EPISODE 1219

[INTRODUCTION]

[00:00:00] JM: Email has become such a routine feature of knowledge work that we often take it and the entire email client we use for granted. While advancements such as intelligent spam filtering have improved the experience, many email clients retain the same basic structure and offer a largely similar work experience. Superhuman is building a modern email client meant to reimagine email from the ground up. Superhuman is built to be fast and seamlessly integrates insights from social networks such as LinkedIn. It offers features such as undo send, AI triage and mail status tracking. Superhuman works offline using service workers to serve cached assets when a network connection is not available.

Emuye Reynolds is the head of engineering at Superhuman. She was formerly a senior software developer at Apple and led the development of the UI for Apple TV. She joins the show today to talk about what's lacking from traditional email clients, what engineering challenges her team faces and how Superhuman's new features represent an evolutionary step towards the email client technology future.

[INTERVIEW]

[00:00:58] JM: Emuye, welcome to the show.

[00:00:59] ER: Thank you. Thanks for having me.

[00:01:02] JM: We previously did a show with the Superhuman CEO, Rahul, and in that episode we covered a lot of the high-level challenges, design challenges of building Superhuman. And in this episode I think we'll cover some of the same stuff but also dive deeper into some specific engineering challenges. I want to start off by asking you a question that I also asked him, which is what are the canonical challenges of building an email client?

[00:01:29] ER: One thing I'll say is that the surface area is really large. So the things you need to build to even arrive at a basic email client is it's a lot of stuff. And then the way we go about building it where we treat speed as a feature, we want it to be incredibly robust and we also want it to be really remarkable in quality, that creates another challenge. Like any time you're trying to balance performance, reliability and quality, there's some tension there.

[00:01:58] JM: Do you want to go deeper into that, like into the canonical challenges? Like given that you have been working on the engineering side, what are the engineering problems that you see over and over again?

[00:02:10] ER: Syncing is one of the engineering challenges trying to keep an inbox in sync. When I talk about reliability I'm really thinking about offline. And offline is a challenge in general, but flaky network is even harder than offline, right? Like intermittent connectivity. So that's absolutely one of our challenges. The sort of vastness of data, right? You're dealing with individual email threads and messages each of which has there's just so many different kinds of things that we see. That's another challenge, like parsing the different content and making sure it renders correctly. Certainly something we spend a lot of time thinking about.

[00:02:47] JM: There are so many email clients that are out there and yet there are new ones that are constantly being developed. What are the places where there are room for innovation in email clients?

[00:03:01] ER: Yeah, that's a great question. I think one of the main areas is just around the user experience. A lot of what you see is people building and companies building the same email client over and over again. The same experience that hasn't changed in years, and I think there is – There's just a general product challenge there, right? Anytime you have billions of people using something, they get used to using it and trying to introduce new behavior can present a challenge, but I think there is a lot of room for companies to try to innovate there and to be opinionated. It could be things like Superhuman like treating speed as a feature. Like what we do on mobile, trying to allow the device, allow users to interact with one hand. It could be other types of input like speech. I think speed can also extend to like the speed people are typing and trying to allow more writing assistance. It really comes down to the user experience.

[00:03:51] JM: Does user experience design, does that translate to tough engineering problems? Like I can imagine making a really smooth user interface or making a user interface that has “AI” built in. These are things that translate to tough engineering problems.

[00:04:09] ER: Yeah, I think so. I think it turns out to be sort of, like I said earlier, the balance, right? Like if you're just trying to build a really innovative user experience, I don't think that's particularly challenging. But if you're trying to build it in a way that is incredibly fast, it also allows a user to be really powerful, but you want it to be easy to use. I think those things present tough engineering challenges. Also reliability, right? You want every interaction to be extremely quick but you also want it to ensure that the user's intent is captured. I think those do present very tough engineering challenges.

[00:04:43] JM: How do you divide your time in your engineering role?

[00:04:48] ER: I think it’s different every day, every week, every month, but I have a few things that. I’m really focused on I really focus on building out our engineering team and culture, and that's growing the team as well as making sure that we have an organization that can support happy engineers that are working efficiently. I spent a lot of time on the product side and really focusing on delivery and execution. Making sure that a lot of project management tasks are on track. I collaborate a lot with our product team and with our CTO. I also spend a bunch of time actually with my hands on the code. I think it's really important that engineering leadership especially with small teams, that engineering leadership continue to sort of really be hands-on with the problems. So it's a lot of those kinds of tasks. Thinking about our overall company and business strategy and how that relates to engineering and how we can really drive it as an engineering org.

[00:05:41] JM: Let's take a top-down approach. Let's start with the frontend. What is your tech stack for the frontend to Superhuman?

[00:05:49] ER: The frontend on desktop is JavaScript, React, Electron. We use WebSQL. We actually just did a conversion from CoffeeSript to JavaScript and are beginning to explore TypeScript, which I think would be interesting.

[00:06:04] JM: And is an Electron app?

[00:06:07] ER: So we have a Chrome web app and we also have an Electron wrapper.

[00:06:10] JM: Okay, got it. What kinds of engineering challenges have you found in building the desktop web application?

[00:06:18] ER: I think one of the challenges is most of the time desktop apps are not storing a lot of data, right? They're more transient, and we have a really thick data layer on both our iOS and our desktop clients. That type of – It kind of comes back to the sort of syncing challenges I mentioned earlier. But state management around a really thick data layer is extremely challenging. The other is performance, right? Like pushing. In many ways we're pushing web apps beyond what some people even think are possible and sort of achieving our performance goals, it's a constant challenge, right? Any time we have features, anytime we – It's something we revisit over and over again to really try to optimize.

[00:06:58] JM: And as far as the networking between the frontend and the backend, what are some of the networking challenges? Do you have to – Is the client pulling from the server? Or is the server pushing to the client? And how is that interaction functioning?

[00:07:17] ER: Yeah, actually it's both. Depending on exactly what type of data, we do have push mechanisms, we have pull mechanisms, and we kind of balance those. One of the things we spend a lot of time thinking about is how to limit the different network connectivity so that we can – There's a lot of things that we could be fetching or that could be being pushed at any given time. And so trying to be able to turn off the data that we don't need in a particular instance and turn on the data that we do need is definitely a challenge.

[00:07:46] JM: And for communicating that information between the client and the server, are using GraphQL?

[00:07:53] ER: No. We're not using GraphQL.

[00:07:54] JM: Okay. Or are using GRPC or do you have any interesting networking technologies you can discuss?

[00:08:05] ER: Nothing particularly comes to mind in terms of interesting network tech. I'm trying to think of what might be interesting there. I think one of the things that's interesting is we've been building support for O365, building on the support for Gmail that we already have. And the API layers for the two services are super different. O365 does have a model that's much closer to GraphQL and Gmail's is much more like a traditional REST object structure. So that I think is going to be quite challenging for us to sort of massage these two different inputs into a similar experience.

[00:08:44] JM: The mobile clients, how do you keep the user experience consistent between the mobile clients and the desktop web client? Is that at all a design challenge?

[00:08:55] ER: Yeah, it's really challenging, because we want to keep it consistent not necessarily to our desktop client, but to the Superhuman experience. But we also have to be true to the platform, right? Like there are certain paradigms that just don't make sense on mobile. One of the things that I think is really different is just the idea of focus. When you're interacting on a desktop client, you have a mouse, you have this really specific focal point, and you can also navigate with keyboard, right? Which means you're actually navigating through the elements that are focused, whereas on a mobile device, especially on a phone, everything is touch-based. The targets need to be much larger. There's not really a concept of focus. So we've been spending a lot of time thinking about how to translate some of the concepts and how to translate some of the speed that we get on desktop to mobile.

One of the things that's really unique about our desktop app is how keyboard-driven it is. Users can fly through email using nothing but the keyboard, right? And that's one of I think the things that really set Superhuman apart. Like it's something we're super used to as engineers using IDEs. On mobile there isn't – Especially on phones, there's not really an equivalent to keyboard shortcuts. Keyword shortcuts also in addition to being able to go really fast, you have hundreds of options at any point in time that are all equal priority within the app. So you sort of don't have to be as an opinionated when you're thinking about product design because you can use infinite shortcuts.

On the phone you have much smaller real estate. So we've translated some of these things, shortcuts I think. We've tried to use gestures really heavily. As I mentioned earlier, you can use the app with one hand, which really allows people to go fast instead of pulling to refresh, which is a really common paradigm in apps. For us, pulling down in the inbox open search because it's just a much more common action, and at this point you don't really need to refresh an email app. It's going to stay up to date. But yeah, that's something we spent a lot of time thinking about.

We have a feature on the desktop that we call Superhuman command, which is kind of like a traditional command palette that you might see in an IDE. You can open it and it allows navigation to other parts of the app. It also allows you to take actions on specific context, like if you're looking at a conversation. And that was one of the most interesting things that we translated to mobile, because as I mentioned, there's no concept of focus. You don't usually have a keyboard on your phone, but you do on your iPad, right? So there's this really split, this unique experience of having to design for mobile. You're designing for something that's really close to desktop, which can also be used as a touch device and then something that's really far away from a desktop device. And so we did this, we brought a command palette to the phone, and I don't really think there are many examples of this especially where you can take action on a specific context. You can also navigate to any part of the app. I think it's really powerful and it's going to really allow our users to go fast in the product.

[00:11:53] JM: So can you tell me a little bit more about the management of the mobile and desktop experience, like how you manage keeping the features reasonably consistent between the different platforms?

[00:12:06] ER: I think that's a great question. So for some time, the platforms were actually in different places, right? We started our desktop app about a year and a half earlier than we started our mobile apps. And that meant the mobile app was still working on kind of core functionality where the desktop app was beyond that, and over time those sort of converged and we caught up. The management was a little different in those early days versus what it is now.

So in terms of how we're organized as a company, we structure largely by technology. So we have a desktop team, a mobile team, a backend team and product and design. But when it comes to building out features and functionality, we formed almost lightweight matrices for those particular features and we dissolve them when the projects are over, and those are cross-functional, right? So you would have people from backend, from desktop, from mobile and product and design so that we can really collaborate on that functionality while we're trying to build it. That's one of the things that we do.

We try to really encourage engineering to cross platforms. Now, of course people have their particular expertise, but I think it's really important in order to build a cohesive experience especially when you're doing something like email, which is really an ecosystem product. I think it's really important that people see the problems, the technology problems and the product challenges from not just one device, but kind of holistically. So we do encourage cross-platform development quite a bit. And certainly this architectural discussions across the teams so we can really think about these challenges at a level that's not specific to any particular technology.

[00:13:48] JM: So let's talk a little bit more about the backend. Can you give me an overview of the server infrastructure for Superhuman?

[00:13:56] ER: Sure. So the backend is written in Go and we use Postgres. We're using Kubernetes to manage our specific services and we're also on Google Cloud, and we have Redis for a few different things as well.

[00:14:10] JM: And can you tell me why? Just give me the justification like why does Go make sense and what are you using Redis for?

[00:14:18] ER: Sure. So when we started our backend in 2015, 2016, I’m not actually sure that code did make sense to be honest. It was still pretty early. Now I think it does make sense. It's evolved quite a bit and it allows us to – A lot of what we're building, we don't have to build a ton of boilerplate for and we've sort of developed that over time. But I think a lot – I don't think our language choice on the backend is going to make or break our product to be honest. So I think Go is modern. It really helps in terms of adding people to the team. People are excited to work on something a bit newer and that's still evolving at the pace that Go is.

[00:14:59] JM: And can you tell me more about the cloud services you use?

[00:15:04] ER: Sure. Yeah, I can kind of give you just an overview. We have a few different services. We have a media proxy that we use to proxy our images. I'm trying to think of what might be interesting to sort of talk through. We have like our core backend. We have analytics kind of run through another separate service. We have our Repl. Yeah, those are primarily our services.

[00:15:30] JM: But as far as like managed services on the cloud provider, so you mentioned Kubernetes, for example. Are you using like a Kubernetes managed service from AWS or from Google?

[00:15:42] ER: From Google, yeah. I don't spend a ton of time day-to-day thinking about our backend infrastructure, but in general we kind of split it into services like our analytics, our media proxy. And we have a few other internal services that we're running on a Kubernetes cluster.

[00:15:58] ER: What is a media proxy?

[00:16:00] ER: It's the way that we load images on the clients. So yeah, we're proxying media through our backend.

[00:16:08] JM: Interesting. What's the purpose of that? Could you tell me? I've never heard the term media proxy before.

[00:16:14] ER: Oh, interesting. Yeah, it allows us to authenticate our requests for media for things like images that need to load content within emails, things like that.

[00:16:27] JM: Got it. So you don't spend much time thinking about the backend. Is the backend like fairly simple to build? Is that why? Or is it just not under your dominion?

[00:16:40] ER: I think it's grown quite a bit and it does. I wouldn't say that it's simple, but we do have a lot larger data layers on the clients than you would typically see. But yeah, from an overall like day-to-day, like building perspective, it's not in sort of my core area.

[00:16:56] JM: Got it. Well, let's talk a little bit more about when you say those data layers on the clients, can you tell me more about that?

[00:17:03] ER: Yeah, sure. In order to support offline, our clients need to be able to interchangeably and sort of simultaneously talk to either a local data store or a remote data store, which is another thing that's quite challenging, but that does mean we're storing in a lot of emails for search and for being able to go back in time in your inbox. We store a lot of that data directly on the client.

[00:17:29] JM: So has it been a challenge to determine how much data is the optimal amount to store on the client?

[00:17:36] ER: Yeah, definitely. I mean, I think the optimal amount is all the data, right? But of course we can't store all the data. Where this gets particularly challenging I think is when you're thinking about like a mobile device that isn't necessarily plugged in, right? You're using the battery anytime you want to update that data. And I wouldn't say that we worry too much about taking up space on a phone or a desktop at this point. It's a little bit of a concern, but I think phones and the hard drives and drives are getting larger and larger, but it is really challenging to keep the data up to date, right? To keep getting the data and to know and to refresh and those kinds of things, because if you're doing that constantly on a device that isn't always on, it just can really drain the battery. So we've spent a lot of time optimizing so that it feels fresh to the user and we have, if you go and search for something that could be years ago, that we can pull it quickly and retrieve it and trying to find the right balance between like how much of that data do we store and how much do we fetch.

[00:18:38] JM: What are some ways in which machine learning comes into play in designing Superhuman?

[00:18:46] ER: Yeah, it's a good question. I think it's something that we're going to lean more in the future, because I think where it will really become a major player for us is when we think about things like writing assistance, right? When we think about improving the speed that people can input, I think that's going to be a really large investment for us. Another area where I think it will be really important is helping people prioritize what's important. When you look at an inbox, we see people with thousands of emails. And one of the things we usually do is really help people get a handle on their inbox early on. But the reason people have thousands, hundreds and thousands, if not millions of emails is because that's how much data they're actually getting. And I think machine learning is really important in terms of surfacing the relevant important information for people.

[00:19:39] JM: And are you heavily involved with that work on surfacing the right information?

[00:19:45] ER: We're not spending a ton of time on it now. I think it's something that we're going to invest a lot more in the coming years.

[00:19:52] JM: Got it. Do you feel like there's more simple things that you need to focus on right now?

[00:19:57] ER: Definitely. I mean, there's so much we want to do in terms of driving our core experience forward. And as I mentioned sort of speed for us is a constant challenge. We also want to expand to more platforms. We've been primarily focused on iOS and desktop and Gmail. And for us building out android and O365 are some of our more immediate challenges.

[00:20:21] JM: Can you tell me a specific engineering challenge that you're focused on right now and how you're working through it?

[00:20:28] ER: One of the specific challenges we have right now is – So I mentioned O365. That's something we've been spending a lot of time on. And I think the biggest challenge there is there's complexity in the APIs. We are every day running into, I don't know if it's design decisions or bugs that make it – That are unexpected, right Things that are not behaving as documented or just don't seem in line with the overall design. And we're sort of teasing those out. So I think that's quite a challenge for us.

But kind of if I zoom out from the specific API, I think one of the challenges there is we have built a really robust application that is sort of in line with Gmail, right? It's a really, really deep extension of how Gmail models problems and that's very different from O365. And so a lot of the user interactions are just – Sort of paradigms are quite different. And so trying to figure out where we fork in our codebases, where we make those kinds of transitions I think is quite a challenge.

[00:21:40] JM: Tell me more about that. So where you fork, like where you try to differentiate from Gmail or from O365?

[00:21:49] ER: Yeah, and I think that's going to be – Yeah. And if you think about email providers, and I mentioned that we have these thick data layers. Well, that means we have this data on our client and we could decide to differentiate as the data is coming in, as it's being transformed, sort of even kind of allow the data to even be more integrated with the code. And I think that's one of the challenges. I'm trying to think of how to explain this without going into too much detail. But, yeah, one of the examples I'll give is the way that labels work versus folders, right? So in Gmail you have a concept of labels. Threads can have multiple labels and users can say, "Give this a GitHub label or give this a team label, whatever."

In O365, the emails are in folders, right? They can only be in one folder. So it's a really different concept. And we could decide – So in addition to the engineering challenge you have the product challenge, right? Like which behavior do you want to expose and then where in the clients do we differentiate between that particular behavior? So that's something we've been thinking a lot about as we're like uncovering the specific details of how the APIs work.

[00:22:59] JM: What is a mistake that you've made that has resulted in some technical debt that you're now having to pay down?

[00:23:06] ER: It's always hard to call something. When you look back, it's hard to call it a mistake, right? Because the decisions help you get where you are. One thing that I mentioned that we were making the transition from – We've just finished the transition from CoffeeScript to JavaScript. I think in some ways we could call adopting CoffeeScript a mistake, but it was an interesting time. Actually we had sort of two similar-ish decisions to make. On desktop we were deciding between pure JavaScript or CoffeeScript, and on mobile we're deciding between Objective C and Swift. There were sort of like pivot points for both platforms in terms of these languages, the versions being somewhat unstable. And so it's kind of interesting. On desktop, CoffeeScript was just newer and sort of in some ways I think that we definitely had to spend a bunch of time like reversing that decision. IOS we decided to go with Swift, which this was probably around Swift version 2.1. So we're on five now, and five has finally brought API stability. But in some ways I think we sort of saw opposite results, right? Like I think Swift was the right decision, CoffeeScript maybe not the right decision, but we chose both for sort of the same reasons. Like they were both modern, they were both – While they were unstable, we thought they would be the future and kind of had different results. So that's an example I'd say.

[00:24:25] JM: Interesting. Have there been any bugs that have resulted in really problematic user experiences like things that you've had to triage immediately or kind of outages that have been really scary? Have there been any kind of fire moments?

[00:24:45] ER: Yes, definitely. Let me think of what's a good example of a fire moment. Well, I'll say one of the things that we have spent a lot of time thinking about is send reliability. In our early days, our very first users, our sends were not super reliable and we have a PagerDuty rotation where when send fails to actually go through for any reason, whether it's – A common problem we see is people send and then they immediately shut their laptop. Or they send on their phone and they immediately back around the app. Well, in that case the send isn't going to go through. But in all cases we have done a lot to track send reliability and to try to systematically like improve different aspects of that of our send pipeline. And at this point, actually there's a lot of complexity in the send. We're doing things like, in addition to actually doing the send, we're like logging that we want to do the send and both have the payloads actually for the email. It's sort of like a failsafe. Like if the send doesn't go through, well, maybe the log about the send will go through. And in addition to allowing us to kind of track the progress of the send through our pipeline, it allows us to fall back, and in many cases make sure that there's reliability. So I think that's been an example. And I think about fires and some of those early days with much more common send failures. Every one of those was a fire, right? You have someone's really important mission critical work and it's not working properly. At the time we were sort of working, operating in a beta mode, but that's been something we spent just a ton of time on.

And it's kind of interesting because at this point our sends are super reliable and we still have our PagerDuty rotation. And the thing that we'll face more than anything else is dealing with outages and dependencies or sort of like unique scenarios where something we haven't seen before, which I think is really interesting. But the rate of failure has gone from maybe, I don't know. We used to track sort of one in 100 and now we're tracking like thousands and failures per thousands of send.

[00:26:47] JM: So what is it that makes the send event so difficult or so prone to outages or failures potentially?

[00:26:56] ER: Yeah. I don't know that it's any more – Well, I mean, one thing that can make it more prone to failures is, as I mentioned, people take the action of sending and then sort of forget about it, right? They might step away from their laptop. They might close the laptop, because the action of sending people in many ways think is the send. But often they'll be attachments which have to upload or the send itself will contain a lot of data. And so it takes time to actually send. I wouldn't say that it's necessarily more prone, but it's super important that it's reliable, right? So that's partly why we spent so much time on it.

We've built this kind of interesting mechanism on the phones where if you send an email and then back around your app, well, the app can get suspended, right? Much the same way as if you close your laptop, the app is not actually running. It can't actually send. On the phones we have this mechanism where we actually – We call it kind of like our ping act mechanism where we send notifications from our backend to try to wake up the phone so that it can actually send and put the app in an active state even if it's in the background. So it can actually finish the send. But while it's in that background, it's still running in a throttled mode, because on iOS if you're not in the foreground, the app is limited in the amount of data throughput that it can upload. And so even sending in that state just can take longer, which we spent a bunch of time kind of trying to improve the reliability around it.

[00:28:25] JM: So when an email is getting sent, do you get an act that the email has been received by the recipient provider? Or are you like throwing something into the dark?

[00:28:40] ER: In many ways – It's an interesting question. There're a lot of steps to sending, right? Sending goes through our backend. It eventually goes through Gmail to actually be sent out. And so depending on which step we may or may not get an act. In some ways you can say it's throwing something into the dark, but I think most of the lack of reliability comes through on the client side, right? So it's not like we sent it in. We don't know if it's sent. It's more like it never – We tell the backend we're going to send it and it never receives it.

[00:29:08] JM: So you are the head of engineering. We could talk a little bit about management. How has your management changed as the company has grown?

[00:29:19] ER: Yeah, it's an interesting question. Well, I think some things – In order to talk about what's changed, I'll talk about some of the things that have stayed the same. We've always been organized largely by technology. We've always had engineering leadership that's very hands-on in technology. And I don't think those are things – Those are philosophical choices. We have small teams, which I think is also a philosophical choice. But one of the things that's changed is sort of the amount of time. We're at a point now where we have engineers who spent a significant portion of their career at Superhuman. And when you're 10 people in a room, you're really focused on building out your product or proving product market fit, making your product super reliable. At this point we spend a lot more time thinking about the growth of each individual person on the team and trying to build the type of culture where people can spend their careers. So I think that's one of the things that's changed. We went from ad hoc goals for each individual around their growth to more formalized approach where people have growth goals. We have engineering levels and a ladder and definitions for what it means to progress in your career at Superhuman. So I think some of the formalization of what it means to be at Superhuman and what it means to be an engineer has changed. Certainly, the amount of time I spend collaborating with other people, coaching and mentorship, has gone up as opposed to those early days, which was much more focused on building out tech. Yeah, I think those are some of the things that have changed over time.

[00:31:00] JM: So one of the things I'm curious about is search. So on a client device, you've only got so much space and you want to be able to search over all the emails that are on the client as well as on the server. How does Superhuman handle search?

[00:31:18] ER: Yeah, I mean, it's exactly like you said. We're always searching both locally and against the remote store. One of the really interesting challenges is ordering, right? If you have results, you could have any set of results on the client. And so if you go into search and you type something, we could immediately show you the results that we have on the client that match your search terms. But that might mean when the remote results come in that they come in in any order. And so even what you're looking at could change and it just could be like a really like unstable feeling experience.

So we spent a lot of time thinking about like how do you – What's the best way to optimize towards showing people results quickly but also making sure that the results they see don't move around? And that's definitely one of the interesting challenges. I'm trying to think of what else we've dealt with. Another thing we deal with is just we're running a full text search using a SQLite database. And the way that the words are – The way that the search is interpreted can be a little bit different when you're talking to two different data stores. So that's another thing we spent a lot of time thinking about.

Yeah, and I guess another interesting thing is when you're searching, trying to optimize towards like when to show results. So I mentioned one challenge is just like do you show what you have immediately? Do you wait for the entire set to come in and sort of merge them? But the other interesting challenge is at what point in typing do you even attempt to search? So searching is somewhat expensive, but you have the network request. You're receiving the data, but you also have the results that come in have to be parsed. They have to be manipulated into a form that you can then show on the screen to the user, which is really expensive. And so if you really want the user to be able to search and type and interact as quickly as possible, you have to somewhat limit the amount of resources that the device is using to actually fetch the results. And so we spend a lot of time thinking about like when do you throttle fetching versus optimizing towards actually going getting the data? Yeah, I think those are some of the interesting differences.

One other thing that I thought was kind of interesting. On the desktop, when you search, we don't show you results until you like input the search, like you hit enter. Whereas on the phone, most people don't think of like having to go down and hit the search button, right? You have this sort of disconnect between your input field where you're typing in the search button, and the keyboard isn't super fun to use. And so we actually have a difference there, whereas on the desktop, you have to input on the phone. We try to predict when we think we have enough to search and then we go off and fetch the results.

[00:33:55] JM: So as far as the SQLite database for managing search, what kinds of updates do you need to make to that SQLite database over time? Is that like maintaining basically the search index?

[00:34:11] ER: Yeah. I mean, it's got all the data that's searchable. So in addition to the actual content of the email, that's the recipients. So it's not indexed in the sense that like it's not like we're – It is a SQLite database, right? It's like not in-memory or anything like that, but we have had to be pretty smart about what we store in the form that we store that so that we can retrieve the data associated with each – So you're searching at like the message level, right? Not the thread level. But you can also search the thread level. So we've had to be pretty smart about how we store that so that you can search in either of those forms. The challenge doesn't really come in in terms of – The database does get larger and larger over time, but not to the extent that the performance of the search is a major limitation. I think the bigger limitations are once we have the data, being able to render the results to the user. That's just much more expensive.

[00:35:06] JM: Are there other ways in which the client experience is kind of tests the resource availability of the device and in ways that you have to throttle the client application because it would otherwise take too much resources?

[00:35:23] ER: Definitely, I think that's what we spent a lot of time thinking about. Yeah, one example. I talked a little bit about this, but I think one example of where we have to throttle is, again, with phones, because it's not always on. If we sent a notification to the phone every time – So just kind of backing up a little bit. When we want to update, when a change comes in, we notify the phone and it's the kind of notification that doesn't necessarily wake the phone up, the app up. If it did, right? If it woke the app up every time a new email or a changed email came in, for our highest volume users, those would be like – That'd be millions of updates a day.

So one of the things we think a lot about is how do you balance freshness? Like keeping the app really fresh when you open it and making sure that it doesn't need to go and fetch a whole bunch of data. How do you balance that with not completely draining the user's battery, right? Because both lead to a pretty negative experience. So that's an example of where we have to intelligently notify the phone about updates and kind of intelligently decide when to fetch and also really optimize what refreshing looks like when you first open the device.

[00:36:37] JM: So I'd love to get now some perspective on the future of email and the future of Superhuman. How do you expect the app to operate differently in five years?

[00:36:51] ER: Five years. I love that time period. In five years I think we will – Input will look very different than it does today. I think we will lean into speech and other mechanisms of like inputting to either your phone or your desktop in part because the speed that you can type is a limitation to the speed that you can actually get work done. So looking for different ways of inputting information, I think it's going to be really interesting, and speech is one that comes to mind.

In five years I'd expect to wake up and instead of having to write drafts to every email that I need to respond to, I'd expect preformed drafts that I'm reviewing. I would want Superhuman to really guide me in what's important in my inbox. Now, we do some of that now and a lot of it is allowing the users to set up what – We have this feature we call split inboxes that allows users to divide their inbox into unique work streams. A lot of the way that we do that is the user can configure what those unique work streams look like. In five years I'd expect Superhuman to, based on my email, configure that for me and divide my inbox and sort of elevate the emails that I need to actually focus on.

I do think email is going to continue to be a really important part of the work day. I think there are other communication means. There're internal chat tools. I would expect to see a deeper integration across the different communication tools. And also I spent a lot of my day in email. I spent a lot of my day in calendar and I spent a lot of my day thinking about what I need to do.

And I'd like to see a deeper integration across all of those different mediums so that work is really a unified experience. Yeah, those are some of the things I think would be really cool.

[00:38:56] JM: And is there anything more general that you've learned about building and operating consumer application software from your time at Superhuman? Some general engineering lessons that you might be willing to share.

[00:39:11] ER: Yeah. One thing that I think is really interesting is at a lot of companies there's a lot of distance between engineering and the customer, right? Maybe engineers are seeing bug reports or feature requests, but it's really just a distant kind of communication. One of the things that we've tried to foster at Superhuman is something I call empathetic engineering, and it's really this idea of keeping the customer and engineers close. And some of the ways we do that are by having open design and product reviews. Our product requirement docs are always open for everyone to read and comment on and we always begin those documents with anecdotes from users, right? And as much as possible we like to do this in the words of the user so that when people are thinking about how to – When they're thinking about some problem they're solving or some delightful feature they're adding, they're actually thinking about it in the context of how this is going to improve someone's work.

We emphasize something that Rahul, our CEO, talks about a lot, which is how we make people feel is just as important as what we make. And while I think it's super important for engineering, it's interesting because it's just not often close, right? Like engineers are mostly thinking about what they make and not how people feel. And so one lesson I definitely have and sort of something that I think has worked quite well for us is really trying to foster that culture where you're creating engineers who are empathetic to the customer.

[00:40:49] JM: Cool. Well, that seems like a great place to close off. Thanks for coming on the show, and it's been a real pleasure talking to you about Superhuman.

[00:40:55] ER: Yeah, thanks so much for having me.

[END]