about scaling companies like Uber, New Relic and GIPHY. You can download the Software Engineering Daily mobile app to find all of our old episodes and easily discover new topics that might interest you. If you don't like this episode you can easily find something more interesting by looking at the recommendation engine in the app as well. These mobile apps are all open sourced at github.com/softwareengineeringdaily. If you're looking for an open source project to hack on, we would love to get your help.

The Software Engineering Daily Open Source Community is building a new way to consume software engineering content and the different projects in github.com/softwareengineeringdaily are wide-ranging. We've got an android app, an iOS app, a recommendation system, a web frontend platform. Again, if you're interested in contributing, go to softwareengineeringdaily.com and go to the Slack channel or go to github.com/softwareengineeringdaily or send me an email; jeff@softwareengineeringdaily.com.

Let's get on with this episode, and I hope you enjoy it.

[SPONSOR MESSAGE]

[0:02:16.6] JM: You have a software project that you want to build, everybody does. I love building products, but I know more about how software fits together that how to actually write the code itself. I don't spend a lot of time writing code, but I do like to build software, that's why I use Toptal. Toptal is the best place to find reasonably priced extremely talented software engineers to build your projects from scratch. You can get a pair of Apple air pods when you use toptal.com/sedaily to work with an engineer for at least 20 hours, and I recommend it. I think it's a great way to build your projects if you don't have time to build them yourself. There's a misconception that engineers have to build all of their own projects just because they're capable of doing that. It's not true.

Toptal has only the top 3% of developers. They turn away 97% of the developers that apply to work on the Toptal platform and that's how you get a matching process that's unlike anything else I've seen in the freelancer marketplace, and I've tried a lot of different freelancing platforms. Toptal has such high quality engineers and they listen through the design specifications that you have. They handpick the perfect developer for your project, and this has saved me countless hours in my development process. There’s really nothing that compares to Toptal that I have seen.

You can get a free pair of Apple air pods when you try Toptal at toptal.com/sedaily. Find an engineer that's going to help you build your side project and get your MVP off the ground. As long as you do at least 20 hours, you get those free Apple air pods. If you've already got a company that you're working on, you can also use Toptal to scale your team and get everything done faster and raise the bar for your engineering or get through that blocker that's preventing your company from getting to the next level. Check out toptal.com/sedaily and find an engineer who will help your project succeed.

[INTERVIEW]

[0:04:37.6] JM: Brian Lee is a director of engineering at Tinder. Brian, welcome to Software Engineering Daily.

[0:04:42.2] BL: Thank you. I'm super excited to be here.

[0:04:44.3] JM: Today we're going to talk about a variety of things. You are in the engineering management side of things of Tinder. We talked to Alex, who is an engineering growth manager for the growth team of Tinder. From you, I want to get a different side of things. I want to talk about more the infrastructure and the development process and the mobile app development process, how that all works. I guess before we get started we should give an overview for the listeners for what Tinder is in case they're not familiar with it. Describe the Tinder app from your point of view.

[0:05:20.9] BL: Okay, sure. Tinder is a location-based mobile application that basically lets users who are mutually interested in each other to match and start a conversation. We also pioneer the swiping experience where you go on the app, you download a list of recommended users you can start swiping with, right? Swiping left means you're passing on the person. Swiping right means you are interested in matching with the person and swiping up is the super like and indicated a strong interest in matching with the person.

Most our users use Tinder as a dating app, but I've come across users that are just interested in meeting new people when they move to a new location and as well — I've actually come cross people that are on Tinder looking for a job. I've matched with two people that were specifically looking for a job, which I actually had a pretty lengthy chat with them about how that was going. That was a really interesting conversation.

[0:06:25.0] JM: What kind of job were they looking for if you don't mind me asking?

[0:06:28.1] BL: They were definitely looking to get in the tech job. I can't get any too much in the details, but I remember one of them were just fresh graduated from — Had an MBA degree and was looking to get into engineering management position. In their profile they actually specifically said that. She wasn't looking for a date and she was actually looking for a job.

[0:06:48.3] JM: Okay. I think this illustrates Tinder is a type of application where the core breakthrough of the app is so interesting that there's a million different ways that the company can grow, a million different directions it can grow. It can grow horizontally and get into new

market. It can grow vertically and focus on improving its own product offering and it's doing both of those types of growth. I think the core offering that Tinder made a breakthrough on was gamifying the relationship introduction experience and turning it into a fun way to meet people with the double opt-in where both people express their interest before either of them feel spammed or fronted, and so the interface is extremely simple, but that simplicity comes from a set of insights, brilliant product insights that I think you kind of have to mess around with the app to really understand why it's so cool and it's so useful and it has so much potential for growth that is being capitalized on.

We talked about some of that in the growth episode, which we'll air before this so people can check that out if they're interested. You're director of engineering, what are the teams that are under you?

[0:08:09.3] BL: Yeah, sure. I have multiple teams that are under me. First and foremost it's the mobile engineering team. It's divided up by features or components. We have the discover and profile mobile team, which his responsible for maintaining the recommendation view where you can swipe on users and the profile review where you can actually go in and get the details of the users as well as a close matching team that is responsible for maintaining the match list as well as the chatting experience.

I also manage theWEP team. A lot of people don't know this, but Tinder actually has a WEP product. We're only testing in a few countries right now, testing countries. We have a team of about eight web engineers right now working on a WEP product. I also manage the revenue team. The revenue team is basically responsible for making money, keeping the lights on for Tinder so we can work on new and cooler features.

[0:09:11.4] JM: Let's start with the matching discussion. The matching list that pairs people together to potentially meet up is as foundational to Tinder's current product as the newsfeed of Facebook. This is the core thing that you're engaging with when you open up the app if you're an average user. I imagine there is a lot of engineering depth that has gone into the matching system. I'm not sure to what depth you can go into it, but one thing I'm just curious about is like what's the contract between the frontend and the backend for getting matches as a way to start, a way to start off.

[0:09:59.4] BL: Sure. I guess the really most important thing is not the matching itself, but really just showing the recommendations of users that you potentially could match with. That's the starting point where you would get a list of people that Tinder thinks that you would be interested in matching — By the way, this also confined by a radius by location, people around you that you can actually match with. As you are swiping through the users, swiping right basically indicates an interest. You also mentioned about double opt-in and once the user is being swiped right on and it gets sent back to our server and that user will then service on the other list of that user. If the other user also swipe right on you, then there is a match.

[0:10:52.9] JM: The process of creating potential matches, or I guess recommendations is the term you used. When you log in to Tinder for the first time, I think you need to authenticate with Facebook, it's built on top of Facebook, right? You use your Facebook data. That's like the seed for your pairings.

[0:11:17.4] BL: Correct. We also now offer SMS off as well as Facebook authentication.

[0:11:22.9] JM: Okay. Let's say I logged in with Facebook, so the first time I log in, Tinder sucks in all my interests and my likes and some of my interactions that are public to Tinder and uses those as a way to formulate people who might share the likes, share the interests, share the habits of me in order to find an accurate match. Is that correct? Is that what happens?

[0:11:54.7] BL: That's part of it. There is a few other strong indications on potential interests, such as school, or job, but they all go into considerations. The Facebook, we actually don't really feed the Facebook friend list into our match criteria. More is the interest that we suck in from Facebook. We also use that as a signal to potentially match you with other people that you might be interested in.

[0:12:22.2] JM: How strong of a signal do you get from that, from that data?

[0:12:25.4] BL: Obviously, that's better than no signal, but just having an interest sometimes is not really enough for you to really have a deep connection, but it definitely helps as a

conversation starter, just knowing both of the users that like Golden State Warriors, potentially that could spark a conversation.

[0:12:48.6] JM: This is a very complex problem. I took an algorithms class in college, and I'm sure this is not exactly the same problem, but the problem of stable matching. I think if you take two sides where you're trying to find a matching that kind of can optimize for different utility functions. You've got, for example — I think the classic example is you've got a setup nursing residence that they're somewhere in nursing school or medical school and you're trying to pair them with medical schools that they might want to go do their residency in. This was like the classic problem. There was the question of how do you algorithmically match to maximize the optimal matching between those two sides? Is this problem analogous to it, because if I log in for the first time, there's a huge pool of people that are geographically close to me and there's all these different factors that you could optimize on. You could optimize on location, you could optimize on the shared interests. How hard of a problem is that?

[0:13:56.4] BL: It's definitely an extremely hard problem, especially when you first log in and don't have a lot of signals, because human beings are extremely complex and just knowing what their interest are never enough. Just kind of think about a relationship that you see around you that sometimes it could, you know, that you see a couple that potentially don't have a whole lot in common. Then you also see couples that have a lot in common. While that's definitely irrelevant, but it doesn't necessarily kind of guarantee that is going to be a match that could be made.

The stronger thing though is actually more coming from after you've been using Tinder a little bit and we know the patterns of your swipe being preferences. That's a much stronger and accurate indicator of the types of people that you're interested and matching with.

[0:14:53.7] JM: But most of the swiping is based just on physical appearance, right?

[0:14:57.7] BL: That's definitely a part of it, but a lot of user also click into the profile and look at the profile and school and summary. That's also a big part of the consideration. We do see that people with school and job filled out with a much higher success rate than people without.

[SPONSOR MESSAGE]

[0:15:26.6] JM: DigitalOcean Spaces gives you simple object storage with a beautiful user interface. You need an easy way to host objects like images and videos. Your users need to upload objects like PDFs and music files. DigitalOcean built spaces, because every application uses objects storage. Spaces simplifies object storage with automatic scalability, reliability and low cost, but the user interface takes it over the top.

I've built a lot of web applications and I always use some kind of object storage. The other object storage dashboards that I've used are confusing. They're painful, and they feel like they were built 10 years ago. DigitalOcean Spaces is modern object storage with a modern UI that you will love to use. It's like the UI for Dropbox but with the pricing of a raw object storage. I almost want to use it like a consumer product.

To try DigitalOcean Spaces go to do.co/sedaily and get two months of spaces plus a $10 credit to use on any other DigitalOcean products. You get this credit even if you have been with DigitalOcean for a while. You can spend it on spaces or you can spend it on anything else in DigitalOcean, and it's a nice added bonus just for trying out Spaces. The pricing is simple; $5 per month which includes 250 gigabytes of storage and 1 terabyte of outbound bandwidth. There are no cost per request, and additional storage is priced at the lowest rate available, just a cent per gigabyte transferred and two cents per gigabyte stored. There won't be any surprises on your bill.

DigitalOcean simplifies the cloud. They look for every opportunity to remove friction from a developer's experience. I'm already using DigitalOcean Spaces to host music and video files for a product that I'm building, and I love it. I think you will too. Check it out at do.co/sedaily and get that free $10 credit in addition to two months of spaces for free. That's do.co/sedaily.

[INTERVIEW CONTINUED]

[0:17:47.1] JM: How tight is that feedback loop? If I open up the app and I swipe through 15 people, am I going to get an updated set of recommendations based off of those left and right? By the way, for people who don't know, a left swipe is you don't have an interest in the person.

Right swipe is you are interested in the person. If I give it 15 pieces of signal, is it going to give me some new machine learning based off of that, or is the machine learning more of an offline batch process?

[0:18:20.2] BL: Currently it's more offline process. We kind of reprocess it in an interval. I believe the interval is like twice a day right now. Your swiping of 50 people won't go into the next batch of recommendations right away, although we are trying to get as closer to real time as possible.

[0:18:40.9] JM: Can you talk more about how you batch that, because if I do — I could imagine different models. If I'm going to do 15 swipes in a session, maybe you want to keep all those on the phone and then send them at one hour intervals, send the batches of my swipes back to the server or you could do it streaming one-by-one, and then once it hits the server, there's different ways that you could aggregate those — If you've got millions of users that are having these batches of swipes coming to the system and you need to process them all. Give me an overview of that pipeline if you're familiar with it.

[0:19:18.5] BL: Yeah, sure. The batching of swiping definitely get to the server in real time, because we also need to process your likes and dislikes right away so that I'm sure if you've used the app before, when you swipe right on someone and then you get the it's a match screen, it's definitely sort of a very gratifying moment when you get that, it's a match screen. That obviously has to go into real time, because we need to tell you whether that created a match or not.

The actual kind of processing afterwards, it happened more in an offline batch processing where we basically just load all user's preferences and then kind of run it through our model and basically we have certain scores for users that we can then use it to match with other users.

[0:20:12.5] JM: All my swipes, those are going into a production data store somewhere, because it's important for real time interaction. Then at some point it gets loaded — It gets copied and loaded into some offline analytics system that is going to process those swipes and turn them into better recommendations. What can you tell me about the analytics pipeline?

[0:20:37.5] BL: Okay. There is two things that I want to touch on. One is the production data actually — To do the recommendation, actually that doesn't really — The data doesn't really flow into the analytics database. It just kind of gets processed separately, but it kind of still remain in the production realm. The analytics database is for like business analytics for us to kind of — for business and product managers and executives kind of look at how the users are interacting with our app and try to get a better understanding of how to use it and what features that we should develop on.

It's actually not a real time pipeline from the production database and into the analytics database. There's more reason that we don't have a direct mapping from the production database to the analytics database. One is permissions we have and ops team that kind of run the production database, which for Tinder, there's a lot of PII information that's available in the production database. There is a lot of cleansing that you have to do before you would load that data into the analytics database. Instead of actually batch exporting that data from the production database to the analytics database, we actually have a separate pipeline.

I'll give you an example. When a like or a dislike that comes into our backend, we would store that like in our database and we'll also fork that like into a separate data pipeline which will eventually flow the data into our analytics database.

[0:22:12.9] JM: The separate pipeline, so that's going to copy the data into analytics databases. I imagine, also pipes it into the machine learning processes where you kind of only — You just need the like data for some ephemeral period of time and then you can garbage collect it after you use it to create better machine learning recommendations. Is that accurate?

[0:22:37.4] BL: Yes, and likes and dislikes is just kind of one dimensions of it. We also look at the images of a user's profile, because there's actually a lot of information you can extract from the images. We also do image processing to extract keywords from images to kind of see what the interest are, because you can actually tell a lot about a person by looking at the profile pictures. For example, people who take pictures outdoor or people with pictures with an animal, like a dog. We can basically extract dog lovers, for example, or outdoorsy person from pictures.

[0:23:16.3] JM: That's interesting. When I log in with Facebook, are you going through the pictures also to know more about what the person does?

[0:23:24.0] BL: No, we actually don't. We only kind of go through the pictures that the users upload. There is a pipeline for the user to actually pick and choose which Facebook photos they want to import into Tinder, but we don't just import the photos from Facebook without their permission.

[0:23:41.2] JM: Yeah, I like that. It's less aggressive. I think Tinder is in a place where pretty comfortable, you probably don't want to do anything that's going to like scare the people. Okay, talk a little bit more about the — I'm just very interested in the pipeline where you're copying — The data gets copies. The data gets stored in analytics, which I think is Redshift, right? Then the Redshift database can be used by business intelligence people and then the growth team. In that show, when I talked to Alex, he was talking all about Redshift. Then it's copied into this other pipeline. Can you talk about some of the infrastructure choices of that pipeline? Are you queuing all these things up in Kafka and then processing them off of that, or Kinesis? What do you do in there?

[0:24:30.2] BL: We have a separate data engineering team that's solely responsible during the data pipeline. The current choice is Kafka. They're using Kafka to pipe the data in. We also do use Kinesis in some cases, but the main thing that we use right now is Kafka.

[0:24:49.0] JM: Okay. We dove pretty deep into this one dimension of the recommendations. We've kind of sped forward. I want to zoom back out to mobile and the user experience. Not the user experience, but the contract between the frontend and the backend. I worked briefly at Amazon for about eight Amazon and I worked on this one very small system for just — It was a backend calculation system for taxes. What you learn at these big companies is just that when a company gets to a certain size, it's really hard to manage the different services and the interdependencies, and one place where there is an important interdependency is the frontend and the backend, the interaction between the mobile and the backend side of things. Do you have a general contract or a way of doing things for how the frontend collaborates with the backend?

[0:25:43.0] BL: Yeah, sure. Mostly the front and the backend kind of communicate between each other just through RESTful services. We also have push notifications from the backend to the frontend in the form of either in-app push notifications or just mobile push notifications, or the RESTful services, we do have schemas predefined for API services that we use. For example, if you were to send a like from the client side to the backend, then you send it through like API with a predefined post-schema that we have.

[0:26:23.1] JM: What are some of the big problems in modern development where you have an iPhone and android application that's separate and you have to create a unified experience across the two. What kind of problems does that create for you?

[0:26:37.6] BL: We are actually pretty lucky in that front because the swiping was a very Tinder thing to begin with. We just started out with a very Tinder specific experience. We kind of carried that experience on. If you actually look at our iPhone app and the Android app right now, you probably won't notice a lot of difference between the UIs between the two. You can actually see we have the swiping experience. We also have below the swiping, there's the game pad where you have the buttons where you can click on, which is likes or pass or super like. Those are very like Tinder-specific, right?

We do want to kind of respect the OS choices of UI elements and some of that is the sharing [inaudible 0:27:26.8] and things like that. We do try like the small elements. We do try to abide by the guidelines provided by the iOS and android.

[0:27:36.7] JM: One of the teams that you oversee is the revenue team. What is the revenue team responsible for?

[0:27:42.0] BL: In short, the revenue team is responsible for making money; monetizing the applications so that we can pay the bills.

[0:27:51.4] JM: Okay. How does that translate to engineering decisions?

[0:27:54.8] BL: We have three kind of major revenue streams that we have right now. One is subscription model, where you can subscribe to Tinder on a play store, an app store where you

get Tinder Plus, which is a kind of premium features that you can — For one, you can have an unlimited likes. You could have a boost, which is a feature that lets you kind of jump in front of the line for a certain amount of time. It will also let you kind of passport into different locations so you can match with people within different locations. That's one of our major revenue stream.

Second one is more of ala carte purchasing. You can just purchase a super like, or you can purchase a boost on its own and use that. The third revenue stream is ads. On certain number of swipes, we serve you one ad and those are the three ways of Tinder making money.

[0:28:54.5] JM: One of the things I've discussed with Alex is a comparison between Tinder and LinkedIn, because LinkedIn is a company that also has some significant varietal revenue streams. It has an ads business. It has a business for recruiters and it has a business for users that get a premium service, and I think these account for like 20% and 60% and then another 20% of their revenue. The revenue is heavily split, which is very different than a Google or a Facebook where all of their money is in ads and becomes easier for them to focus on business decisions, because the revenue stream is a little bit more focused, unless they're talking about other bets in the future and they're talking about revenue streams and the different revenue streams in the future. Does that cause any interesting decisions where you have to make tradeoffs between different revenue streams?

[0:29:50.6] BL: Certainly, because out of the three streams, obviously we're not pulling the same amount of revenue in those streams. Some are more important than the others. For example, like we don't serve many ads. Right now is not as important as the revenue stream for us. We definitely prioritize the subscription model a little bit over that model as far as also — Because having ads is also a disruptive. It could potentially be a disruptive experience for users.

[0:30:25.0] JM: Did you do a lot of measurement around that, or are you doing a lot of measurement as to what's the cost benefit analysis of displaying one additional ad?

[0:30:35.2] BL: Yeah, certainly. We do a lot of testing around how many ads we can actually display. Before, it's too many. We closely monitor the swiping weight as well as the retention rate just to see how a user is reacting to seeing different number of ads in a set period of time.

[0:30:56.5] JM: Any interesting results from that?

[0:30:58.7] BL: We have found that users actually not — Although they complain on the app store about not wanting to see ads, but largely we haven't really seen that much of a difference between serving them 20 to 30 apps.

[0:31:13.9] JM: You swipe through so many people that you don't like to match with. Probably not a big deal to swipe past one additional ad, right?

[0:31:24.5] BL: Originally we're thinking that. Just having too many ads a user actually might not like that, but through our testing, actually we didn't find a lot difference between people that get to an ad in 20 swipes or 30 swipes.

[0:31:38.4] JM: The Tinder user base has scaled a lot in the last few years, and Tinder is a dominant force in the dating technology space. I think it was acquired by AIC, which is kind of a conglomerate or different dating apps and news — It was acquired by AIC, right? Or was founded from AIC.

[0:32:03.7] BL: It was founded within AIC.

[0:32:05.8] JM: Yeah. Right. Okay, cool. As it has scaled, what were the aspects of the tech stack that had to be refactored that were breaking under the pressure of the increased user base?

[0:32:19.2] BL: Our user base has kind of ballooned over the years. We started actually building our backend stack off of AWS. Fortunately, AWS actually handle of that scaling for us, so we didn't have to worry too much about scaling the servers or anything like that, but there is still breakages within that stack. One of the examples we had — I'm saying this is probably a year ago where we had an outage, because one of our cluster of caching servers were not enough to handle the load that we're getting at the time, and that was actually a central point of failure to the services at that point.

Realizing that, what we had to do is, one, in the short term, just to scale out that cluster to be able to handle the traffic that was coming in. We also try to look at the design of our overall services architecture just to try and avoid these central points of failure so that having one cluster down will not actually break the entire cluster of the Tinder backend services.

[0:33:27.7] JM: How is that growth of the user base translated to growth in the company? Has it changed the org structure of the company?

[0:33:38.5] BL: Definitely. The org structure is — It's definitely reflective of this is a needs, right? The more users we have, we definitely need to infrastructure to keep up with the user base. We also have to hire mobile engineers to build out features to kind of keep up on the feature demand from the users. We started off on the mobile side of things, I think it was four mobile engineers; four, five android engineers just about a year and a half ago and about the same amount of iOS engineers.

They were basically just kind of one big team, like one mobile team that was kind of handling all the features that were coming in. As we were getting more feature request, as we were trying to keep up with that demand, we had to kind of spinoff like several teams to handle more features. So we started to go more towards a feature or component-based teams where a feature team would now have potentially two iOS engineers, two android engineers and one backend engineer and then that would actually go and maintain one feature. For example, the recommendation of cards.

As the team grow, other problems actually surfaced. For example, once the team — Once we have multiple feature teams, we started getting very fragmented development on architectural decisions that were made along the line even just in the iOS or android app itself.

[0:35:13.1] JM: What's an example of that? The fragmentation?

[0:35:15.9] BL: For example, we have three different teams that are working on — One team could be working on the pre-matching experience. One team could be working on the post-matching experience, because they are more focused on delivering their features. They would kind of try to pick and choose architecture that would kind of work more towards kind of their

development goals. Even the architecture might start to diverge a little bit, because they have different timelines. Potentially, they're working with different skill levels with the iOS engineers and android engineers. You just kind of start to see the core base being fragmented at that point, and that's when we started realizing that we also need a central kind of platform team to kind of hold, kind of glue these teams together and make sure we introduce certain architectural guidelines and certain ways of doing things.

One team was using MVP as their kind of overall architectural guidance. Another team started adopting clean architecture, but because these teams are kind of independent, and so we would have to have one overarching platform team that can actually make certain architectural decisions or kind of build a common infrastructure layer so that every feature team is more on a similar architecture so that they can have a common infrastructure to build upon.

[0:36:44.2] JM: I'm not familiar with these two architectural patterns that you described. Can you describe it a little more and explain why they conflict with each other?

[0:36:52.6] BL: They don't necessarily conflict with each other. MVP is —

[0:36:56.6] JM: MVP.

[0:36:57.3] BL: MVP, yeah.

[0:36:58.1] JM: Minimum viable product.

[0:36:59.6] BL: No. It's a model view presenter.

[0:37:03.5] JM: Oh!

[0:37:03.7] BL: Yeah. It's basically one of the fairly popular architectural decisions that you make or when you choose when you actually start building an app. It actually works really well when the application is actually fairly small and the team sizes are fairly small. As soon as you scale out to a bigger team and architecture, it just won't really be able to scale as the features that you're coming in. Other architectures might actually have to adopt other architectures to be

able to help that. Some other team start adopting clean architecture. They don't necessarily conflict with each other, but you end up having two different approaches. When a team member potentially could switch from one team to the next, or sometimes you have to co-review each other's code. Then when you have different approaches, sometimes there's more contact switching and you have to adopt to other teams, sort of co-style. You just kind of start to see things diverge when you don't have an overarching platform team to kind of hold everybody together.

[0:38:08.3] JM: Explain what the platform team did. You decided you needed a platform team. Did you gather that from disparate teams or did you have to hire for it? Then what did they do once they got assembled?

[0:38:20.8] BL: We just started kind of having — First thing that we did was that we're just having individual engineers from that platform, like iOS, android backend. First, they would meet regularly. That start as the base of the platform team. No one was specifically assigned to the platform team, because we're just still trying to scale out the team size. We'll just kind of distribute the platform items amongst the feature team members. For example, let's say that you have to write API service layers and that actually would be shared across different feature teams. Who is going to write that? When we didn't have a specific platform team, that just kind of got distributed amongst all the feature teams, but then we realized that that's actually something that we needed a central team to do so they can actually be reused by several teams. We started just hiring more people and have designated people to join the platform team. We started out with one person and then quickly scale out to about three people right now.

[0:39:24.5] JM: What was their approach to doing the refactoring? Once you had a dedicated person to kind of do the platform engineering fulltime, what did they start to do?

[0:39:36.9] BL: The first thing that they did was kind of trying to analyze what are the hardest problems. What are the most common pain points? For example, people were even using different image libraries to actually load images. We actually were using, I think — I believe we were using up to three different image loading libraries, because they had different advantages to their own. Some might be actually better in doing GIFs. Some might be better at just doing a

JPEG download. The first thing that the platform team comes in and actually kind of just see a common pattern of what are some of the common things that different teams are trying to use and kind of extract the commonality out and then kind of start writing a library. Then once that library is written and it gets distributed to different feature teams and so they can start adopting that.

[0:40:33.7] JM: What's the release process for Tinder?

[0:40:35.8] BL: Right now, the release process — We're trying to adopt the train model. We have a two-week release process. Every two weeks we are trying to release a new product — Or a new release every two weeks. It starts with being feature complete. It starts with planning, where let's say that if we want to release a feature somewhere down the line, we'd first do a planning on how many weeks of development that is going to take. From there you figure out what release I might fit into since we have a two-week release train, then you can sort of predict that your feature is going to land in that particular release. Then you kind of work backwards into here's when I actually have to get the build to QA and here is when I actually have to start doing the translation.

We do have a preleased checklist that each team has to go through. Some of the items such as you would need to get the analytic events in by a certain time and you would also have to get to get the translation in about a week before the release. As you get closer to the release, we have to start the testing process. The first thing that we do is we have an internal dog food process where we — Tinder has about 250 employees. That's about 200 some testers that we can utilize with different models of phones and different iOSs. About a week before the release, we would find a build and we would distribute it across the company so that people can start testing the prerelease build and they would keep filing bugs as they find those bugs and we would have about a week to kind of fix all those bugs before the final release.

When we have a few featured teams, one of the challenges that we ran into was how do you then kind of make sure that if a team doesn't make a cutoff of the release date, how do you actually schedule that in or how do you turn the features on and off so that you can like put that into a modular release? We had to introduce like feature flagging, and we have a pretty complex

feature flagging system where you can say, “I can roll this feature out to a certain percent of the users.” If we detect problems after the release, then we can actually roll that back.

[SPONSOR MESSAGE]

[0:43:16.6] JM: The octopus, a sea creature known for its intelligence and flexibility. Octopus Deploy, a friendly deployment automation tool for deploying applications like .NET apps, Java apps and more. Ask any developer and they’ll tell you that it’s never fun pushing code at 5 p.m. on a Friday and then crossing your fingers hoping for the best. We’ve all been there. We’ve all done that, and that’s where Octopus Deploy comes into the picture.

Octopus Deploy is a friendly deployment automation tool taking over where your build or CI server ends. Use Octopus to promote releases on prem or to the cloud. Octopus integrates with your existing build pipeline, TFS and VSTS, Bamboo, Team City and Jenkins. It integrates with AWS, Azure and on prem environments. You can reliably and repeatedly deploy your .NET and Java apps and more. If you can package it, Octopus can deploy it.

It’s quick and easy to install and you can just go to octopus.com to trial Octopus free for 45 days. That’s octopus.com, O-C-T-O-P-U-S.com.

[INTERVIEW CONTINUED]

[0:44:49.1] JM: Yes, and I think I talked to Alex about the difficulty of testing for UI bugs with unit and integration tests, and in an app like Tinder where the core offering of the product is fairly simple. Seems like a lot of the errors that can arise would be UI errors or perhaps something that lags a little bit and it’s a little nondeterministic. To what degree can you do unit testing and integration testing and to what degree do you rely on that process of releasing the app to the 250 employees at Tinder and having them dog food it?

[0:45:37.0] BL: Unit testing is a very important part of the development process. It’s there not just to enforce the quality of kind of the final product, but it also kind of really make the developers think how to make their code testable and modular before it even gets to the hands of the dog food, the company dog food. We don’t actually mandate a percentage of co-coverage

for unit tests, but we found it to be very useful actually to produce a higher quality code. Integration testing is actually a really interesting piece, because some companies do a lot of integration testing and some companies don't do as much. I think Tinder is at a place where we're not doing a whole lot of integration testing, for one, because we haven't really found a lot of value in doing a lot of integration testing just yet. We actually don't rely a lot on it before we get into dog fooding. We rely more just on the unit testing to kind of catch the small issues before it gets into dog food.

[0:46:45.0] JM: Are you making any inroads to getting to a continuous delivery process?

[0:46:50.1] BL: We certainly want to strive for that, but as you know, it's actually very challenging to have a continuous delivery for UI because, for one, it actually changes quite often. The main reason why we didn't do a lot of integration testing is because, especially on the UI, it's because our UI actually changes quite often, especially kind of right before we ship there. It could be a lot of UI tweaking that happens. There could be a lot of changes that on the UI before that. Our integration testing would actually break quite a bit before the release. That's why it gets very expensive to maintain that test weight as well.

[0:47:34.1] JM: Understood. What's your interaction with Alex and the growth team?

[0:47:40.0] BL: Yeah. Our team actually run fairly independently. I mentioned earlier that all engineers, they kind of interact with each other on a platform level, but that's more on the individual level. Just from a team perspective, we do have some key metrics that we have to make sure that are not affected before some of the growth features can actually be rolled out.

One example is that — I remember Alex's team was working on something, like he had a KPI that he was trying to improve, which was to improve the click through rate of a profile. He was testing on a feature that super swiping up off a profile card it's opening up the profile. That was conflicting with the super like features that we had, which is like super like is one of the ala carte purchase features that have at Tinder, which is a big part of the revenue. When he was experimenting with the swipe up to open up the profile, we actually saw the Tinder super like purchase metrics go down. We realized that, "Oh, there's a negative impact on the super like purchasing." There was an implication on revenue metrics. Then that feature had to be rolled

back. The major interaction is really — The growth team, when they introduced their features, they have to kind of make sure that it doesn't impact some of the key metrics that we have on the revenue team.

[0:49:13.4] JM: What about the feedback loop between the customers and the users of the app and engineering. How do you detect that something is going wrong and work those errors into the engineering process?

[0:49:28.3] BL: One of the best way for us to do it right now, or one way that we rely on a lot is actually talking to our customers directly. Most of our engineers have a test account. Some even have a real account of Tinder account when they actually test the app. We match and chat with a lot of our users. Most of the time we actually put Tinder on a profile and we get a lot of messages from our users, they're really interested in learning more about Tinder. Sometimes they would just kind of send us [inaudible 0:50:01.4] or send us suggestions, like we don't even have to ask for suggestions. They just kind of come in when we test on Tinder. We have found that to be a very, very good way of getting feedbacks from the users. I will say that we've been able to found bugs and we've been able to have a lot of good suggestions coming form that front.

[0:50:22.3] JM: I want to wind down our conversation by talking about management. What's the biggest lesson that you've learned about managing people from working at Tinder?

[0:50:31.9] BL: The biggest lessons I've learned about managing people. When I joined Tinder, we were at about 50 engineers. That was only a short year ago and we have grown a lot since then. I think we're at about 120 people. We've more than doubled the size. When we were growing at a very fast pace, we also have to pay really close attention to the cultural shift that's happening. We wanted to kind of maintain still the kind of startup culture that we had and moving towards a bigger team.

There's a lot of things that we have to kind of keep monitoring and make sure that as the team grow, some of the growing pains wouldn't be so apparent. For example, as the team got bigger, communication got a little bit harder across different teams. We just kind of have to make sure

we keep tabs on different teams and make sure that we're all communicating between all the teams when we are growing.

[0:51:38.2] JM: What's a mistake that you've made while managing engineers?

[0:51:43.6] BL: I remember when I first got started as a manager, I was previously an engineer myself. When I first became a manager, I wanted to still be very, very involved with the implementation process or the design decision process and I would always try to provide feedbacks and I would try to really provide guidance on how we should design a particular feature.

That turned out to be something that some of my engineers didn't like very much, because they wanted to be able to kind of experiment and they wanted to kind of have to be able to hands-off so that they can actually have more control over the design the themselves. That's definitely one of the mistakes I made was just really always trying to be involved in the beginning.

[0:52:31.2] JM: have you learned anything about human psychology? For example, motivating people or choosing the words correctly to give feedback or noticing people's incentives or maybe somebody left the company and you were like, "Well, I had no idea that person wanted to leave. I had no idea that person was not having a good time. If only I would have done X or set the expectation differently, that would have been a better outcome." Has it taught you more empathy?

When I interview people about management they often say things like that, like empathy or better communication.

[0:53:09.0] BL: Yeah, sure. I found managing people sort of like kind of very similar to kind of just the basic — If you look at that triangle where like the basic human needs went. At the very bottom is kind of your basic human needs; hunger, and finding a place, and then you're moving on up just to try to be more satisfied, is being able to have a job and just basically materials that you'd get, right? Then moving on up is actualization and prestige and that kind of thing. It's very similar with management where you start at the bottom. Basically, you want to motivate people

by kind of providing places and just as well as just praise and punishment. That's kind of sort of the very basic management. That's how you kind of get people on the superficial level.

[0:54:03.4] JM: As a manager, are you talking to the employees, to the engineers as if they're on a four-year — Because I think at Silicon Valley and engineering everywhere, you often get options, stock options on a four-year vesting schedule. Most people, I think, stay at a company for 12 to 18 months. I find that sometimes the engineering management will talk to the underlings and be like, "Hey, what are your four-year goals? What do you want to accomplish while you're at the company?" when actually they should be talking about 12-month time horizons or 18-month time horizons. Is that at all of an issue when you're trying to help the engineers grow in the way that they want to and you have to figure out the right time horizon for that growth?

[0:54:47.9] BL: Yeah. I always kind of focus on the personal growth. For example, you have to really have to figure out what motivates the team members. To some people, it could be monetary, it could be wanting to do more things, getting more scope. For different people, it could actually be technical challenges. The main thing is trying to figure out what areas they want to grow in and kind of focus on that and try to expose more opportunities for them so that they can grow in that specific area.

[0:55:23.1] JM: All right, last question. What would you personally like to do to grow within Tinder? What are your personal goals for personal growth and how do you see Tinder itself growing in the next, let's say, five years?

[0:55:40.1] BL: How I see Tinder growing in five years. I do see Tinder kind of start branching out towards more on — And just instead of connecting people more for dating purposes, I see Tinder kind of start being more of a platform just to connect people in general, connect people with similar interests. Connecting people with similar goals. That's kind of how I see Tinder growing in about five years.

Particularly for myself, I still want to remain very technical and try to help scale out Tinder as we are actually getting more users. I would still try to stay more on the implementation level and

more on the backend and help Tinder grow our backend services to satisfy our growing user base.

[0:56:30.4] JM: Alright, Brian. It's been great talking to you. It's a really interesting conversation. I find Tinder to be a fascinating app, and I've used it before and I think it's fantastic how it connects people and the interface, and incredibly high quality also just in terms of the reliability of the app. Great work, and keep up the great work.

[0:56:52.0] BL: Great. Thank you. Thank you for having me.

[END OF INTERVIEW]

[0:56:55.5] JM: If you are building a product for software engineers or dev-ops engineers, consider advertising on Software Engineering Daily. There are 24,000 engineers that listen to Software Engineering Daily on a daily basis. If you've got a product or a service that you would like to get into the ears of those developers, we'd love to you have you as a sponsor. You could send me an email, jeff@softwareengineeringdaily.com, and I'd be happy to tell you more about our sponsorship options and some of the success stories. We've got many repeat sponsors, like hired.com, DataDog, MongoDB, Amazon Web Services. These are big companies that know how to market to developers and they have their advertisements run on Software Engineering Daily.

Send me an email, jeff@softwareengineering.com, or if you work at a company that you think should sponsor Software Engineering Daily, tell your manager and have them send me an email.

Thanks again to listening to the show and all the listeners who support the show by checking out the advertisements, you are much appreciated.

[END]