EPISODE 442

[INTRODUCTION]

[0:00:01.1] JM: Interviewing engineers is not a solved problem. Quite the opposite. Everyone in the software industry will tell you their own personal issues with the hiring process. One reason that technical interviews have not evolved significantly is the lack of standardized process and tooling. Some companies give you one phone screen, some give you two, some companies have you solve brain teasers, like how many golf balls fit in a school bus, and some make you fix bugs in their production code base.

During the onsite interview process, some companies use whiteboards and some let you use laptop. Software companies do so much. They should be outsourcing the things that are not their core competency. Certainly, they cannot outsource the entire hiring process, but they can outsource and standardize certain parts of it to a company like Interviewing.io.

Engineers come to Interviewing.io to practice their interview skills, where other engineers from top companies practice with them as an interviewer. When an engineer has practiced interviewing enough, they can use Interviewing.io to interview for real with real companies and find a job.

Aline Lerner is the CEO of Interviewing.io and she knows about the software interviewing and recruiting process as much as anyone. After working as an engineer herself, she started studying recruiting; consulting with top companies to help them improve their process. From her observations, she created Interviewing.io.

In this episode, we dissect the workflow that she created for engineers to improve at interviewing and find jobs. We also explore the insights that led her to starting Interviewing.io.

[SPONSOR MESSAGE]

[0:01:51.0] JM: Spring Framework gives developers an environment for building Cloud-native projects. On December 4th through 7th, SpringOne Platform is coming to San Francisco.

SpringOne Platform is a conference where developers congregate to explore the latest technologies in the spring ecosystem and beyond. Speakers at SpringOne Platform include Eric Brewer, who created the CAP theorem, Vaugn Vernon who writes extensively about Domain Driven Design, and many thought leaders in the Spring ecosystem.

SpringOne Platform is the premier conference for those who build, deploy and run Cloud-native software. Software Engineering Daily listeners can sign up with the discount code 'SE Daily 100' and receive a $100 off of a SpringOne platform conference pass, while also supporting Software Engineering Daily. I will also be at SpringOne reporting on developments in the Cloud-native ecosystem. I would love to see you there and have a discussion with you.

Join me December 4th through 7th at the SpringOne Platform conference and use discount code 'SE Daily 100' for a $100 off of your conference pass. That's S-E Daily 100, all one word for the promo code. Thanks to Pivotal for organizing SpringOne Platform and for sponsoring software engineering daily.

[INTERVIEW]

[0:03:20.8] JM: Aline Lerner is the CEO of Interviewing.io. Aline, welcome to Software Engineering Daily.

[0:03:26.9] AL: Thank you. It's a pleasure to be here.

[0:03:29.1] JM: How did technical recruiting evolve to where it is today?

[0:03:33.8] AL: Technical recruiting is a hairy beast with a lot of dark corners. I can talk more about the specific aspect of it, which is technical interviewing, then perhaps the industry as a whole, although I have some ideas about why recruiting is the way it is, and then I'll touch on that as well.

I think a lot of people have a lot of opinions about the right way to interview engineers. One thing that we have discovered in Interviewing.io after looking at thousands and thousands of

interviews on our platform is that nobody really knows what they're doing, us included. Although, we do have a lot of data and that gives us at least an idea of what not to do.

I was always curious about why technical interviewing is the way it is. It's an interview process that's different than you see in a lot of industries, especially older ones whether some established credential like the board exam for doctors, if you're board-certified and things like that. My best guess, and I did a little bit of research on this is that the way we interview now originated in the 50s at Shockley Semiconductor.

At that point, this industry was growing really, really fast because of the Cold War. They needed a lot of smart people to come work there and do stuff that nobody had ever really done before. It was very different than the kind of assembly line skills-based work that had preceded it. So rather than interviewing people for a specific set of skills at Shockley and the guy that was running Shockley himself, I think was pretty adamant about this approach.

He was much more into interviewing people for just their ability to solve puzzles. A lot of these puzzles ended up over time making it into the technical interviewing process at large companies in the 90s. Microsoft being one, and Microsoft in a lot of ways at the standard for a lot of how technical interviewing was, and still is.

These puzzles would be things like finding which coin is heavier than other coins using less weightings than you might think and stuff like that. At Microsoft, especially and then later at Google, this is one of the first times when you needed a workforce that wasn't just dependent on managers deciding exactly how to implement everything and then just having coders be drones.

Everything was moving too quickly, so you needed these individual contributor aspects of your workforce just to be able to make hard decisions and think on their feet and potentially do a lot of design and architecture.

Programming languages themselves are changing so quickly that potentially interviewing for a set of skills would be too limiting. All these companies are trying to just find ways of figuring out who is a good problem solver and who can think like an engineer. They ended up with a lot of

these brain-teaser problems. Later when Google figured out that brain-teasers don't predict much of anything, more algorithmic kinds of problems.

The last thing I'll say about it is that one of the reasons these large companies still persist in using this kind of interviewing, whether it's good or bad is that as you scale and you scale your hiring process, you need to have this notion of interchangeable parts. You need to plug in any interviewer into any interview and have them still be able to do it, which means that your interviews have to be very general.

This way, there is a minimum ramp-up in onboarding new interviewers, and you can just swap them in and out and really scale your hiring process. What ends up happening now back – we're in present day, we have this process that evolved semi-organically for a variety of reasons that ship the needs of the day.

What ends up happening is a lot of smaller companies are looking to these giants; the Microsofts, the Googles, more recently the Facebooks and looking at their process and copying that process blindly almost in a cargo-culty kind of fashion, where you're looking at the process when the process is really not the reason for why things work.

Rather, people want to work at Google, because Google does cool stuff and as smart people, it's almost like despite their hiring process, no matter what they do, they'll probably still going to have a revolving door. That is not true of some small startup that no one has ever heard of. One of the things I'll hark in to over and over again is that if you're a small company and you don't have much of a brand, it doesn't mean you should lower your bar, but you should think critically about whether you're making people jump through hoops.

[0:07:40.1] JM: The smaller startups then, should they have a more unique and differentiated hiring process, would it make sense? Because it sounds like the repeatable, highly data-driven, interchangeable parts driven strategy of a Google, maybe that's necessary to have a scalable hiring process. But if you're just a super small company, you might want a more tailor-made hiring process. Is that what you're saying here?

[0:08:13.4] AL: That's exactly right. One of the things that I think is really important about having a tailor-made process isn't even in necessarily identifying the best talent for you, although that's important, but in just cognizant of market dynamics when you're hiring a software engineers. The fact is that at least for the time being, it is really hard to hire good engineers and the supply of these engineers is completely outstripped by demand.

That means that the economic power is in the hands of labor or the engineers themselves. That means that if you're going to interview people, you can't make them jump through hoops. Rather, especially if you're a small company without much upper end, you should be using your interview as an opportunity to sell at every step of the way.

How does that actually translate into your questions? Well, a lazy process might just involve opening, cracking the coding interview, picking a random question off that page and then starting to ask people that question. That works if you're Google, because people are already sold on your brand. If no one's heard of you, what you might want to do is tailor your questions to be more indicative of the kind of work that you do every day.

That doesn't mean even necessarily that they have to be super practical. You can still ask a lot of the same – if it's important to you, you can still ask a lot of the same algorithmic kinds of questions while putting it in a framework of, "Hey, we recently ran into this problem. Let's think about it together." Then if you're a good interviewer, you can progressively lay your complexities.

You can start with a pretty easy version of that problem, and then by the end after four or five iterations you will probably have gotten to the point where you don't even know how to solve it yourself anymore. One of the best interviewers I've ever met once used this expression, or like the point of an interview is to see if we can be smart together. I really, really like that.

If you do this well, you can pick an interesting problem and then not only does the candidate appreciate that they got some insight into the work if the problem is compelling enough. It's something that's probably going to stick in your head if you're the candidate. Then when you leave the interview, you're still going to be thinking like, "Oh, is there a better way? Is there a

good way I can solve this?" Then you've really gotten under their skin, which is the point of a good sales approach. That's something I would very strongly advocate.

One of the things I recommend when companies ask me about this is just start a shared Google Doc or something else. Just any shared document with your engg team. Every time one of the members of your team or you does something cool or something unexpected, or runs into a problem that makes you think, just dump it in there in an unstructured way. Pretty soon, you'll have a nice repository of the kernels of good interview questions, and then you can take these ideas and massage them into something that you can repeat.

[0:10:58.4] JM: Before we dive into your solution, maybe you could touch on some of the other problems of –

[0:11:04.8] AL: With recruiting?

[0:11:07.0] JM: Maybe you could dive into some of the other problems of interviewing and the current model, and then we can just talk about how Interviewing.io solved some of those problems.

[0:11:17.4] AL: Yeah, happy to. So yeah, I'll talk a little bit about some of the other interviewing problems and then kind of as a problem as to the beginning, I'll touch on little bit on what I think is wrong with recruiting in general, so then I can really setup how we solve everything. No, I'm just kidding. We don't solve everything. But this like a setup that the problem is real.

One of the main other problems with interviewing, and this is kind of a by-product of the kinds of questions people ask is that it's pretty non-deterministic. What does that mean? It means that if the same person does a string of interviews over a pretty short span of time, they're probably not always going to end up with the same results. So they're not always going to pass.

This is something that I think people just really suspect, especially if they've been through a few interviews themselves. But this is something we have actually collected data on at Interviewing.io. The data is exactly as I described, so you look at a person who participates in a series of technical interviews over a fairly short span of time and then you see how they do.

The fact is that most people, even if on average they're killing it, like doing really, really well. They're going to have an interview that they bomb agree 1 and 5, every 1 and 10. While it's not that often, most people are not that consistent, so people will mess up 1 and 4, 1 and 3. These are still very, very good engineers. Many of whom are getting offers from top companies, but what ends up happening is not only is it a poor signal, which means that interviewers waste more time and companies spend more time paying time and spend lunch time on interviewing.

You can have a candidate who ends up getting rejected by Lyft, and then getting an offer from Uber or getting an offer from – or vice versa, right? Then whichever one of those you end up at, and I'm just using those as an example of companies with a fairly comparable bar who are operating in a very similar space.

No matter what, after a few months, a recruiter from the company where you got rejected is probably going to be hitting you up again. So the whole system just feels very clunky and inefficient. The other piece of this outside of interviewing is that the resume, and this is even before interviewing. So like how do you even get your foot in the door at a company? Well, typically you have a friend refer you, or maybe a recruiter reaches out to you on LinkedIn. If not, you're going to apply online.

But in the latter two cases, recruiters are typically just looking at how you look on paper. The resume is just such a fundamentally flawed and well signaled document, especially for engineering. As a result, some of the best people are being kept out. Some people that look very good on paper, but actually can't code their way out of a paper bag are being let in.

You end up with so much noise. And as a result, by the time people get to the technical interview, most of them fail in part, because the process is flawed in a large part, because the filtering that happens even before you get to the interview is completely flawed.

Then we end up spending all this money in time interviewing the wrong people. Every company in the valley seems to be chasing the same 10 candidates. In the meantime, there is this long tail of amazing and perfectly qualified engineers that don't even get to show what they can do.

[0:14:17.3] JM: All right. Well, explain what Interviewing.io is.

[0:14:20.0] AL: Yay. Okay. So having set up the problem, we just want to get rid of resumes in technical hiring entirely, or at least ditch them as a means of deciding who gets their foot in the door. What we've created instead is a platform where any software engineer can come and practice technical interviewing.

This ends up being pretty appealing, because technical interviewing is scary. It's scary to everybody and almost counter-intuitively it's especially scary in our experience to senior engineers. Many of whom have and had to deal with the kinds of algorithmic questions that interviews features since they were an undergrad versus since the last time they had to study.

Before we existed, people would probably practice with their friends or more likely they would setup a few burner interviews. So pick a few companies where you don't really want to work. But that have a pretty indicative hiring process and you will just go there and interview with them warm-up, and then go to the companies where you actually want to work. Like dating, where you go on a few throw-away dates if you're getting back out there to find your sea legs before you actually go and date the people you want to date. So same idea.

We decided we're going to make that easier. We give out free practice interviews to anybody who wants them, as long as they're a software engineer. The really cool part of it is that everything on our platform is completely unanimous.

A lot of our users are engineers that do work at companies like Facebook or Google and have been there, for I don't know, like four years and they're maybe thinking they're a little bored and maybe they want to get out there and then try a startup, but they realize that they have to go through the interview pamphlet if they're going to do that.

If you're an engineer with that seniority, with that much brand sparkle behind you, it's so intimidating to have to get out there and represent one of these big friends. Because if you fail, you really look like an idiot, right? Everyone is expecting you to kill it. Then if you're a Google engineer who can't reverse a link list or whatever, you really look stupid.

We de-risk the whole thing, so you can show up, get free anonymous interview practice and you're not solving a-synchronized challenges. This is all going to be with another person. A lot of these are interviewers that are in our line-up that are coming from top companies and are willing to give their time. The cool part is once you've done a few of these interviews, we essentially create a profile for you and figure out where you stack up in the grand scheme of things.

We also tell you your percentile, which is nice, because a lot of the time you have no idea. Everybody thinks they're doing poorly all the time and some of them are, but most of them aren't. Once you're established as a top performer in our system, you can go through and for any of the companies that hire through us, any of the employers on our platform, you can just click a button and instantly book an anonymous technical interview with them effectively bypassing the entire resume screen phase, all the scheduling back and forth that you'd have to do and you can skip right to the part that matters where you can get to talk to an engineer at that company.

What ends up happening is people get in the door based on their ability. Of course, it's not perfect. Technical interviews are flawed, but at least we have aggregate data and not just one data point. Beyond that, one of the most exciting things that we keep hearing over and over from our customers, and these are companies like Cora and Lyft, which Facebook and so on is that a lot of people that applied there ended up getting rejected based on their resume. They didn't even get to interview.

Then they come in through Interviewing.io, sort of as coming in through the back door and do really, really well in interviews and get hired. For us, that's such a win and it just makes us so happy to hear. Because I think everybody knows the current system is completely fucked, and anything that we can do to make it just a little more fair makes us really, really happy.

[0:18:11.1] JM: Zoom in a little bit more on your process, how process is different from the conventional way that a candidate might be screened and potentially hired at an organization.

[0:18:23.4] AL: Conventionally, your resume is going to matter. Unless you have the good fortune of coming in as a referral, if you have a friend that refers you your resume probably doesn't matter as much. But if you're not well-networked, then you didn't have the good fortune

of maybe going to one of the few schools, you're probably not going to have connections in the heart of Silicon Valley, right? I said that with like air quotes, but you can't tell that.

Anyway, if you aren't well-networked, then your resume is essentially your identity. If you don't have a top school or a top company on that resume, the doors of top companies will likely forever be closed to you. For us, this is a way of opening up those doors to people who code, as well as somebody that did go to one of these top schools or top companies as expected to code, but doesn't have that pedigree.

Even beyond that, we want to make the process just more efficient for everybody. Even if you are a well-pedigreed candidate and in track, I think about 60% of our users are, we still want to make the process less painful for you. This way you don't have to bother all your friends, right? You don't have to find that old recruiter e-mail or LinkedIn in-mail if you even ever open LinkedIn, and hope that that recruiter still work there, and you don't have to send your resume.

Even if you're a good candidate, a lot of the time you won't hear back, because applying inbound in this market is like screaming into a black hole, when the abyss doesn't often look back at you. It's this attempt to completely take traditional credentialing off the table. Make how well people write code the only thing that matters. Then on top of that, just completely streamline the process for all engineers regardless of whether they're pedigreed or not. As long as they can code, we want them.

[SPONSOR MESSAGE]

[0:20:12.0] Simplify continuous delivery with GoCD, the on-premise, open-source, continuous delivery tool by ThoughtWorks. With GoCD, you can easily model complex deployment workflows using pipelines and visualize them end to end with the value stream map. You get complete visibility into and control over your company's deployments.

At gocd.org/sedaily find out how to bring continuous delivery to your teams. Say goodbye to deployment panic and hello to consistent predictable deliveries. Visit gocd.org/sedaily to learn more about GoCD. Commercial support and enterprise add-ons, including disaster recovery are available.

Thanks to GoCD for being a continued sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:21:11.4] JM: Talk a little bit more about the anonymity, because this is one of the technical challenges that you've overcome is implementing a way to interview anonymously.

[0:21:21.3] AL: Yeah. This is something we're super proud of. Thank you for asking about it. There are a few aspects as we learned that are needed in order to make an interview truly anonymous. Let me tell you how our interviews work and then I'll tell you a little bit about the stuff that we built. Because I didn't have a hand in building most of them beyond wheeling them into existence, some of these explanations will be a bit hand-wavy, but hopefully you'll forgive me.

The way it works is every interview on our platform, whether it's practice and if you make it into the top performer pool and talk to real companies, whether it's real, it's completely anonymous. The way it works is when you actually show up for your interview and all scheduling on our platform happens automatically as well, and obviously you don't see who you're scheduling with, except the company name.

Once you show up to your interview, there is no video. You just get dropped into a collaborative coding environment with audio and text chat and a white board and a place to take notes. Essentially, everything that we think you will need to do a technical interview on both sides.

For the collaborative coding environment, we're actually fortunate enough to be able to license code our pad, which I'm sure many of your listeners have used either as an interviewee or an interviewer. In our mind, it's the best interview tool on the market and we're very happy to have them in our product.

Unlike a typical technical phone interview that uses code or pad in order to preserve anonymity, we can't just have both parties using Skype or using phone or something like that. So we ended up building a bunch of stuff around coder pads. So we have a VoiP connection, so VoiP

probably just starts automatically once you enter the interview. If you're in a place with bad Wi-Fi, we'll just give you a conference room that we spin up automatically in that conference room.

He is also anonymous. Both parties get a pen. They just go in there and we'll stitch everything together, whether you are on phone or on VoiP. The other cool thing that we do with audio, and this is one of the things I'm most excited about is that we have a capability of actually making gender completely anonymized in real-time.

So we can make women sound like men and men sound like women, or make both parties, both gender sound completely heterogeneous in this uncanny valley creepy kind of way. So we have complete control over that. That actually happens in real-time on the client, and this is something we have a patent pending on. There is just a bunch of math happening on that sound wave, before we even send it up to the server.

[0:23:48.4] JM: What's the importance of doing that? What kinds of biases are you removing from the process?

[0:23:54.0] AL: Yeah. Well, I think one of the biases that gets the most air time is gender bias. Whether there is a bias against women in tech, that one is obvious. I don't know the answer to that. I can talk a bit about some of the data that we've uncovered in looking at our own interviews and comparing ones where gender was off the table to ones where it wasn't. I'll come back to that.

The other biases that maybe are a little less popular and get a little bit less journalistic time are ones that have to do with pedigree. To me, those are definitely a real problem. Like gender bias. That's potentially a thing, and it's something that is a serious thing if its. But one thing that we know for sure is happening is that companies are very, very much biased against people who did not attend a certain school or work previously at one of a few companies.

[0:24:47.6] JM: Even today? That's even today?

[0:24:49.3] AL: Absolutely. This is something that is a huge problem. I think that this has become a louder conversation, and out of economic necessity companies are starting to come

around, because if they continue to chase after the same 10 MIT grads in the valley, they're not going to have a workforce. I went to MIT disparage. MIT is a great school, but there are plenty of great engineers that I've worked with and did not go to school at all, and plenty of people I went to school with that can't code. It goes both ways.

Definitely, when I first starting Interviewing.io and this is just a few years ago, I was still working as a recruiter and one of the things I was doing as a recruiter was because I come from a technical background that I use for my code before doing for about five years. I was in a position where I could interview my own candidates.

I always felt like if I'm going to endorse a candidate and say, "They're good." I want to make sure. I run them through some technical questions, so I could feel good about it. Then I present these candidates to some of the companies I was working with. They would say, "No." I'm like, "Well, what do you mean no? I know this person can code." They're like, "No, it doesn't matter. We have a hiring spec. Essentially, we are looking for people from these schools and these companies."

There is one startup that I actually with that I won't name, but actually gave me a flowchart to make my life easier. They're like, "You're a recruiter. You work with us. Here is a flowchart. Did they go to this school? No, they didn't. Okay, then do not pass – do not collect the $100. Fuck you, we're not taking this candidate."

This is something that really pissed me off as you can imagine. One of the companies I work with actually issued me this challenge and they ended up being one of my favorite companies to work with, and I still work with them in the Interviewing.io capacity.

Today they said, "Look, you have a bunch of people that look really weird on paper. We're going to give this a shot. So no matter who you send us, if you feel good about them, we'll talk to the first five. Then by that point, if –" I forget exactly what the terms were, but it was like, "If at least two of them don't get an offer, or at least one of them doesn't get hired, whatever it was, then we're never working with you again." I was like, "All right, guys. Challenge accepted. Let's do this.

That ended up working out so well. I have such tremendous respect for this particular company. Whereas a lot of the other ones that I work with, and I was working with some of the top startups in the valley. They were just saying, “Nope. Not going to engage.” I think that I understand why recruiters have a job to do and they’ve been trained in looking at a certain set of proxies, because those proxies do carry some signal. They’re not entirely worthless. Many recruiters don’t have the domain expertise to do anything else. They’re just trying to do the best they can.

One of the only ways that I think to break out of this pattern, and this is still something that’s pervasive today. This is why Interviewing.io insists that every interview be anonymous even when they’re real is because companies still do put so much weight into people’s backgrounds. One of the only ways to fight this is with data, and that’s why we set up Interviewing.io the way we did is so we can collect so much data and so much convincing data about people’s performance that companies then can’t be in a position to say no.

In fact, it’s working. Most of our candidates end up making it through company’s processes successful, because how well you interview is actually a pretty good predictor in aggregate of how well you’re going to interview. Not rocket science. The data collection is the hard part, not the consent.

[0:28:13.1] JM: Yeah. You have a pretty interesting background. This background of going from engineering to studying, recruiting and detail. I remember, I was an engineer, I was working and I would see your blog posts and your articles, because you were just writing about recruiting for a while. It seems like you were really just trying to explore the space and figure it out. I remember your blog post going to the top of Hacker News and people were clearly reading what you had to say.

I was reading it too, because it was intriguing. You were finding contradictions and assumptions in the industry. Obviously, it led you to starting a business. I’m a little bit curious about that transitionary period, because I’ll tell you as an engineer like you maybe, I mean – I like coding.

[0:29:04.7] AL: Probably a coder one. I never really liked coding. I was like, “I would get the hell out of this industry.”

[0:29:10.3] JM: But you must have liked – I mean, you must have liked it enough to at least, and I think you – you got a computer science degree, right?

[0:29:17.1] AL: Actually, it's in brain and cognitive science. I studied comsci. I actually started coding when I was very young.

[0:29:23.9] JM: Right. You like it enough. Asking this is kind of taking a side path, because my personal experience, I was engineer for a while and I enjoyed it, but I wanted to try something that was tangentially related. That's how I went into software engineering podcasting. It's random, but you took a – I would say a similar path into studying recruiting of software engineers, which is a strange and niche route and probably surprised some people when you started delving into it.

Maybe for people who are listening, who are software engineers and maybe they're fairly new to software engineering, maybe they graduated in the last couple of years, or they just went to a coding boot camp and they're in their first job and they're starting to have this sensation where, "You know, maybe engineering is not for me. Maybe I shouldn't be a software engineer. Maybe I can explore something laterally to software engineering." Maybe you have some words of advice or just some anecdotes for your own career exploration?

[0:30:24.2] AL: Yeah. Well, I'm happy to talk about myself. One of my favorite subjects. I guess, the best piece of advice that I can give is that at least for me, nothing felt deliberate. My career path is one of the most tangled and windy ones that I have heard about, at least in our industry. In our industry, many people are very goal-oriented and they just do the thing they're supposed to do and then they succeed. That's not what happened to me.

First off, the reason I ended up switching away from computer science as a major is because I flunked. At MIT back in the day, they don't do this anymore, but back in my day there was this class that you had to take called circuits and electronics or something like that. It was just a class where you just messed with a bread board all semester and did stuff with circuits and circuit design.

You had to have this class for a computer science degree. It was one of the few electrical engineering classes you had to take. I actually flunked it. I flunked it not once, but twice. I actually enjoyed programming quite a bit back then. But for me, it was always a means to an end. I never found a lot of elegance in it. I'm a very practical person, and it was a nice way to get shit done and it was a nice way to automate away tasks that I maybe didn't want to spend a lot of time on. I'm very inpatient as a person, and very instant gratification and programming helped with that.

At that point I realized that I probably wasn't going to get a CS degree on this. I wanted to take this class for a third time and flunk it probably, which I didn't. I got this random degree in brain and cognitive sciences, which ended up feeding into a lot of the blogging that I did, because that's where I first started a lot of interesting papers about human behavior and learning a lot of statistical techniques.

These things ended up serving me much better in a lot of ways than some of the programming assignments that I did, although I had no way of knowing that at the time. Then after I graduated, I ended up cooking professionally for three years, which was one of the most intense periods of my life, and one where I got to meet people that I never would have met otherwise.

I'm really grateful that I did it, one because I have some crazy stories, but that's not the main reason. The main reason is that that was the first time that I really got to see a different hiring process as in aside, like when you get a job as a cook, you don't really talk about your resume or your experience or your hopes and dreams or your five-year plan or whatever. You just show up and you bring your knives and then that's what you do. You just start doing the work.

You're at the restaurant, in the morning you're prepping for the station where you're going to be working. Then the evening, you're putting out dishes that the station that you've been assigned to is responsible for. The whole time, someone is watching you. At the end of the night if you did a good job, then you get a job offer and they feed you. If you didn't do a good job, maybe they feed you then they send you home.

To me, that was just eye-opening, because I had always thought that engineering was supposed to be something that was super meritocratic. Then I realized that the way that engineers are

hired is just not meritocratic when compared to this other industry that's much older. That stuck with me and informed a lot of the ways in which I thought about hiring, although again, I had no idea that that was going to be in my brain or come up later in my life.

The reason I got into recruiting too was also very unplanned. I was working at this tiny company where nobody was doing hiring. The whole company was 20 people and the engg team was about five. All of us, the whole engg team was taking on recruiting in this distributed fashion, so we were all looking at resumes and we were all scheduling phone screens and we were all doing interviews. It was exhausting.

Finally I thought, "Whatever. I'll take one for the team, and then we can all get the hell back to work." Of course, it ended up being a much bigger job than all of that and I found myself rather enjoying it, but it also wasn't on purpose. When I got into it, I just realized how messed up this whole industry was and how third party recruiters were making a lot of money and doing all the wrong things.

There are all these generally bad practices and inefficient means of hiring people that perpetuated a lot of discrimination in this industry, not maybe in a traditional gender or race sense, although that might be an issue as well, but just against people that didn't meet some very arbitrary seeming set of process.

To answer your question in a very long-winded way, what I would encourage people to do is to try stuff, and to not worry if it's the right thing or not the right thing. I've had some conversations with especially younger ones where they talked about how maybe they don't necessarily want to program and there are plenty of women that do. The ones that I've spoken to sometimes don't, but they feel like they're abandoning the mantle and failing women everywhere, because they're leaving computer science to do something.

It's like, "No, you should do what makes you happy." A lot of the time you don't know what that is. The best thing you can do is try stuff. But when you try it, like it try it in a way where you're actually immersing yourself and where you're looking around you and you're noticing things and you're trying to understand why things are the way they are, and noticing maybe inefficiencies,

or just noticing patterns, or really just doing anything that takes you out of the way you usually think and challenges your perspective.

I guarantee, if you're a thoughtful person and you have any kind of motivation at all, if you just acquire some of these data, your brain is going to turn it into something amazing and you'll figure out what you want to do, if you're fortunate enough to actually have that choice. Of course, some of us don't have that choice and we have to do a certain work – line of work, because that's what pays the bills. For me, that was software engineering for a number of years. I was fortunate enough to kind of be able to marry that with something that I'm interested in.

[SPONSOR MESSAGE]

[0:36:09.1] JM: Dice.com will help you accelerate your tech career. Whether you're actively looking for a job or need insights to grow in your role, Dice has the resources that you need. Dice's mobile app is the easiest and fastest way to get ahead. Search thousands of jobs from top companies. Discover your market value based on your unique skill set. Uncover new opportunities with Dice's new career pathing tool, which can give you insights about the best types of roles to transition to.

Dice will even suggest the new skills that you'll need to make the move. Manage your tech career and download the Dice career's app on Android or iOS today. To check out the Dice website and support Software Engineering Daily, go to dice.com/sedaily.

Thanks to Dice for being a continued sponsor of Software Engineering Daily. If you want to support the show and check out some new career opportunities, go to D-I-C-E.com/sedaily.

[INTERVIEW CONTINUED]

[0:37:28.0] JM: Indeed. Well, I mean I could share your – echo your advice. I think people who – if you work hard and you spend some time developing skills in software engineering, it gives you a lot of license to go and explore other things. You can always come back to coding. It's a pretty good safety net, at least for now, especially in your younger years it's such a good – can be such a good strategy to go and explore a random walk.

[0:37:56.7] AL: Definitely.

[0:37:57.5] JM: Let's get back to Interview.io. You're a source of very interesting data, and you're looking closely at that data in a way that other people are not. Give a little bit of a taste of what you're looking at right now. People who want to see some very interesting studies for example on the correlation between what programming language you choose in your interview and how the interviewers might see that.

For example, if you choose PHP for your programming interview, you might be perceived negatively at certain companies. Maybe at other companies, you might be perceived positively perhaps Facebook or Wordpress, but those would be outliers.

I mean, I've read a number of these blog posts about the data that you're studying. Tell me what you're looking at lately. What kinds of interesting stuff is coming out that you haven't had a chance to write about yet perhaps?

[0:38:47.9] AL: You want the new stuff.

[0:38:49.2] JM: The new stuff.

[0:38:51.3] AL: All right. I'll see what I can do.

[0:38:53.3] JM: Give me the scoop.

[0:38:54.7] AL: Just for background, we have a lot of interview data and I have alluded to it a few times in this interview, but we collect basically everything that happens during a technical interview. All the code people write, whatever happens when they hit run, whether it compiles or what the output is. Also, every text chat, everything that people draw, and of course the audio track for both the interviewer and the interviewee. Did I mention that already? I might have mentioned that.

But in any case, the interesting thing is once you get a lot of these interviews, you can start to actually draw some interesting results from those. Of course, one of the pieces of data that makes that possible is what happens after each interview, which is how people do.

We look at everything from whether gender actually matters to how people do from interview to interview, to whether a top school or other things might be more or less predictive of your interview performance. A lot of these findings have been surprising. In large practice, I think nobody really knows anything about interviewing and we're all just learning.

One of the things that's coming up that we haven't published yet, but that I have started digging around a little bit into is what makes somebody a good interviewer. There are some very, very repeatable patterns that we've noticed and repeatable behaviors and markers among interviewers that are rated highly on our platform, versus interviewers that are maybe not rated so highly.

I won't give away all the stuff. I'll leave some surprises, but one of the things that we found over and over that seems to matter, and this should not be a surprise to anybody that's ever done interviews is engagement.

People spend a lot of time trying to find the perfect question and trying to calibrate in doing all of these things, but the fact is you shouldn't be reading Reddit or Facebook when you're interviewing somebody, because they can tell. That is probably the worst mistake that interviewers make. That is of course, again one of the more obvious things. But I'll leave the salacious, less obvious details to the post itself if you forgive me.

[0:41:03.7] JM: All right. I'm closing Reddit right now.

[0:41:04.4] AL: Well that's fair. When I talk, I expect everyone's reading Reddit all the time.

[0:41:11.5] JM: Tell me about some of the companies that are using Interviewing.io. Explain how they use it. Maybe you could just refresh people who may have – didn't quite understand what your platform does. Give an example, like end-to-end why this is useful for the companies who are on Interviewing.io.

[0:41:32.9] AL: Yeah. Value prep for engineers is pretty clear. You get free practice and then you get direct access to companies without having to jump through hoops. For companies, we think it's the fastest, most efficient and in many cases cheapest way to hire senior, and more recently interns and new grad software engineers.

Think about how a company typically runs their hiring process. They have recruiters going out and sourcing candidate. That takes a lot of time, right? In our experience, it takes about 10 hours to source one candidate that actually gets into your process. Sourcers are not actually very cheap. They used to be. Not anymore. Good sourcers are expensive, almost as expensive as engineers themselves.

[0:42:14.3] JM: Sorcerers?

[0:42:15.9] AL: Sourcers. So people whose job it is to send you the spam that says, "Hey, you should work at our startup."

[0:42:23.1] JM: Sourcers. Okay, got it.

[0:42:27.6] AL: I think some of them do – what they do does border on witchcraft for some of the good ones. But yeah, so sourcers are people that go out and throw a spaghetti and a bunch of balls. Some of them do it very well, some less so, but it is their job to get people in at the top of the funnel at the beginning of the process.

At that point, you have a candidate talk to a recruiter, and the recruiter sells them on the company and also evaluates them on some of the more intangible aspects of the candidate's profile. Can they communicate, or are they a good culture fit, whatever that means. Then and only then after a candidate gets through that do they actually get to talk to an engineer.

By now, you've spent something like 60 hours of – I think it's 15 hours a candidate or something to get to that one technical phone screen. What happens now is you do the screen, and if you have a good funnel, maybe one in four people will actually make it through, because up until now you have no indication that this person can code at all.

By the time you get to a single onsite, you've invested so much time and so much money. Most companies track some metrics that don't necessarily track how much time it takes to get to an onsite and to a hire, but it's a lot. In our experience, it can take months to hire a good senior engineer, in large part because the process is so noisy.

We're trying to short-circuit all of that. We think that instead of going out and trying to get a bunch of candidates to come to you in this market where most of them aren't going to respond and where you actually have no idea if they're any good, what if we can just bring really qualified engineers to you?

The reason our company has I think worked with us is because with Interviewing.io you can get with a local candidate from first interview to offer in as little as a week. This is something that makes us very proud, and the reason we can do this is because we have such a good idea of who our candidates are and whether they can code.

Of course, lack of technical ability is the main reason that companies reject candidates. Unfortunately, even getting to that point in their process is pretty buried, because there has to be a lot of stuff that happens before.

If you're a company on our platform, especially if we're deeply embedded in your process, you can tell us how many senior engineers you want to talk to every month, and we'll just have them show up, and you can interview them on our platform at pre-appointed times. We even have a mechanism where scheduling happens completely automatically.

So everything just lands on everybody's calendar, mindful of their availability. Companies are regularly making hires one in every six interviews, something like that and making offers much more often. Lyft and Twitch are two of our biggest subscription customers where they're just doing this at a regular clip every month. Now instead of having to file these disparate channels to get senior engineers into their pipe, they're just coming to them and these are already people that have been pre-vetted.

It's just a really, really nice shot in our – for any company's hiring process. We do specialize primarily in backend and full stack engineers, so we're not yet at a point where we can do neutrals like machine learning, frontend we'll probably do soon, but not yet. But if you want solid generalist engineers, I can confidently say that we're probably the best way to do it.

[0:45:42.7] JM: All right. Pretty good business. What are the marketplace problem? Because this is a two-sided marketplace. Every two-sided marketplace has its frictions, it's got surpluses or it's got scarcity. What are the issues that you have around two-sided marketplaces and how do you solve them?

[0:46:01.6] AL: Yeah, it's a great question and it's a real challenge. I think that one of the biggest problems that marketplaces, especially two-sided marketplace companies encounters is the notion of chicken and egg, right? You have to have one side set before you can attract the other side and where do you start?

With that, we got a little bit lucky because I was able to get our first influx of users through Hacker News, which has treated me so well and has been such an amazing channel for Interviewing.io and for me before that. We ended up just putting up a marketing site saying, "You can get free anonymous technical interview practice with engineers from top companies."

We ended up getting, I think 7,000 sign-ups in the first 36 hours and we were number one on HN for two days. That was unreal. At that point, I realized that I had shut down my stupid recruiting business and do this instead. But that certainly helped in this market if you have the candidates, most of the time that the companies will come.

But there are other challenges as well, making sure that your side has balanced one another. You never want to have one side completely outstrip the other. So how do you grow them at the same rate, even though growing them takes very, very different kinds of effort? Getting companies on the platform is a very different experience than getting candidates.

That's generally been one of our big challenges is making sure that we're growing steadily on the candidate side, but then that we can keep up with companies. Of course, the more good companies you have on the platform, the more of a draw it is for candidates. But we've been

very mindful of our metrics, so at this point we know exactly how many interviews it takes to make a hire, and we know how many candidates we need coming in every month in order to get to a certain number of monthly interviews and we know roughly how many people are going to be doing well in practice.

Getting to that point where it's like a well-oiled machine, and I don't know if we're well-oiled. We're like at a machine. We're working on oiling it up, but getting to a point where those things definitely takes a lot of time, and it's something that we had to figure out in a very real way when we were raising money fairly recently. Because when you raise money, people want to know these numbers. If you don't know them, you look like an idiot.

[0:48:25.4] JM: The oil is oftentimes customer satisfaction type of people that help guide the candidates or the companies through the process. Have you had to put in a lot of work towards scaling that aspect of your business?

[0:48:41.2] AL: Yeah. This is a great question. It's one of my favorite to answer. There are a lot of very good players in this space that help companies hire. It's a crowded space, and there is a lot of money in it so it's not surprising that people want to do it.

One of the things that makes this business really interesting to me is that we're trying really hard not to operate like a recruiting agency, but rather to operate like a true software platform. That means that unlike many other companies in the space, we're not trying to hire more staff in order to grow. Or we're going to need to hire engineers of course, and product people, and some amount of account management and customer support, but we are not looking to hire recruiters.

We are not looking to be a recruiting agency with a clever gimmick and a nice skin that makes it seem like a software product. We actually want to be that real software product. For instance, all our scheduling happens automatically. We have a series of reminder e-mails on the platform that take the place of a person nudging you.

We really try to make it candidate-driven. Candidates are in complete control of their entire job search the whole time and they can just control everything they need to. For us, the biggest and

most important indirection that happens during a job search is a candidate talking to a peer or a hiring manager to company and having this really organic high signal conversation about the actual work, and about the company, and the roadmap, and all of these things that you need to talk to a domain expert in order to uncover.

We just want to have as many of – we want to take everything else away and put that center stage and then have as many of those interactions as possible. We're not looking to hire recruiters or talent managers, or talent advocates, or any of these other things. We want this software platform to do the work – done glamorous work and then have the real and interesting stuff be that conversation between the candidate and the company.

That said, we are a – any software product needs at least one customer support person, and we recently hired one. She's amazing. Her previous background, doing interrogations in the US Army, I think. Lends herself particularly well to doing customer support.

[0:50:55.2] JM: What are the hardest engineering problems that Interviewing.io is faced with today?

[0:51:00.1] AL: Yeah. Well, probably the most "real piece" of engineering that we've done has been our voice modulation. If you go in our blog, you can hear a demo of me sounding like a guy. We're also fortunate enough to demo it on NPR last year. That was like real hardcore, like fast foray transforms, shit that you see on college and then you never see again. Well, we actually got to do real engineering at this company, so that was really, really exciting.

Outside of that, I think that we suffer from a lot of this – or not even suffer, but we face a lot of the same concerns that a startup of our size does. Living with some legacy code, refactoring things, setting things up to scale. We definitely subscribe to this idea that we shouldn't build tech until we absolutely have to, and our eng team is very, very small.

At the beginning, we were doing credit to program of course. Doing things that don't scale, that was our approach. Then as we grow, we have to start building stuff and codifying a lot of the things that we were doing while we were still learning. That's always a transition that is not

necessarily straightforward, and one that takes place all the time. It's really important to have good product; people that can figure out the best way to make that transition.

In terms of other tech, although these problems are interesting, they're not necessarily super hard. We're really, really trying to come up with the best experience around recruiting that we can create. So making sure that scheduling is airtight. Making sure that people are getting notified of everything that they need to be notified about without us having to ping them ourselves. Having this very robust system that can ensure that nobody is slipping through the cracks.

This is hard. There are a lot of great recruiters that do it very well. It's going to be a matter of time if we can see whether we can replicate some of that in product. Maybe we can't, but it's an exciting challenge to take on and it's something that we hope we can do very well, so that all of that minutia can get out of the way.

Whether it's recruiters or engineers or anybody else, like we just want to put the emphasis on two people that should be having a conversation, having a conversation.

[0:53:17.0] JM: Aline, thanks for coming on Software Engineering Daily. It's been great talking.

[0:53:20.3] AL: Likewise, thank you so much for having me.

[END OF INTERVIEW]

[0:53:23.0] JM: Thanks to Symphono for sponsoring Software Engineering Daily. Symphono is a custom engineering shop where senior engineers tackle big tech challenges while learning from each other. Check it out at symphono.com/sedaily. That's symphono.com/sedaily.

Thanks to Symphono for being a sponsor of Software Engineering Daily for almost a year now. Your continued support allows us to deliver content to the listeners on a regular basis.

[END]