1:11

Alexey

**This week, we'll talk about the role of a staff AI engineer. We have a special guest today, Tatiana. Tatiana works as a staff AI engineer at LinkedIn. Before that, she worked in physics and in health care. It's actually not the first time Tatiana is a guest here on this podcast – around two years ago, we already talked with her. The topic there was changing careers from academia (from physics) to machine learning. Now, after two years, we have Tatiana on again, and we will talk about the staff role. Welcome, again.**

1:50

Tatiana

Thank you. Thank you for inviting me.

# Tatiana’s background

1:53

Alexey

**Before we go into our main topic of being a staff engineer, let's start with your background. We already spoke about your background years ago, but maybe you can run us through it again, and also tell us what happened in the last few years.**

2:09

Tatiana

Yeah. I started my career with a PhD in physics, and I worked for a while developing lasers and optical systems. Then, during maternity leave, I got quite interested in the machine learning field, which was new back then, so I started doing pet projects and Kaggle and doing my own research and publishing papers in machine learning. After maternity leave, I moved to healthcare, where I was working on a computer-aided diagnostic tool for labor monitoring. At that point, we talked almost two years ago.

After one year of working in a hospital, I started to go to interviews at different companies, big tech companies, different startups, remote works, and so on. I started to work as a tech lead at LinkedIn, after some difficult choices, where I have worked for the last year and a half. That's been pretty much all the news for the last two years, but I learned a ton because it's a new industry and new environment.

3:15

Alexey

**That was pretty concise. [chuckles]**

3:19

Tatiana

I try. [laughs]

# Going from academia to healthcare to the tech industry

3:24

Alexey

**I'm just curious – it’s probably two different worlds, or even three different worlds – academia, then working at a hospital, and now working at a big tech company. All three are very different. [Tatiana agrees] How was it for you going from working in a hospital to joining LinkedIn, a big tech company? How was it for you? Overwhelming, I guess?**

3:49

Tatiana

It was transformational. It was overwhelming. In the first three months of onboarding, I was working about 80 hours, 70-80 hours a week. Big kudos to my husband, who understood that I needed the time, so he was helping with the kids a lot when I was just studying in the evenings. Yes, it was a big challenge, but also a big shift was in the mindset. When you work in academia – also a hospital is quite close to academia – your timeline is a bit different. You plan projects for one-two years. You're not in a hurry. You're relaxed. You don't have that many meetings. Therefore, you really learn a bunch of skills in academia: effective communication, working in a fast-paced environment, planning quarterly, and delivering results faster.

What you do in academia –what you need to do in a year, you accomplish in three months and things like that. In academia people talk like they have all the time in the world. For one hour, you can have a meeting without that much outcome. In industry, you have 20 minutes of a meeting where you have to decide what you're going to do. Then the next 20 minute meetings, 30 minute meetings and so on. You learn a lot about collaboration, communication, prioritization, and so on. But that is really useful.

5:19

Alexey

**80 hours a week? That's insane. Why did you need to put so much effort there? I guess, at the beginning, it was just too difficult and too overwhelming, so you needed to put extra effort into catching up with everything?**

5:34

Tatiana

I think yes. Well, it's not that I always worked 80 hours a week. It's not. [cross-talk]

5:39

Alexey

**At the beginning, you had to, right?**

5:43

Tatiana

At the beginning, I came and I realized, “Oh! Okay. Scala, Spark. I never thought about that. Kubernetes?” A lot of internal tools that I never used, and you’re also a tech lead, so you actually don't have that much of an onboarding or a relaxed onboarding. You're a tech lead and you actually lead others. You already need to deliver some design of the new system at scale that you never delivered at this scale. For example, a recommendation system for 80 million users – that's not something that I was doing before. But I was the tech lead, so all the decisions were on my side.

To make those decisions, I had to learn a lot about how other companies do it, read a lot of blog posts, and papers as well. At the same time I was learning all those tools that I never used. I guess I was just overwhelmed, so I tried to do my best to catch up. If I had to do it a second time, I would tell myself to take it easy. They give you time. You're not supposed to deliver results in the first two months. It's okay. But I think I was so worried that I wouldn't be effective enough that I worked the extra mile. Now I know that I could have taken it easier and taken my time to learn.

# What staff engineers do

7:04

Alexey

**Yeah, that's pretty interesting. I want to ask you more about the tools and how you approached this learning. But before that, I think it would be interesting to know what tech leads do – what staff engineers do. You mentioned that you need to lead others, you need to design systems, and you need to make decisions. What does your day look like? What kind of things do you need to do?**

7:30

Tatiana

In short, I'll say it like this – I'm paid for my opinion. That's in short. In detail, as a staff engineer – there are different kinds of staff engineers. Specifically, our team is a horizontal team. This means that we have a lot of internal stakeholders in different orgs. That means a lot of interactions with different teams, with products, with data scientists, with other teams, with annotators, with the legal team, and so on. Almost half of my work is different meetings and interactions. You are analyzing strategy, you're defining a roadmap for a set of projects, you agree with other leaders and align on different cross-functional collaborations, different projects.

You also own the delivery of the results and their impact. A lot of work goes into those meetings on alignment and defining the roadmap and strategy and what you're gonna do and defining the objective – both business and technical goals, basically. You collaborate with products a lot on that. On the technical side, you also mentor a lot of people – not only in your immediate team, but across different teams as well. You work with ML engineers, annotators, engineer linguists, data scientists, UI engineers – and you supervise them all to make sure that the project is delivered on time. You are part of promo committees and hiring committees, where you review the work of others. As technical AI staff, you do a lot of machine learning design. Obviously, if you're staff in some other area, like infrastructure, you probably work on system design more.

A lot of work is dedicated to creating designs and also reviewing designs of others. When people create some designs, that has to be reviewed, and often staff is the person who is brought in to give their opinion. You also do a lot of code reviews. People who work with you write code and you review it for your team, and also outside your team. You don't write much of the code. I still try to keep hands-on. I dedicate one day a week for myself to write code. That's my choice, basically. But most of the time, you don't really write the code.

There are some staff engineers who don't write any code and there are some staff engineers who write more code – it depends on what kind of staff you are, as well. As for me, I write a lot of docs, obviously, and a lot of plans and roadmaps. Everything needs a doc. If you have a legal alignment, for example, you need to document it – and things like that. That's pretty much all the pillars. So design and systems, mentoring others, giving your opinion, improving craftsmanship of the team and outside the team, defining strategy, defining roadmaps, and making sure that the bigger projects involving many teams go to production and that they make an impact.

11:00

Alexey

**Quite a lot for one person, isn't it?**

11:04

Tatiana

Well, you’re more leading that. So you don't do all that yourself. People who implement things work with you and you collaborate with products as well, who define what business impact to expect and things like that. And you collaborate with the legal team, which advises you on what you can and cannot do. Things like this. But yes, it's a lot of coordination and meetings. On a weekly basis, you can meet like 5 to 10 different teams and with different people. Again, this is cross-team collaboration type of stuff – when you have a lot of cross-team collaborations. That's my role.

There are also different types of staff. For example, there are people who have very great technical expertise and they know some particular narrow field very well (not necessarily “narrow”). They're pulled into troubleshooting. If there is a critical problem, and something doesn't work, they're pulled in to solve the problem, like “firefighting”. So there are these kinds of staff. There are people who have great depths and widths in a particular field, for example, machine learning. They act as advisors to the leadership. They're like personal advisors on the technical side. There are also these kinds of staff.

So there are people who have very great expertise compared to others, so they mentor and teach others this expertise and focus more on hands-on. They create a lot of leverage by actually coding. But that's not very common, usually all the staff don’t do a lot of coding. It depends on the role. If you're a horizontal team and you have a lot of different interactions, you're not paid for coding – you're more paid for your opinion and making decisions and realizing and all this and defining roadmaps. However, I'm still trying to be involved in coding myself, not only reviewing. But most of the code I don't write – I review it. It’s like one to ten – for each submission that I do, I review ten. Something like that.

13:24

Alexey

**Just to summarize what you said – there are multiple kinds of staff engineers (staff-level people) or profiles, I think I've also heard the word “archetypes”. [Tatiana agrees] What you do is collaboration and coordination work. Then there is somebody who has great, deep technical expertise. They are brought in when there is a problem in some specific area and they help solve it. And then there are people who have broad expertise and they act as advisors. Did I miss anything?**

14:03

Tatiana

Broad and deep in a particular area, yes.

14:09

Alexey

**It’s one person, right? Deep in one and brought in others.**

14:13

Tatiana

No, there are people who manage to have both after a while. If you're 20 years in the field. [chuckles]

14:20

Alexey

**It's like this T shaped profile, right?**

14:23

Tatiana

Yes. There is a book – I think it's called Staff Engineer Path, which defines those archetypes quite well. I quite agree with the definitions that I read in the book.

# Transferring skills from academia to industry and learning new ones

14:41

Alexey

**Okay. So you're paid for your opinion. You spent 50% of your time in meetings. You work on strategy, defined roadmaps, and you also mentor people, and take part in design decisions. You mentioned that at the beginning, you needed to work something like 70-80 hours because I guess many of these things were difficult for you. You weren't certain that your design decisions were good, so you wanted to improve your skills. Did I understand that correctly?**

15:16

Tatiana

Yes. Some of that was very well transferred from academia. For example, all this communication, collaboration, roadmap strategy – it's exactly how you define your research proposal. You write a plan for two years, you define goals, you loyalize different teams, and so on. Mentoring is a big part of academic work, so nothing changed there. But I did not work in NLP before and I did not work exactly with this type of recommendation systems at that scale. So that was a big part of learning. I took a couple of courses, both in NLP and recommendation systems. But I also read a lot in blog posts and in papers that other companies published.

Also, a big part of learning was internal tools. You can't learn them outside, pretty much. You can but it's not that common outside. You learn a lot of those internal tools. Plus, I didn't work with Scala and Spark before and with Kubernetes I did not work before. There were also some gaps, because I never worked as a software engineer. I probably had a little bit more gaps in the technical part, but a little bit fewer in the strategy and communication, thanks to academia.

16:37

Alexey

**How did you approach learning? How did you know what to learn next? Did you create a plan and follow this plan? Did you ask your colleagues for their advice? How did you do this?**

16:47

Tatiana

My onboarding was rather hectic. I was just learning whatever I could without properly understanding what to prioritize. In the beginning, you're overwhelmed simply because you don't know what you need to prioritize and how. If I had to do it a second time, I would understand that better and I wouldn't require that much time. Basically, I was taking one course after the other without a clear understanding of what was more important at the very moment. I only got this understanding in about two-three months.

# The importance of having mentors

17:23

Alexey

**That does sound very overwhelming – trying to learn everything at the same time, or at least one after another. If you could give a piece of advice to yourself in the past, what would it be? How would you suggest approaching this to yourself in the past?**

17:45

Tatiana

Yes, that's a good question. Onboarding is always a challenge. And onboarding in big companies is definitely a challenge. So I would find more mentors from the start and just ask them to help with this technical plan. In fact, I had a mentor, but he immediately went on parental leave. That's why I didn't have a mentor and such during onboarding. Usually, in the onboarding process, you have a mentor who helps you and guides you. After that, we spent quite some time developing our onboarding for the next engineers, so they wouldn't be overwhelmed. Hopefully, that worked better. So that's number one – find mentors immediately and ask them to help. Also, communicate with your manager all the time about priorities and expectations. Don't feel the urge to deliver something in production in the first two months. It's okay – you have your time to onboard in a big tech company. Of course, if you're in a startup, you need to deliver by the end of the week. [chuckles]

18:59

Alexey

**[chuckles] No pressure, right?**

19:03

Tatiana

I’m just kidding. But yes, the onboarding time depends on the size of the company.

# Skipping junior and mid-level straight into the staff role

19:08

Alexey

**Maybe startups don't really need these kinds of profiles? They don't need somebody who coordinates because if it's just one or two teams, then people can just talk to each other without this special role, probably. I know that you joined LinkedIn as a staff engineer after working in healthcare. You managed to skip like a few roles before staff. I know that their usual career progression is – you join a company as a junior, there are juniors, there are middle level professionals, then there are seniors, and then only after that staff engineers.**

**From what I understood, you didn't have experience in industry, yet you didn't join LinkedIn as a middle level engineer or a senior engineer – you joined immediately as a staff engineer. I’m wondering how this happened? How did you manage to kind of jump over these roles and land a pretty high level position?**

20:16

Tatiana

Yeah, I get this question a lot. The reason is that people think that whenever you just start a new career, you have to start at a junior level. This is not correct. It's not that I didn't have a career before. I actually had a PhD in physics and then I worked in science for six years, leading my own projects as a research fellow. When you're a principal investigator in science, what you do is the following – you come up with an idea for a project that nobody on Earth has ever done before. You need to find partners to make this happen.

You need to find a host Institute who has enough equipment, a partner institute, industrial partners – ideally, five different organizations across the globe. You need to write a proposal when you put your implementation plan forward. You sell motivation – why you need to do that. You do budgeting. You do your strategy, your OKRs – your everything. And if you are in the top 9% of those who submitted proposals, you get money. If you don't, you're jobless. That's how it works in science. And that's what I was doing.

24:24

Alexey

**Cruel.**

21:26

Tatiana

Yes, it's much more complicated compared to the situation in industry in this regard. I got like three successful grants in a row, which is quite good. To get three successes, you obviously need to have three or more failures as well. Not all of my proposals were successful. But they teach how you find collaborators, how to align with them, how to agree, how to write a plan, how to make a strategy, and so on. And then you lead those projects, so you hire PhD students, you mentor them, and you go through problems, because it's high-risk projects in science since nobody's done it before. You don't know if you can do it or not. Sometimes it leads to invention disclosures, some great papers, and sometimes – oops, it's not gonna work. This is also normal since it's very high-risk projects.

All that you can transfer – a lot of those skills are actually transferable, in terms of ownership, leadership, mentorship, strategy, roadmapping, and all that stuff. There are some things that you don't do. For instance, you don't program in Scala, but you program in FORTRAN or MATLAB. It's not exactly that, but not that much different. It's technical skills where you need to fill the gaps. But for technical skills, I'd say 6-12 months, you work yourself, you learn to get it. It's much harder to learn soft skills, in my opinion, and to uplevel your communication and collaboration. It's way harder than learning Python, for example, in my opinion. When you move to industry, it's important to put the right angle on all your previous experience and sell it in the right way. In the beginning, I was making mistakes when I was talking about lasers too much.

At the end, I stopped talking about lasers at all – I was talking about what I was doing in terms of collaboration, alignment, delivery, and so on. You could see straight away how it aligns well with the staff role. What also helps – if you look at the tech lead, lead role, staff role in the organization, in terms of the descriptions and expectations, you see how your previous experience aligns with what’s in the docs. The docs are quite possible to find. Although it's not typical to have a career change like mine, I'm not the only one like that. I know a bunch of people who were in academia and/or healthcare, like my mentor who worked in the hospital, who was leading me during my internship. He moved to Qualcomm straight as a staff engineer. Now he is currently a director. Then my manager at LinkedIn, she also moved from academia to LinkedIn, straight to the staff position.

It happens, it’s just not that common because not many people change the area. If you're a really great scientist, you actually can start as a distinguished. If you have this level of expertise, like a highly recognized scientist in academia, you can move to the highest level. That also happens. Not very common, but it happens. The other option is – some people do a PhD, for instance, and they do a startup. Then if the startup is acquired by a big company, they can start at a high level. I know an example, when Lyft acquired a startup and that startup CEO started as a director. So he joined the industry at the director level. It's not that you have to start as a mid-level or junior every time. If you already had some career growth and you have experience, it just depends which role you can fit most of your transferable skills to.

# Convincing employers that you can take on a lead role

25:30

Alexey

**Yeah, that's interesting. The main takeaway for me is that it's a myth that you have to start from a junior. You need to sell yourself in the right way. I have two follow-up questions here. Maybe first, okay, yes – you need to sell yourself. But how do you convince them to actually trust you?**

**How do you convince them that the skills that you have are transferable from academia to the industry? At the beginning, you also said that they’re very different environments. So how do you convince your employer that the skills you got in that environment are transferable to this new environment? Because they're pretty different.**

26:18

Tatiana

Yeah, true. However, it also depends on the angle. In the beginning, I wasn't quite successful with that. In fact, in the first interviews, I was failing. But then, after some practice, I learned which angle to choose, how to present my projects. One of the ways is to focus on the research projects that you've done in partnership with industry, and I had some of those. If this was a partnership with industry, and you can say that, “Okay, we delivered an optical system for Alcon, who produces lenses, to identify defects on those lenses. We used a non-contact method,” and so on. That's pretty much an industrial thing. It very much applies. So focus more on applied projects, for example.

If you’ve collaborated with big companies, that also helps. Things like that. Obviously, they had to take a risk with me, as well. But you just have to be better than all the other candidates. For example, if your experience was different, it's probably not that preferable, compared to if you just moved to another big tech company in the same role. However, you're always assessed compared to other candidates. If you pass all interviews better than other candidates, it's okay. They may consider you, even if you changed the role. So you have to prepare better for interviews and be better than others.

27:50

Alexey

**We wanted to spend quite some time talking about interviews. But before that, I'm really curious – how did you understand for yourself that the staff role is really for you? You didn't have this prior experience in industry. Somebody probably told you that you don't have to start from a junior position, right? Then you learned about this staff role – you learned it exists – and you learned that it's something that might fit your background and your experience, and you started preparing for this. So how did this happen?**

28:25

Tatiana

No, it wasn't like that. I was just applying for any positions. I did not contact them, they contacted me first. So the same was Meta? I did not contact them. They wrote to me first. In fact, when I was working in the hospital, I did not even consider big tech companies. I knew it was very hard to get to and that you need to prepare for interviews and solve those LeetCode problems. So I didn't apply to those, but rather they reached out to me. The first was Meta who wrote “Your resume is fantastic for our role.” I was a bit surprised, but I decided “Why not try the interviews? At the end of the day, I have nothing to lose.”

That's when I talked to the recruiter, who decided that it would be for a senior staff role after we discussed my experience. I thought “Okay,” and I went to the interviews and I failed coding. [laughs] But the rest was brilliant. I got overwhelmingly great feedback on system design, machine learning design, cultural fit, everything – just coding, obviously, wasn’t good, because I didn't prepare. Then I realized, “Okay, it's not that hard to actually pass interviews at big tech companies. All I need is just to start doing LeetCode and that’s it.” [chuckles]

# Seeing failure as a learning opportunity

29:38

Alexey

**You say it as if it’s so easy. [chuckles]**

29:41

Tatiana

Back then I was also interviewing with startups, and they were considering me for a lead role. In fact, I had an offer for a lead role at one of the startups – leading the direction. They kind of justified it this way after we discussed my experience. That put my level up a bit from what I initially thought, because this was validation from outside. As I said, I felt coding would matter, but they did not send me away because I was quite good at system design. So they decided to give me a second chance and waited. They agreed to give me a chance to prepare for coding interviews and gave me 5-6 weeks. I started learning LeetCode full time.

At this time, when I was almost ready, LinkedIn came in and just wrote, “Hey, we have a staff position. Would you mind applying?” I decided “Why not go to interviews? I have nothing to lose.” And I passed because the LeetCode was ready. [chuckles] So it was them who decided to contact me. Why did they decide to contact me? Because somebody referred me. And why did somebody refer me? Because I helped that somebody two years before that. Help people – Karma comes back. When they offered me a staff role, I thought to myself that I wouldn't even apply for that position, maybe. But they approached me first. Then I found that, indeed, I am quite a good fit for this particular type of stuff. There’s a lot of cross-team cross-functional collaboration. For this particular type of work, I'm a good fit.

31:21

Alexey

**I think two years ago, we discussed this at length – how exactly you can help others. We talked about forming teams, how you approach a career change, and helping others was one of the things there. I guess you were doing this for quite some time and then karma thanked you for it. I liked what you said just now multiple times about having nothing to lose – every time an opportunity came, you said, “Okay, whatever. What can possibly go wrong? They declined? That's not the end of the world.”**

32:08

Tatiana

If they deny you, you're exactly at the stage where you are now. You cannot lose. I don't know. Some people are afraid of failures, but in fact, those are not failures. Failure is when you lose a lot. You cannot fail in ML engineering. That is a bit of a problem. But for interviews, it's not a failure, even.

32:28

Alexey

**Rejections suck, right? It's kind of bad for your self-esteem, for your self-worth. Like if they say, “I'm not worthy of this staff engineer position, then I'm not worthy at all.” No?**

32:44

Tatiana

No, not for me. I look at it like an adventure, “Let's try it and if I fail, okay. I learned this, this and this.” I failed a lot before I passed, by the way. In the beginning, I was failing every interview. I failed Lyft. I failed Meta in the end, because I didn't pass that coding again. [chuckles] And I failed lots of startups and so on. But I don't feel that it affects my self-esteem because I consider it learning. I need to learn the interview process. The interview process has nothing to do with my abilities, expertise, and so on. I don't consider evaluation at an interview a true evaluation of you. It's more like an evaluation of your interview skills, in a way.

33:38

Alexey

**How did you arrive at this conclusion? I mean, I totally agree with you, but it probably took some time for you to realize that it's not about you and that you just need to know the process.**

33:54

Tatiana

Well, trying and failing. [chuckles] That's how I arrived at it.

33:58

Alexey

**I see. [laughs]**

34:00

Tatiana

Yeah. I was trying in the beginning, and I learned that, “Okay, you need to change this. You need to work on this. And the next time it will go better and the time after that it will go even better.” I did a lot of mock interviews. I remember you were helping me. Thank you for the mock interviews as well. I asked many people to help me with that and I learned a lot about that as well. When I started to realize which problems I need to solve, how to present myself, and so on, I started to be better at this and clearly understood it's more about preparing for the interviews and learning how to explain your thoughts well.

# Preparing for coding interviews

34:40

Alexey

**I think it's also one of the things we talked about last time, that failure is not necessarily bad. It's an opportunity to learn. [Tatiana agrees] You said that during the interview with Meta, you did quite well at system design, ML design and there were other things where you didn’t do well, like coding (LeetCode kind of questions). So how did you actually, without working in the industry, manage to prepare yourself and pass these interviews?**

35:15

Tatiana

For the big tech companies – let's separate startups from big tech companies, since they have different flows. For the startups, it's actually hard to prepare because they can ask you whatever they want and in any manner they want. For the big tech companies, the process is more or less standardized and goes through certain steps, which you can familiarize yourself with and start preparing. One of the important aspects of big tech companies, during coding interviews they ask you the types of questions that you can find in LeetCode.

So far, I haven't seen many people who are able to solve them without preparation. There are guys who can come and solve it, but they participated in competitive programming and they solved about a 1000 of them in school. So that doesn't count. [chuckles] If you never solved them, and you never saw 1000s of them in school, the big chances are that you will not pass that interview. That interview is more about preparation than about your coding skills, which is a bit annoying for most programmers in the boat. But this is a scalable solution and that's how you can assess different candidates fairly. That was the biggest challenge for me. Although I know that for some people, machine learning design is the biggest challenge. Meanwhile, some people are not very good at behavioral interviews. Everybody's different.

For me, the biggest challenge, as I realized, was this LeetCode-type of interview. I got LeetCode Premium. That was a good idea. In LeetCode Premium, you don’t only see popular questions, but you can also see solutions, and they also provide you cards with theory. So I will start by studying recursion, how to form it, and then I'll go through some common tasks with solutions. After you've done 10 of those, you can solve the 11th yourself, hopefully. For every topic, it was like that. All the topics were new for me, I didn't even know about recursion when I started, because I don't have a computer science degree – I didn't do algorithms in university, so maybe that's why it was the biggest challenge for me. In total, I'd say two months is enough.

As for me, at that time, the work in the hospital stopped because that's exactly what happens when you don't raise funding for the next proposals, as I described. Sometimes you fail and then you're jobless. That’s exactly what happened in the hospital – we didn’t raise money for the next stage – and everybody who was not a doctor (who was a contractor) became jobless. This means that I had plenty of time for coding. I was coding nine to five every day. In 5-6 weeks, I got like 300 problems in LeetCode. So if you solve about 300-350 problems, mostly focusing on medium (not easy) and do all the theory – I'd say for some big tech companies, it will be enough. Maybe not for Google, but for many it will be enough. That's how I prepared for LeetCode.

38:42

Alexey

**That’s admirable perseverance – six weeks in a row, nine to five, 350 problems on LeetCode. That's really an accomplishment. How did you not give up?**

39:03

Tatiana

I liked doing it by the end. At the beginning I hated it, but after some time, it's actually cool! I liked it. I like physics, maths, and those types of algorithms. I found it pleasant doing those challenges after some time. So I am a geek, yeah. I like such things. [laughs]

39:22

Alexey

**I guess when you solve a problem yourself without looking at hints, you kind of get a…**

39:27

Tatiana

Dopamine. A lot happier.

39:29

Alexey

**Yes. [chuckles] Exactly.**

39:31

Tatiana

It's a bit like gaming, right? Some people like gaming – when they're successful they get these hormones. It was the same for me when I solved LeetCode myself.

# Preparing for behavioral and system design interviews

39:44

Alexey

**LeetCode is just grinding, right? You have to do this a lot and you have to solve a lot of problems, especially if you don't have a computer science background, such as in your case. How about other parts? System design, ML designed, behavioral… I think for behavioral, like you said, you have the soft skills from academia, and then it’s probably just thinking about situations you had in academia and then describing these stations. How about these two others? System design and ML design, which you don't do in academia, I guess.**

40:26

Tatiana

Surprisingly easy for me, was ML design. I realized that in academia, I was designing optical systems in lasers, and they have much in common with machine learning designs. You also start from scratch with the first principles, you understand what you want to accomplish, what data you need to collect, how much data, how you want to annotate it, and so on. You can see the different options and their tradeoffs – basically, ML design, in the way of thinking, it's quite similar to design in physics. Yeah, they’re completely different fields and you need to know different techniques and stuff. But the way of thinking, of decompositioning a problem, and planning, and about the details, and your intuition – physics helps a lot here, to be honest. That was my discovery.

To do that properly, I've done several mock interviews on ML design with people I know. Mock interviews helped me the most. I also found some blog posts about ML design and a couple of YouTube videos. There was also the Grokking ML Design course, but I did not like that one, to be honest. The best was a blog post by Patrick Kalina, who basically wrote how to answer ML design on the staff level. He's a staff at Pinterest. I followed his guide, more or less – his plan.

Another useful thing was, going on the blog post for each particular company and reading about their ML designs. For example, I go to Lyft’s site, open their blog post and read about their ML designs. Then in interviews, when they ask me, I say, “Yeah, I've read in your blog post that you do it the following way.” That's already a plus, because you prepared – you know what stack they're using and you know what kind of designs they’re working on. If you go to YouTube – you can just open the papers, they publish recommendation systems and archives even. Read whatever is published in the blog posts and familiarize yourself with how they design things. I think it makes sense to do that.

When you study for a number of companies, like YouTube, Google, TripAdvisor, and so on, you start to see the patterns as well. Also, they often write – for example Airbnb also wrote about their ML design attempt – they write what worked, what didn't work in industry (in real life). Like all those academic narrow papers, what you see in academic papers is not always really applicable to industry. Or maybe they just move a metric by 0.001% with a lot of effort, and that's not what you're gonna do. So having those blog posts from companies helped identify patterns and answer a lot of questions.

43:36

Alexey

**How did you approach systematizing all this information? To me it sounds like a lot – a huge load of different sources and how do you even manage to process and learn from this in a way that you could talk about it at an interview? Did you take notes? Did you use some special approach? How did you do this?**

44:08

Tatiana

I'm old school, so I keep things like this [shows notebook]. Oh, that's about birth. That's when I was studying about birth. I make… [cross-talk]

44:18

Alexey

**“Old-school” meaning listening to this without videos?**

44:21

Tatiana

“Old school” meaning you go through courses, you write notes, and then you summarize. So I have notes on ML design, on system design, and I have three of those notebooks on LeetCode. With LeetCode, I realized that it works better when you write by hand. Whenever I have a challenging task that I can't solve and easily understand, and even after I understand it, I still want to engrave it in my memory – I just take a pen and write it down, like literally write it down, because that's how I remember things. It helps me. It was a lot of systematic knowledge, but for ML design, it just took maybe a week. For system design, it took me like five days to prepare. For LeetCode, it was almost two months. For other people, it can be different timelines.

For system design, I just followed some advice on the internet and I bought the Grokking System Design course. It was quite cheap – I think $50 or something along those lines. I bought it and I started it. That was enough to pass the interview. Although, if you want in-depth understanding, people advise reading the Data Intensive Applications book, which I also read parts of. But for the interview, actually having that course and 20 mock interviews (mock interviews are important, very important) that will be enough to pass the interview. But this doesn’t make you a great system designer, of course. You need to have some hands-on practice after.

46:00

Alexey

**After you pass the interview and start working at the company, I guess. [chuckles]**

46:04

Tatiana

Yes. You understand you understand the basics. People who do ML design need to understand system design, because you're designing ML systems at scale. But if you're hired for an ML role, the ML part, I'd say, is more important than your experience with load balancers and things like that. There is always the infrastructure team in big companies who take care of that. But you need to understand on a high level how it works, more or less. But if you don't know all the load balancing techniques after some time, maybe it's not crucial. It depends on your role.

46:44

Alexey

**One week to prepare for system design is pretty impressive. I don't know much about your prior background before that, but I guess terms like “load balancer” didn't mean much to you before you started reading this thing. Processing all that in one week is impressive. You probably have a very good memory.**

47:12

Tatiana

Yeah. I have a good memory, plus what works for me is deadlines. I just scheduled that interview with Facebook, because why not try and then for the ML part, I already tried and failed at Lyft, and I passed at Shutterstock, and I had some offers. So the ML part was pretty much trained. System design was specifically for Facebook, because in other companies, they didn't ask for it. They didn't ask for system design at LinkedIn, for example. At Facebook, they asked for system design. I did not have a chance to prepare for that before but I already scheduled an interview, so I had a deadline.

I just did the entire course in five days and had a few mock interviews with people who worked at Facebook. A great help was that I could practice by doing a mock interview with a person who worked at Facebook, and he knew what was expected on system design. But I didn’t pass because of that – I passed because the role was more about an intersection of machine learning and optics. We didn't dive that much into system design, but we were discussing things like 3D reconstruction and epipolar axes and optics. And that's my field. I got a bit lucky as well with the topic, but that's how they selected my resume out of all these other resumes. I think they needed somebody who has expertise in optics. And that helped.

# The importance of having a network and doing mock interviews

48:43

Alexey

**From what I heard from you, having a network is really helpful. If you know somebody or can get to know somebody from Meta, or from the company where you have an interview, and just ask them “Hey, can you have a mock interview with me, please?” And then they just say “Yes,” and that's how you prepare, right? As I understand, just taking the courses is not enough, right?**

49:09

Tatiana

No, it’s not enough. It's hard to put it all in your head properly. During mock interviews, you learn more. Yes, it's nice if you have mentors and people who are willing to help you, but ideally, I try to have a mutual connection, where I give value to the person and he gives value to me. What can you do if other people are much more experienced than you? You cannot provide them more than they contribute. But there were also mock interviews when we did it mutually. I interviewed the guy and he interviewed me, and we both prepared for the same level of position.

There were a couple of interviews like that. Luckily, I had mentors who were willing just to help me, but I would just send them a painting as a thank you, for instance. I like to give something in return, but it’s not always possible to give the same level of return because they're more advanced. Well, then you can gift a painting, for example.

50:09

Alexey

**Sometimes people are just fine with helping without expecting anything in return, because who knows? Maybe in 10 years, you will be their manager or something. Like this karma thing that you mentioned – you help somebody and then after some time they refer you and you get a job.**

50:27

Tatiana

Yeah, I think that's also a good strategy. I always try to help other people. There are people whom I mentor, where I’m more advanced than them and there are people who mentor me. So across the globe, there’s all this getting help and providing help. I like to keep it balanced – even more on the helping side. Also, I think it makes you happier if you help somebody and you see their progress.

# How much do staff engineers work with building pipelines, data science, ETC, MPOps, etc.?

50:52

Alexey

**It does, yeah. I see that there are some questions. A question from Harry, “As a staff AI engineer, how much of your time do you deal with data engineers, data scientists, with building ETLs, building pipelines, and doing all this MLOps stuff?”**

51:10

Tatiana

Yes. Again, it depends on the project. There are some projects where you have a lot of collaboration with data scientists, because you require some initial analysis and things. And there are some projects, where you collaborate more with the product team. In some projects, you'll have to collaborate quite a lot with legal. So it's kind of project-based, but as I said, all those collaborations take around half of my time.

Then the other half goes on design and reviewing and preparing the docs and things like that. For ETL pipelines, I do implement some because I like to stay hands-on, but most of the time, they're implemented for you by people who you mentor. You just decide what has to be done and they implement it and you review the code.

52:10

Alexey

**You said that you do quite a lot of code reviewing, right?**

52:15

Tatiana

A lot, yeah. Quite a lot of code review.

# Context switching

52:19

Alexey

**For me, doing code review is difficult because if I'm working with multiple projects at the same time, every time I review a piece of code, I need to get back into the context of that specific project and understand why this code is written in exactly this way. This is very difficult, especially if it's like three, four projects at the same time. How do you do this?**

52:43

Tatiana

Yeah, I could have like five projects at the same time. But I remember each of them quite well. You jump between meetings literally, now you’re discussing this project, the next project, 25 minutes, 25 minutes – you just jump between meetings and you're already switching contexts. I think this just trains you to always be in context. Or maybe it’s just my thing that I am quite easy to switch and I remember what's going on in like five different projects.

53:14

Alexey

**Did you do anything special to get this skill? Or was it just always like that for you?**

53:21

Tatiana

It was always like that for me. Context switching was always easy for me.

53:30

Alexey

**Okay. Because I was going to ask you for advice on how to approach that. But… [chuckles]**

53:36

Tatiana

This is just a gift, maybe? I don't know. [chuckles] Yeah, it's easy for me to switch from meeting to meeting, completely changing contexts, different projects that we discuss. Hard to advise on what to do about it. [chuckles]

53:57

Alexey

**Maybe I guess what worked for me in the end, I think, is somehow to arrange things in a way that doesn't require context switching, meaning doing things sequentially, and then maybe have this box. But yeah, it's not easy, at least for me.**

# Advice for those going from academia to industry

54:13

Alexey

**Another question that we have in the list that I didn't ask you is “Let's say somebody wants to follow your path and go from academia to industry, and also get on a staff level. What advice would you give to them?”**

54:36

Tatiana

Yeah, that's a good question. First of all – don't be shy going to the tech lead role or lead role. If you were leading in academia, you have most of the challenging, transferable soft skills. It's much harder to learn that than to learn programming or Git and other tools. Prepare some pet projects that are in line. But also look at your own career as a whole and the parts where you were leading things, initializing things, mentoring, teaching, upleveling the team – all that is quite transferable and important. If you are able to understand the requirements of the future role of a lead or tech lead, get them, and see how this aligns with your previous experience. Prepare well for the cultural interview.

The behavioral interview is exactly when you show yourself at the right level – most of it. You do that during all the interviews, but the cultural and behavioral interviews are where you mainly show your leadership skills, your ownership. Prepare well for that. You may also want to practice that with somebody who already is in industry and in a high level role – if you happen to have such a mentor, like a director. I didn't, so I didn't prepare. But if I could, I would use that chance. Also, get feedback. Some people even get coaching and consulting and all that.

Obviously, also prepare well for the other interviews, because even if you're a great leader and everything, but you completely fail the coding, we know you will not get the job, as we know. [laughs] You have to be ready for the other interviews too. Give yourself time. It's okay, if you fail for some time. It took me a year between when I started interviewing and I couldn't pass, up to the point when I had all the interviews turn into all offers. I had like five interviews in process and like five offers – 100%. So from 0% to 100%, it took me one year and a lot of failures. Be ready for that. If you don't fail, and you don't try, you're not gonna get it. It’s like kids who are learning something – they always fail from the first time.

56:54

Alexey

**Learn to live with failure, right? It's not about you, personally.**

56:58

Tatiana

No, it's not about you. I wouldn’t even consider it as a failure. For me failure is something where you really lose. Like mountaineering is dangerous, but with interviews – if you fail, you're exactly where you started, so it's not really a failure. It's more like a try with no risk. You have nothing to lose, really.

57:24

Alexey

**You're saying this, like the third or fourth time? But this is the case, right? [chuckles]**

57:29

Tatiana

Because it is the case, yes. When people don't want to try because they're afraid, I completely don't understand it.

# The most exciting thing about working as an AI staff engineer

57:40

Alexey

**A question from Chali is, “What is the most exciting thing about working as an AI staff engineer?”**

57:49

Tatiana

Oh, yes. This is a very interesting field. Look what's happening now with all this ChatGPT and diffusion models and so on. You’re at the edge of technology, you're actually like one of the first people who are making the future and moving into the future. That's very exciting. You can actually try lots of different technologies and you have the freedom of what you want to try, because you can also lead some R&D projects, for example. I find it very exciting that you have to learn in this fast-changing field, when the future is really coming. Every few months, you have some new revolution, some changes – that's very interesting and that's very cool. Mastering this and understanding how it all works is very interesting.

The other part is, if you really like this (I was craving for this in science) to see quite a big and fast turnover between your work and some impact metrics of revenue and so on in the company where you work. When you actually push things into production and you see how they influence things and you see the impact – it's very pleasant. This is not always the case in science. Sometimes you develop lots of things, you write papers, but you don't feel its need and you don't feel the impact. It takes time in science to make an impact.

# Tatiana’s book recommendations

59:25

Alexey

**Yeah, thanks. I know we're running out of time, but this one will be the last one and really quick. Are there any books or other learning resources that you can recommend to the listeners?**

59:37

Tatiana

Yes, there are lots of books depending on what you want. [chuckles] I think… [cross-talk]

59:41

Alexey

**I was thinking like one or two. [chuckles]**

59:45

Tatiana

I like communication. For communication, I think it's important to learn how to communicate. There’s “Crucial Conversations: Tools for Talking When Stakes Are High” and “Never Split the Difference: Negotiating As If Your Life Depended On It”. Those are the best two. If you want to understand about the staff role, the “Staff Engineering Path” is the best book on that, I'd say. I like books in general about leadership and ownership and stuff. There is the book “Extreme Ownership: How U.S. Navy SEALs Lead and Win,” and for career it's highly advised to read the book “Rise,” by Patty Azzarello. That would be career-oriented books.

60:34

Alexey

**Thank you. It’s always a pleasure having you here. Maybe in two years we should have a chat again and see what changed. Thanks for joining us today. Thanks for sharing your knowledge, your experience of getting this staff AI engineering role, and sharing what you do with us. Thanks, everyone, for joining us today. Everyone, have a great weekend.**