Engineers' Survival Guide

Advice, tactics, and tricks

After a decade of working at
Facebook, Microsoft, Snap

Merih Taze

Engineers' Survival Guide

Advice, tactics, and tricks

After a decade of working at
Facebook, Snapchat, and Microsoft

Merih Taze

ENGINEERS' SURVIVAL GUIDE

For more information, contact book@engineerssurvivalguide.com.

Visit the book's website at www.engineerssurvivalguide.com.

ISBN-13: 979-8-9853499-0-0

To my family, who always believed in me and
supported me with everything they have.

Table of Contents

1. Introduction and Who This Book Is For

I'm so excited to meet you here. Being an engineer is a constant challenge. I have spent over a decade in tech, working for Microsoft, Snapchat and Facebook. I am still learning something new every day. There are key differences of engineers who excel in their career. Inside you will find the most important ones that made the difference for me and thousands of brilliant engineers I have observed over the years. I will share all the advice, tactics and tricks that will help you survive and thrive in your career.

If you are **working in tech**, this book will be your best friend.

If you are **not inside the tech world**, you will find many useful topics but expect a few topics to be unrelated.

This book focuses on soft skills like conflict resolution, design discussions, unblocking yourself, dealing with ambiguity, and how to set up your career to be successful, and more importantly, how to be happy in your life.

Interested In More Resources?

If you are interested in learning more and staying up to date:

Blog / Contact Us Form:
EngineersSurvivalGuide.com

LinkedIn Page (Engineers' Survival Guide):
bit.ly/survivalLinkedin
(case sensitive)

Facebook Page (Engineers' Survival Guide):
bit.ly/survivalFb

(case sensitive)

Discord (Engineers' Survival Guide):
bit.ly/survivalDisc
(case sensitive)

Email List:
bit.ly/survivalEmail
(case sensitive)

Would love to hear from you

Don't be a stranger. I love meeting people and would love the opportunity to improve myself. We are going to be talking about the importance of **Collecting Feedback** in a later topic. This book is no exception. If there is anything on your mind you would like to share, any parts you think needs to be improved, or if you simply would like to say hi and connect with me, please do not think twice. "**Act then think**". There are multiple ways you can reach out to me;

- Use the Contact form at engineerssurvivalguide.com

- Email me at book@engineerssurvivalguide.com

- Connect me at linkedin.com/in/merihtaze/

- Use Discord, LinkedIn or Facebook pages shared above

The Power of Giving Credit

We will talk about the power of giving credit in detail in a later topic. Giving credit is a great way to share love. We are all fast to act on our frustrations but not that fast to share the things we appreciate. If you find the content in this book useful, please go ahead and leave us a review. I personally read every review and there is no better feeling than seeing I was able to help someone.

If there are parts you aren't thrilled with, please give me the opportunity to improve by contacting me directly with one of the ways shared above.

2. First Day at Work and Imposter Syndrome

I was exposed to programming at a young age. My father and uncle were both familiar with programming and expected me to learn the basics. Like every other kid, I was mainly interested in playing computer games. The price of doing what I wanted was to solve some little math problems with code. I never enjoyed it but always tried my best to make sure I could get back to my games as soon as possible.

They were simple programs to do things like addition or subtraction done in bite-sized codes, no longer than a few lines. Learning how to code at an early age helped me stay a step ahead of my peers throughout my university career. Thanks to the bell curve, most of my grades were A+. I graduated with high honors and became the valedictorian.

During my last year, I interviewed with Microsoft and was thrilled when they extended me an offer. It was unheard of. No one from my university had ever been given an offer to work for a world-famous company like Microsoft. Without knowing how quickly it was about to change, this gave me security, self-respect, and the belief that I could achieve anything.

The long-awaited day arrived. It was my first day at Microsoft. I was expecting to meet my team members and start making the world a better place. The blood was rushing through my veins, and I was getting more and more excited with every second that passed. I started driving to the Microsoft building where the New Employee Orientation was going to be held. I had never seen so many buildings belonging to one company before. If you have ever visited Redmond, Washington, you know what I am talking about. People refer to it as a campus, but it is more like a city with over a hundred buildings. I could not believe my eyes. So many buildings, soccer fields, basketball and tennis courts, hiking paths, and amazing roads with beautiful trees of all different colors. It was an astonishing sight. It was like a dream come true.

Finally, I arrived at the orientation building. I walked around as if I was the creator of the future. I picked up my badge, which had the magical powers of unlocking every door to the wonderland. The only thing missing was a red carpet with someone holding a sign with my name to welcome me.

Orientation was being held at the Microsoft Visitor Center. I scanned my badge and entered the future. The first thing I noticed was a huge crowd. I thought they must be tourists. There were so many people, and I was wondering if there were always this many visitors every day. Then I saw that they were going in the same direction as me. Then I began noticing they all had badges. A crowd of over three hundred people with badges were all walking toward the orientation room. I was trying to make sense of what was happening. Could there be so many people starting on the same day as me?

After walking through the super modern-looking glass doors, I was inside a huge conference room with dozens of tables. I grabbed a seat, and people started introducing themselves. Some were from MIT, some were from Carnegie Mellon, and some were from Stanford. I started wondering if I would meet anyone from a university that was not world-famous. When it was my turn, I shared that I had attended Cukurova University. People asked where that was located. I told them it is the fourth-best university in Turkey. No one had ever heard of it…

That was the moment I realized that a new era was about to begin. I was not feeling so confident anymore. I was the engineer from the university no one had ever heard of. I started wondering how smart all of them must be to have graduated from these top universities. My self-image was shrinking while everyone else's was getting bigger.

There were new feelings making themselves known: not belonging and not being good enough. They were feelings I had never experienced in my life before. I was used to being one step ahead. I was used to already knowing what the teachers were presenting. At that moment, I met a new friend who has not left me alone to this day. I am sure you have met this friend. Most people know it as **Imposter Syndrome.**

I started feeling that it was only a matter of time before everyone looked at me and said, "Sorry for the trouble, but you do not belong here. The exit is that way." With every second that passed, I was getting more anxious. I could not even pay attention to the presenters. So much was going on in my head. As soon as we were on a break, I ran outside to get some fresh air. I cleared my mind and kept repeating to myself that I belonged there and that I could do this. I just needed to look confident and give it more time.

I was looking forward to saying goodbye to my new friend. I liked my old friends like self-confidence, trusting myself, and the feeling that I could do anything. This was a decade ago. Do you think my new friend has left me alone since then?

In the past decade, I have worked hard, launched many mission-critical services, received plenty of promotions, worked for three tech giants (Microsoft, Snapchat, and Facebook), built a few start-ups, led teams with dozens of people, and worked closely with hundreds of engineers. You would think by now that I would have learned to say goodbye to my friend, right? I learned to manage it, to not let it give me panic attacks, to not let it stop me from signing up for the most challenging projects. Unfortunately, I have never learned how to leave it behind entirely. I still struggle with the feeling of not belonging and not being good enough and the fear that one day, people will realize and say, "Sorry for the trouble, but you do not belong here. The exit is that way."

This was one of my best-kept secrets for years. If you were to ask my coworkers or friends, they would likely describe me as highly confident, brave, someone who enjoys jumping on risky projects, performs well, and knows what he is doing. I received a good chunk of positive feedback over the years. I tried hard to keep up this image. I never shared my fears, and I worked harder and harder to make sure no one would ever realize I did not belong there.

One day, I decided to share this with my close friends. After years of hiding behind my self-confident mask, I finally decided it was time to tear it off. You can imagine how hard this was for me. This is why I decided to start the book with this topic because if you are feeling the effects of **imposter syndrome**, you are not alone! Everyone has their masks up trying to look like they know what they are doing. We were taught to be self-promoting and to keep up a good image. What surprised me the most was that as soon as I shared the way I felt with my friends, most of them said that they feel exactly the same.

Yes, you read that right. Most of my friends were also suffering from **imposter syndrome**. Hearing this was the first time I could feel the effects of imposter syndrome starting to disappear. Instantly, I was feeling a lot more self-confident. And this time, it was not a mask. It was real self-confidence flowering inside me again. I started talking about how I felt with more peers. The more people I talked to, the more I heard that they felt the same way. These people were not only engineers. There were PMs, testers, data scientists, managers, and many more.

If you like reading self-development books or watching videos, you will hear a lot of people talking about **imposter syndrome**. It is a sneaky problem that affects most of us, and the more you fear it, the more control it has over you. I am not suggesting walking around letting everyone know that you do not believe in yourself. But if you have close friends, try opening up to them. If they are actually good friends, they will tell you that they have very similar feelings.

The good news is the more you learn how to fight it, the easier it becomes. I find it useful to keep a list of my achievements and talk to my close friends when I feel like an imposter. I focus heavily on changing my perspective to *It is fun not knowing what to do and figuring it out* as opposed to fearing it.

At the end of the day, we are getting paid to figure things out. If everything we do was already known, we could easily be replaced by a script.

Suggestion: Be aware we all feel like imposters sometimes, and do not let it get to you. It has nothing to do with how good you are, your level, experience, or anything else. If your job was that easy to do, you could easily be replaced by an automated script.

3. The Importance of Data to Convince Others

There have been numerous times I have found myself in situations where I believed in something, but others did not share my enthusiasm. I used to think this was due to people not believing in me, and I expected them to trust an idea simply because it was coming from me. Now take a second and turn the tables. Let's assume I came to you with a great idea. Would you trust me? What if it was about something you had no idea about?

Engineers are data-driven people. No one argues with data. When you have a great idea, that intuition comes from data. Believe it or not, the reason you think it is a good idea vs. a poor one is because of your experience and the data insights you have gained from it. If you have seen similar ideas be successful, you will be biased to think this will be too. But what if you do not have enough data to produce an intuition?

This is where the conflict starts with people. They are not against your idea; they just do not have enough data about it. The more data you can provide, the easier it will be for them to believe in it. Most engineers have dozens—if not hundreds—of tasks to catch up with. They are in a constant prioritization loop to catch up with everything that is high priority.

Let's assume we have a rental listing app. I believe that the location on a map of the listing is a significant signal and that we should show it as the first thing when a listing is opened. Currently, it is the last component on the page, and users need to scroll to the bottom of the page to see it. Would you agree? Disagree? What if I told you I have run a query and noticed that 70 percent of users scroll to the bottom of the listing and click "View Map"? Or what if I told you the exact query resulted in only two percent of the users scrolling down and clicking "View Map"? Now instead of arguing whether this is a good idea or not, we can argue how good of an idea it is. I do not think anyone would be against moving the location up on the page if you prove that 70 percent of the users are scrolling to view it. Of course, it is not always this easy to prove a point, and running a few queries takes extra time. But give it a try, and you will be surprised how much time it saves during discussions. It will make up the time lost running queries.

Suggestion: Always back up your ideas with data. The more you can provide, the easier it will be to convince others.

4. How Often Should You Interview?

During the first four years of my career, I never interviewed with another company. Partially because I was thrilled with my team and partially because of the fear of my manager finding out and asking me to leave. I have heard many friends say they feared similar things and were always postponing interviewing and only doing it when they were burnt out or started to hate their work. Or even worse, they started after their manager told them that they were unhappy with their performance and needed to improve (which is the polite version of *You might be fired soon*).

Other than the obvious one, the problem is that you will be way too stressed and in a hurry to find a new place. You will end up jumping into any opportunity you see. When was the last time you rushed into a decision and liked the outcome? I cannot think of many instances (except for the one lucky time you will hear about in the following paragraph). Everyone performs poorly under too much stress, and you do not have the luxury of holding out for something you love. If you were not performing at the level of your manager's expectations, how well do you think you will do when you start spending extra time preparing for and doing interviews while performing your job?

I was unlucky and lucky at the same time to learn this early on in my career. My uncle, who I love dearly, had a traffic accident and needed brain surgery. I was working on an important project, and my manager did not want me to take an extended leave. I had suggested working from Turkey, but due to legal reasons, that was not an option.

I was stuck between not visiting my uncle and risking my career. Risking your job might not seem like that big of a deal if you are a permanent resident or a citizen. But I was on a work visa, and being fired meant I had to pack up and leave the country if I could not find a new job right away. It was not an easy decision, but I am a strong believer that life will always present more opportunities, and family always comes first.

I decided to accept the risk and take the leave even though I was advised against it. I thought I might not have a job when I got back, so I had scheduled a few interviews. One of those interviews was with Snapchat, and the interview was right at Venice Beach. The office was facing the ocean. During the interview, I could not stop imagining working there and always hanging out at one of the most famous beaches in the world. After my third interview, the recruiter grabbed coffee with me, discussing what they were planning to offer me to test the waters. I did not have many other choices and was amazed by the beach, so I told him I would take it before even letting him speak. Of course, it is better not to look so excited that the recruiter knows you are not even going to negotiate. But circumstances were not in my favor.

Snapchat extended me an offer at the end of the interview, and I accepted it. I must have done something good in my life that karma paid me back. I got to enjoy the beach and the ocean every day and also joined Snapchat a few months before their IPO (initial public offering). While I thought I was taking a significant risk of ruining my career, it turned out to be one of the best decisions I have ever made. Not only it was a lot better package financially, but I have also learned so much working at a start-up. I have built many complex projects in very short times, thanks to the start-up environment.

Looking back, if my manager knew I was interviewing, one of two things would have happened. If he liked me, he would have worked harder to keep me around. If he did not want to keep me around, he would have given me more space and lightened my workload to ensure I did not stay around for too long. There is nothing worse for a manager than to keep an unhappy employee on the team. There might be some vengeful managers out there who might take action against you if you interview, but I have never met one to this day.

When I shared my experience with my friends, I learned that some of them interviewed regularly—some every six months, some every year, and some every couple years.

I cannot tell you how often you should be interviewing but can share why it is a good idea to do it more often than not.

1. You meet amazing people.
2. You learn how much your market value is.
3. You get backup options.
4. You get an opportunity to find something you love and can be excited about.
5. You stay sharp. Interviewing means you need to remember all different data structures, algorithms, etc.
6. You learn a lot. There is nothing better than high-stress, short-time situations with someone putting all their effort into getting data from you. You learn what you know well and what you do not.

Let's say you interviewed and got a few offers. Now you are in a more powerful position if you are interested in negotiating. You have a few options to choose from.

1. If you are happy in your current team and the pay is similar, you can simply turn the offer down.
2. If the offer is better but you are happy in your current team, you can ask your manager to see if they can match the offer. If you are an essential part of the team, management will be inclined to match the offer or at least give you some increase to convince you to stay.
3. If you are excited about the new opportunity, you can jump on it.
4. If you have multiple offers, you can also negotiate with the companies to see who is willing to give you the best package.

Suggestion: Interview as often as you can.

5. Learning to Say No and Brutal Prioritization

I am sure you have heard this before. You need to know when to say no. It is easy to agree with but not that easy to do. In the first year of my career, I said yes to everyone. Every time someone asked a favor, every time I was assigned a task or saw something was broken, I always felt I had to say yes. I slowly started focusing on build automation, testing, and writing scripts. At some point, there were twelve machines in my office running automation scripts. I truly enjoyed doing this, at least initially. The problem is you are assigned work based on your experience and reputation. I became the go-to guy for any scripts or automation needs. Even though building these pipelines was fun at first, I kept finding myself deep into running and watching scripts. There was a time I felt it was only a matter of time before I became a tester instead of a software engineer.

I started asking my manager for other types of assignments, but I was not good at low-level system design, scalability, or anything else available at the time. It was a considerable risk for him to trust me with a mission-critical assignment when I had not proven any of these skills. I had to learn to prioritize my tasks, start saying no to people more often, and start searching for projects to help with. When I started mostly focusing on high-impact tasks and spending the remaining time on other projects, I started learning a lot. It was not long after that I was assigned more challenges that corresponded to my interests.

It is always hard to say no. The thing that helps me the most is calculating the opportunity cost. It helps with both convincing yourself why you should say no as well as convincing others why no is a good answer. If you say, "No, I do not want to work on this," it will sound like you are picky and only want to work on things you enjoy. When you say, "I cannot work on this because I am working on X, Y, and Z," you feel better about turning someone down, and they feel better knowing you were willing to help but did not have any more space on your plate.

I use opportunity cost calculation a lot in my life to help manage my decisions. Do you want to say yes to a party invitation? What do you need to give up for that? A football game? You can probably watch it later anyway. Wedding anniversary? You probably would not want to leave your partner alone while enjoying a friend's party, right? Do you want to prioritize helping a coworker with their queries to prove a point? Of course. What if it came at the cost of missing your own deadline? Not so much, right?

We may not always think like this, but believe me, every decision you make carries many opportunities you have missed.

Suggestion: Always calculate the opportunity cost, prioritize brutally, and learn to say no to people with an explanation of why it is a no.

6. Never Say No!

I know… We have just talked about how to say **no** to people and the importance of it. And yet here I am asking you to never say **no**. What do I really mean? We talked about the importance of prioritization and working on high-priority tasks toward your goal and how you should say no to people if there is too much on your plate. So here is the trick: You have to learn to say no to people but never use the actual word **no** while doing so.

We are all busy people, and when we approach someone, we know that they are not our subordinates, and they do not have to say yes to what we are asking. But we also do not like hearing no. Especially if someone says no to a few different things, people start disliking them, taking it personally, or complaining to their managers. I have personally been both on the giving and receiving side of this feedback.

So how are we supposed to say no to people without actually saying it? By saying "Yes." Yeah, you have read that correctly. We can say yes to people instead of no with the same outcome. Why do you want to say no to people? Is it so you can chill in your office doing nothing or playing games? I hope not. It is likely because you are busy with something else, or you do not believe what you are asked is impactful enough. What if we change our sentence from "No, I do not have the time" to "Yes, I would love to work on this. But to set expectations, here are the tasks I need to do for the next month that are higher priority." In both sentences, you are letting someone know that there is no way you are working on this task this month. But in the first one, you are simply pushing back. In the best scenario, people will let it go; in the worst, they will go to your manager to force you or start saying behind your back that you push back all the time.

How does hearing the second example make you feel? Would you get mad at someone for being overworked? Or would you want to help them instead? Even though you are saying no in both sentences, in the second one, you are establishing trust, showing you are willing, and proving you are busy.

The example above is just a basic example that does not fit into every situation. But I think you got my point. Keeping a good backlog, prioritizing, and using it to your defense will be your best friend. You can even offer to sit with them to go over the priority of the tasks to see if you can get to it any earlier. This way, you can increase your visibility even more and maybe avoid some low-priority tasks they help you realize.

Suggestion: Keep a good backlog and always use it to your advantage to tell people why you cannot prioritize what they are asking instead of saying no. Stay away from saying no as much as possible.

7. Finding a Mentor

Finding a mentor is one of the best pieces of advice I have ever received. And it is one of the best pieces of advice I can give to anyone. Now think about everything you have been through, all the challenging problems you have faced, all the challenges you have solved, and all the stories you have listened to. We all carry so many years of experience with us all the time. When you find a mentor, you are getting a summary of all those years of experiences they have. They usually share the most important parts of these experiences. It is like reading the summary of a great book, but imagine that book is personalized for you. Can you imagine if you could read a great book and have it suggest solutions depending on your actual problems? Mentors are breathing books in the flesh that can provide valuable insights in small amounts of time.

I had quite a few mentors. Each of them taught me so much. I used to pick a design decision, study it as much as possible, and then go to one of my great mentors to discuss. They always had great ideas, and I would measure how long of a discussion we were able to have without them being able to find a blind spot. The better engineer I became, the longer I could keep the conversations going. There was no better pleasure than the days my mentors could not find a blind spot in my designs.

Another significant thing you can get from mentors is perspective. Perspective, if used right, can be the greatest asset you would ever need. For example:

- Let's say you are on the edge of getting promoted, and someone at your level joins the team. Would your perspective be that they are an enemy? Or a great asset? Do you think you can get them to help you excel? Or do you expect them to be backstabbing you at every opportunity?

- Let's say the project your team has been working on had an outage, and management is looking for who is responsible. Is your perspective this is a nightmare, and you should hide? Or is this an opportunity to claim responsibility and suggest a solution?
- What if half of your team quit their job? Where does that leave you? Do you think you will have more work than you can handle? Or is this an opportunity to prove yourself, hire new engineers to work with you, and possibly become a manager?

It is not always easy to have the best perspective. It is a workout that requires you to focus on a regular basis. We are all ego and fear-driven creatures. These are the most primitive parts of our brain. But the better perspective you start having, the better your attitude against these situations becomes. Your attitude makes the whole difference in any scenario between people walking away from you vs. everyone wanting to work with you.

Suggestion: Always look out for mentors. The more, the better. But pick them carefully to make sure their goals are aligned with yours.

8. Prototype Fast. First Working Prototype Always Wins

Coming up with ideas and proving they work is one of the hardest challenges. Every engineer goes through hundreds of ideas a day. And we all have jobs to do. In my experience, the most return for investment always comes from the prototypes to prove a concept that it will work. One time, we were testing out multiple people using our tool simultaneously. One of the engineers came up with a prototype that made fundamental operations to simulate multiple users using the tool. The tool's name became associated with the engineer who wrote it. Everyone called it Philip's test tool.

A handful of engineers were tasked to make that tool production quality. They worked on it for months, adding many features, making it bug-free, automatizing most of the testing steps, and writing unit tests for every part of the tool. So much investment by so many engineers. Guess who took the most credit. It was Philip's test tool at the end. Everyone called it Philip's test tool. Even today, I do not remember the names of the other engineers who worked on it. I do not remember what features they added to it, but I do remember the name Philip's test tool. People have short attention spans and cannot remember small details. They only remember important events. Prototyping something and proving it works is one of the best ways to achieve that.

This used to feel unfair to me. Philip spent only a few days on this tool, and multiple other engineers spent months. I kept feeling they should get more credit. But years have changed my perspective on this. There are millions if not billions of ideas floating around the world every day. But only a few of them become real and reach real users. Coming up with an idea, investing in it, making it work, and proving to people that it works is one of the most complex parts of software development or any other field. This is why the founders of any company make most of the money when the rest of the employees do all the work. If it were not for the founder, there would be no work to be done or any value to be produced.

Suggestion: Always turn your ideas into working prototypes as fast as possible. Everyone has great ideas, but a working prototype is a game-changer.

9. Visibility Is Everything

Let's talk about Jane. Jane is an engineer at company X; she is loved by her peers, and you can always find her at her desk, working on a few different projects. Jane wants to apply for a promotion. Do you think she should get it? No idea? Why can’t you say yes or no? Because you do not know her enough to have an opinion. Jane might be the best engineer the company ever had, or she might be the worst one. But you do not know because she is not visible to you. Until this second, you did not even know a person named Jane existed at company X.

You might be thinking this example does not make too much sense because people at your company know you, right? Do they really know you? How much do they actually know about you? An effortless way to test this is to turn the tables. First, I want you to write down the names of five of your teammates. Or five people from your broader division if your team is not big enough. Now try writing down their accomplishments and decide which one deserves the next promotion. Here is a tougher one: Which one do you think should be let go if the company was downsizing? This is a straightforward exercise, but it will help you realize how much you do not know.

You will likely keep remembering a name over and over again because they are good at being visible and advertising themselves. Now let's think outside of work for a bit. What is the brand of cereal you eat? What is the brand of your TV? Or your phone? Do you own anything from an unknown brand, or is everything from the most advertised brands? There is a reason advertisement is one of the most important parts of sales.

It is essential for engineers too. It would be best if you found ways to advertise yourself. I am not suggesting you purchase an ad on TV and run some videos of yourself (though why not?), but I am suggesting you need to learn ways to be more visible. There are an unlimited number of ways to accomplish this. Here are the main ones I try to do the most.

1. Sharing status updates. Sharing your progress on an internal blog post or team's file-sharing location on a regular basis helps people track what you are up to. It also is very beneficial when it is time for performance evaluation. I cannot explain how many times I would have forgotten what I have worked on if it was not for my notes.
2. Commenting on public posts. Most companies have some form of internal platform to post things. Be the first one to comment, even if it is just tagging the responsible individuals. The more people see your name, the more they register you as an authority.
3. Be active on design documents. Keep actionable comment numbers high.
4. Comment on people's code reviews. A wise friend once told me there is no better way to lead people than with code review. They are more concrete than verbal arguments, and it is effortless to apply the changes. It gives you a way to gain people's trust (given you provide actionable feedback that makes things better) and make sure no bad changes enter the codebase.
5. Avoid answering people's direct messages. This is the hardest one to do. I still sometimes struggle with it to this day. When you establish ownership, people come to you with questions. And you feel bad leaving them unanswered or asking them to make a post about it instead of just answering. The problem with this approach is that you are investing a lot of time to help one person. If they were to ask this in a post, and you answered there, you would be answering many more engineers who will have a similar question. The worst part of answering directly is that it is so easy to forget. There were days I would respond to hundreds of messages. Even though some people ended up writing good feedback on how I helped them, there was an insignificant return on all the efforts I put into answering their questions. The problem is no one will remember every chat thread or what you have helped them with. And simply saying, "Merih helped me" is unfortunately not enough for anyone to have enough data about

what you actually did. Please do not fall into the same trap. Try to ask people to make a post and save yourself the trouble of answering the same question multiple times while increasing your visibility.

Suggestion: Always find ways to increase your visibility. If no one knows you did something, no one will believe you have done anything important!

10. Let People Fail. Help Them Fall Slowly

When I was a junior engineer, my least favorite type of senior engineers were the ones designing every part of the system, telling me exactly what to do, watching every step, and trying to control everything to make sure nothing failed. I do not know if this sounds good to you, but it was a nightmare for me. People learn the most when they are doing something hands-on. My favorite type of senior engineers were the ones who put their trust in me and only vaguely described the problem while guiding me through the process to make sure my direction was set correctly but let me navigate the ship. It is always a pleasure to work with engineers like that. Not only do you learn a lot, but they also build a safety net around you in case you fall. And this takes less of their time so they can help a lot more engineers.

If there is a clear winner here, why do engineers turn into micro-managing monsters when they become seniors? Remember the definition of the least favorite type of senior engineers above? I developed the same tendency when I became a senior engineer. How come I turned into the monsters I was so afraid of as a junior engineer? It was a tough transition…

When you are a junior, you have a sense of control. You know what code you wrote; you know which parts to trust; you know which part needs more testing. You know where you got lazy because there was a party invitation waiting for you (I know. You are much better than me, and you would never leave your desk with some half-done code that you are going to fix tomorrow). When you become a senior engineer, suddenly, many unknowns are introduced into your life. You lead a few different moving pieces—if you are lucky. Usually, there are a few dozen, and you simply do not have enough data to trust the pieces. You spend your days and nights thinking about which part might blow up in your face and how long it will take to be demoted or fired when things start to go wrong…

When you are a junior, you give estimates based on how long you believe it will take you to do something, and if you choose, you can add padding to make sure there is some room for error. When you are a senior, you are given estimates with very few details. Did they add padding, or didn't they? Does this estimate require them to work 16 hours a day? What is their backup plan if they cannot perform to their estimate? You are responsible for every promise given to you, even if you just met the engineer for the first time that month.

There is a whole other area of unknowns called life. When giving estimates, most engineers focus on precisely how many work hours they think it will take. The problem is there are no estimates for breaking up with a partner, waking up with a sore throat, losing a loved one, jury duty assignment, etc. When these things happen, engineers can simply take time off. Everyone understands what they are going through, and no one expects them to be a robot. As I mentioned earlier, family always comes first, and I had to take time off a few times during my career to take care of my family. We are all humans. It is life! As a senior, you need to accommodate all of these unknowns.

When you write code, everyone can see it. Everyone knows what lines you wrote. The code review has your name. The commit has your name. There are even some tools that show you how many lines of code you wrote, how many commits a day, how many projects you worked on, and how many teams you helped. When coding is involved, it is so easy to generate dozens of metrics to track success. However, when you become a senior, it is hard to prove what you actually worked on. For example, suppose you have six exceptional junior engineers doing everything perfectly. Does that mean you should get promoted because the project is a great success, or should they fire you since they perform just as well without you being there? How do we measure if the seniors did any actual work since they are not coding and are busy with leading?

It takes time for any senior to adjust to life with so many unknowns. Some senior engineers are lucky to have great managers setting crystal clear expectations and mentoring them throughout the way. But some engineers are just thrown out there, not even knowing what they need to do. Unfortunately, most engineers who have recently become seniors try to do it all. They keep coding because that is what they know, and they try their best to help other engineers, trying to grab as much control as possible. They go over the estimates again and again, squeezing the engineers to make sure things are not going to start falling like dominoes.

You need to remind yourself of how you felt as a junior with a micromanaging monster. As time passes by, you will get better at reading people and estimating the estimates. You will be able to come up with your own estimates to judge juniors' estimates and how much time it would take another engineer to save the day if things start failing. You will learn to run meetings better, ask the right questions, and have more confidence in both yourself and your team. But the most critical thing you will learn is that you cannot save everything, you cannot win every war, and you cannot always be right. For me, this was the most critical lesson. Instead of trying to make everything perfect (perfectionism is a monster that we will talk about later), you learn to prioritize and track the most critical parts.

It is like learning how to drive. When you first learn how to drive a car, you are constantly turning the steering wheel left and right and right and left. You are trying to perfectly align yourself to the lane, motivated by your fear of crashing the car. But the better you become at driving, the less you touch the steering wheel. You know where the vehicle is headed; you just set the direction and know exactly where you will be in X amount of time.

Leading people is not that different. You cannot keep steering engineers left and right and right and left, hoping to stay perfectly aligned. You need to give them room and just set the direction, measure the speed, and know precisely when you will get to your destination. Of course, there will be roadblocks and accidents, maybe a truck blocking the whole highway, but you will get to your destination sooner or later by taking the detours.

The most crucial suggestion I have for engineers about to become seniors is to make sure you have a good manager who is willing to spend the extra time mentoring and setting crystal clear expectations for you.

If you are already a senior engineer, stop jerking the steering wheel around. Trust in yourself and your peers, make sure the direction is set, and let them drive. Let them fail; failure is nothing but a great learning opportunity. Just make sure you have built a safety net around them, so if they fail, neither of your careers are at risk. If you can show that you trust them and let them be themselves while having their back, it will flower into a long-lasting relationship, and those people will follow you wherever you go. I have friends who have been following their managers for the past decade.

Suggestion: Do not be a micromanager. Set a direction and let people do their best while preparing a safety net in case they fall.

11. The Tiebreaker—Reaching Consensus

We are humans driven by emotions. We all have an ego, a desire to be listened to and be right, and a passion for respect. You will find yourself in many design discussions over the years. Some will be very important; some will be small. But the satisfaction you get from another engineer's approval always feels great no matter how important or small the discussion is. There is instant gratification for your ego, making you feel great. Feels great to be on the winning side, doesn't it? Especially if the other engineer publicly admits they were wrong, and your opinion is better. Maybe they went ahead and told everyone you are a much better engineer.

Do you see where I am going with this? Initially, you were thinking about the time your suggestion was accepted and how good it felt. But to think there is a winning or losing side to any argument is a toxic perspective. It forces you to pick a side and sometimes push for things you do not even believe in to keep your ego happy. I have witnessed discussions about the smallest things lasting for hours and hours because nobody wanted to be on the losing side. I must admit I have had a couple of those discussions as well. Do you think I felt great when I finally got home? No! All I kept thinking about was why both of us had wasted so much time arguing about something so insignificant, something that did not change anything in the design, performance, or sustainability of the project.

We need to separate important decisions from small ones. If you believe something will make a system perform faster, reduce latency, reduce the development time, reduce the resources required (like the number of machines), or make one of the many more quantifiable metrics better, then I am always in for a great discussion. You learn a lot and teach a lot during an intelligent debate. I use the word *intelligent* here because everyone provides data points, suggests a change, and explains what it improves and how. Every engineer enjoys a great discussion like this. Usually, it is simple to come to an agreement when there is some quantifiable metric. I cannot think of any engineer fighting against an idea that will make the code perform 20 percent faster!

The problem starts when the discussion is about something non-quantifiable, and people’s opinions come from their intuition. We have talked about how engineers are data driven. They build their intuition around their past experiences with the data they have been exposed to. But what can we do when our intuition is the opposite of someone else’s? Do you argue until they get tired? Do you give up on your own opinion even though you think their suggestion makes absolutely zero sense? Both of your egos are involved in the decision-making. It is not even about what is best for the system anymore. It is more about who is going to win this time.

You might not believe that this happens, and many engineers think they were only trying to help, but egos are constantly involved in our lives. What is the best way to avoid egos getting involved and reaching a decision everyone respects? Using the consensus algorithms. Seriously, consensus algorithms have been studied heavily, and they are the backbones of IT infrastructure around the world. A server suddenly dies; who should start taking more traffic? A database server crashes while performing an operation; what is the latest status of that data? Do we return different results depending on which computer you talk to?

Of course not. We use pre-agreed consensus algorithms to let servers resolve their conflict with each other. Can we use the same algorithms ourselves? Why not? We have been ruled with democracy for centuries, which means the majority's decision is accepted. How can we apply this to our non-ending arguments with coworkers? Invite another one to make sure there is an odd number of you. Never start a discussion if you are evenly numbered. Always have a tiebreaker. By doing this, you simply remove egos from the picture. Everyone gets to pick the side they want to be on, and the majority wins. There was nothing against you or the way you believed. No one thinks less of you. You were not wrong. It just happens that there were more people in favor of the other solution this time. If you had the same discussion with another group, it would have been just as likely to end up with your suggestion.

This sounds simple, but it took me plenty of years before I learned to use this strategy more and more. You need to note this will not take you to the global optimum. Following the majority will not always help you make the right decisions. You might end up with horrible outcomes. But that is the beauty of it. When things go wrong, nobody is to be blamed because it was a majority decision, so everyone is responsible. And this is precisely why it prevents you from getting your ego involved. Because you still might be right. No one is saying you are wrong. Maybe everything will crash, and you will feel that it would not have happened if they had listened to you. But that will not stop the team from moving forward, and you will all learn from your mistakes, and next time, the majority will make better decisions.

Suggestion: Always have an odd number of people in decision-making so you can always have a tiebreaker.

12. The Importance of Allies in Design Discussions

Design discussions are one of the most essential parts of our jobs. They help us come up with different ideas for implementation, figure out all the blind spots, do a comparative analysis, make sure all requirements are listed, and pick the best solution. A good design discussion minimizes the risks while maximizing the gains for the project. The more eyes you can bring to a project, the better it will be.

At the beginning of my career, I would put a lot of effort into making my designs solid. I would spend hours and days designing the system end to end and ensuring I covered everything. I would spend a lot of time researching, coming up with a few different designs, picking the best one, adding flow charts and explanations, and documenting everything I had done in my design document. Finally, I would try to anticipate questions I would be asked and try to prepare for them.

Even with all this time investment, meetings would not go as smoothly as I had hoped. People had a lot of questions. Each of them would turn into another discussion. People would start wondering what would happen if we chose route A vs. B vs. C, then we would argue all these options. Even though I had already considered options A, B, and C, people still wanted to see for themselves. The worst part was not having those diagrams ready for different solutions. Now we were discussing solutions that everyone needed to follow with no visuals. Sometimes we all would agree on a solution, but sometimes we would split. And when there are two groups in favor of different solutions, it takes a lot of time to reach a consensus, if you can at all. You will need more follow-up meetings and discussions trying to make sure everyone is convinced.

The most important lesson for me was that I had the wrong expectations from the design discussions. I was running design discussions to show the team what I thought was the best solution and focused on convincing others why I believed so. The biggest problem with this approach is that you are not there to use other people's brains. You are there to convince them you have a good plan. And the feedback you receive feels like roadblocks you need to pass through rather than wonderful additions to your design.

The second lesson was that even though I knew why plan A was the best option, other people did not. There is tremendous value in writing down everything in your design document, especially the choices made and the alternatives. The more you explain what other solutions you have already considered and why you gave up on them, the more others will follow and agree.

Coming up with an end-to-end design without any eyes on it is not the best use of your time. A better approach is composing a design document with different options without too many details and sending it around for a small group of people to get feedback as early as possible and keep the design document an evolving collaborative document. This way, you will start using multiple brains and many years of experiences combined to make the right choices along the way.

The best part of doing iterative design with other engineers rather than yourself is that you pick the solution that has the agreement of all. Do not underestimate the power of allies during the design discussion. The worst outcome in a design discussion is a split group supporting different approaches. That wastes everyone's time and blocks any progress on the project. By doing an evolving design discussion with a smaller group, you gain allies who are already aligned with the solution suggested. Then, when you have the design discussion with the bigger group of engineers, you will have a lot less rejections and doubts when everyone sees there are a majority of engineers in the group who already believe in the design.

Suggestions:

1. Use design discussion to utilize as many people's brains as possible.
2. Make sure all the decisions and the alternatives are documented so everyone can follow them.
3. Make the design document an evolving document instead of an end-to-end design.
4. Have a small group of engineers and get them to be your allies before the final design discussion.

13. The Power of Meeting Summary Emails

One of the most time-consuming parts of our days are the meetings. They range from daily stand-ups to design discussions and from people scheduling meetings to ask questions to cross-team sync-ups. I cannot think of a time in my career when I did not have a few of these per week. Generally, more than a few. And I am sure you can think of a few meetings where you had no idea what was being talked about or why you were even there. It is easy to doze off during meetings; some walk away without any understanding of the decisions made, or even worse, with a different understanding of them.

What if I told you there is a golden trick to prevent you from dozing off and making sure everyone is on the same page after the meeting? Try writing down notes during the meeting and sending them to the invitees afterward. Why? This helps you better understand what was discussed and what decisions were made.

First, when you send the summary to people, you help them remember all the points discussed; secondly, you give them a chance to correct any misunderstandings. The best part: It brings everyone to a consensus. By providing an opportunity for people to correct misunderstandings, after the feedback, you will have reached a consensus that everyone agrees with.

Other than building consensus after the meetings, this also helps you create a searchable documentation library. Our brains are not evolved to remember every detail of every decision made. Sometimes the most important decisions slip out of our minds. Occasionally you will notice people arguing that they suggested the complete opposite of what they actually suggested. This is only human. We forget the details, and our brains fill in the gaps with the new knowledge we learned. I cannot express the importance of keeping track of all the decisions made and people's positions. This helps you search meeting notes in the future, prevent any arguments on people's positions, and if things go wrong, it gives you a shield to prove what actions were taken in order to avoid failures.

Failures are part of our life, and we learn so much from them. But one thing to avoid for sure is everyone claiming that they warned you, and you still went ahead with a wrong decision and failed the project. People do not even need to lie about their stance. They will truthfully forget what their original suggestion was and think they did the right thing as their brain will fill forgotten parts with the new information they have learned.

Suggestion: Get yourself in the habit of taking notes and sharing them with stakeholders after each meeting.

14. Align, Align, and Align Again

Alignment is one of the most critical things in a company. Of course, there are top-down and bottom-up driven companies and a mixture of both. But regardless of how decisions are made, the company leadership continuously tracks some metrics and bets on some. And it is essential to be aligned with the leadership.

Imagine you are a child and really hungry, but your mom is too busy cleaning the house. You keep crying because there is no food in the fridge and you are starving, and your mother complains about how dirty everything is. It sounds like a miserable situation for both, right? Exactly. When you are not aligned with the leadership, this is precisely what happens. You will put all your efforts into finding food, but they will keep complaining that the house is dirty. They will keep focusing on getting the house cleaned up while you feel you are starving, and no one seems to care.

The secret of success at any company is to figure out the long-term vision, the short-term direction, and if you are headed in the right way to achieve setting the right direction. Even though it is clear to see how big a deal this is, unfortunately, it is one of the hardest things to accomplish. You will not have too much visibility into what the management is planning. Sometimes it can be because you are not invited to the directional meetings; sometimes it can be because management is not sure either. Given that in big corporations, there are usually several layers of management between you and the CEO, it is likely each of them is trying to align with each other but not doing a perfect job. It is not uncommon for each layer of management to have different directions and goals.

I would suggest joining the all-hands meetings your company hosts to get a better understanding of leadership's long-term goals. There are usually multiple meetings at the company level, organization level, and team level. Try your best to join all. Keep asking questions to your manager to learn more about the long-term goals. Have meetings with skip-level managers to learn from them. Check your projects to make sure they are aligned with the long-term goals. Be a regular user of your product; keep an eye on what other projects are going around. Try to put yourself in the shoes of the management or even the CEO. If this was your company, where would you drive it to? The more you do this, the better you will get at it.

The more aligned you are with the leadership, the higher impact your projects are going to become, and the easier it will be to get more headcounts. It will be easier to get promoted, and all the doors along your path will be easier to open. Make sure you align, align, and align again.

Suggestion: Make sure both you and your team are aligned with the management, division, and company. Align, align, and align again as directions will change regularly. This should be a regular exercise for you.

15. The Power of Giving Credit

We have all evolved with an ego, a desire for appreciation, a desire to achieve something. But what is considered an achievement, really? How do we measure achievement? How do you know if the project you have implemented is a success? Not having any bugs in the first six months it was productionized? Or the code quality being high? Or is it being able to drive ambiguity? Or the number of times you avoided a fight and resolved all the conflicts peacefully? There are a ton of different ways to measure success if you are a manager. But as a person, the most important way is how much you are appreciated.

It does not matter if the project you built was huge or small. If a few coworkers open their eyes wide with excitement and tell you how amazing your project is, there is no better feeling than that. It makes you feel seen, appreciated, included, and like you belong to the team. But if I were to ask you how many times you could think of a similar scenario where everyone was appreciating you and saying it to your face, I bet you cannot think of many times. Or if I were to ask you when the last time was you showed your appreciation for others. There might have been a few projects you liked, but did you verbalize it enough to make sure the project owner felt appreciated?

Even though this is one of the most important feelings we seek on a regular basis and one of the most satisfying ones, we neither do it enough, nor is it done enough times to us. It is easy to see people's frustrations written all over their faces but difficult to see any kind of appreciation. There are multiple reasons behind this. Some people are shy; some people are jealous; some people think it will diminish their own project's success to talk highly about someone else's.

But I cannot stress enough that there is nothing better than giving people credit. The more you give credit, the more there is to give. The more you give it, the more you will receive. Recognizing other people's success will not diminish yours. Appreciating other people will only make them like working with you more. I am not talking about a fake "Oh, this is awesome" speech. I never liked doing anything artificial for gain. And your coworkers are likely smart, so ingenuine or fake compliments will not fool them. The best compliments are the real ones.

Start thinking about all the tools you use, or all the libraries, or anyone who helped you in your current project. How much time have you saved thanks to them? How much knowledge have you gained from them? Did they cover one of your blind spots? Did they find a critical bug that would have crashed your service? Unfortunately, sometimes we are so busy focusing on what to achieve we forget to appreciate the help we received. Now spend a few minutes thinking about these, and if you can come up with something, show appreciation. If you have used a library that saved you a few days, tell the owner the next time you see them.

Is there a better way to show appreciation than directly telling people? Yes, try telling it to others. If you like a library, share that with the rest of the team. If a tester finds a critical bug, tell it to the people you work with. Advertise them. Tell them what a lifesaver they are and how others should ask for their help before launching any service. People always talk to each other. And do not worry: Any compliment you start spreading around will reach the owner.

Why is this a better way of showing gratitude? Because you cannot fake telling multiple people how much you appreciate something unless you do appreciate it indeed. When people hear you were complimenting them when they were not in the room, they will take it more seriously and feel much better about it. And this carries the side benefit of showing everyone how much you like giving credit. Which means if they work with you and help you, likely you will be returning the favor by sharing good things about them. It is a great way to show people you are not obsessed with your own success and actually can recognize other people's as well and spread the love.

Suggestion: Find something useful and start giving credit to the owner either by directly telling them or spreading the love in the meetings you join. The more credit you give, the better.

16. Sharing the Responsibilities

Engineers are busy people. At any given time, we tend to have tasks we cannot finish for months, topped with meetings and people running to us with questions, and as if all of these were not enough, we also have regular responsibilities such as increasing the code quality, fixing bugs, on-call rotations, checking performance regressions, and the list keeps going on and on and on.

When something breaks, and you need to pay attention to that on top of all these other things you need to catch up on, it does not feel great. What are your options? You can either put in an extra hour to catch up to the extra tasks and give up on your personal life for a bit, or you can push back and ask someone else to take on the task. Unless you work with people who do not have too much to do (please let me know so I can join your team), they will not be happy to take over either. Then they have a few options: they can either give up time from their personal life, push back, or take it over and find yet another owner to work on it, and this goes on…

What if I told you there is a better way to deal with these kinds of situations? I suggest the first time this happens you accept it and happily work on it. The second time, take a note of the repetition and continue to do so. When it happens a third time, you are starting to see a pattern. It might be the right time to start thinking of a process. If it has happened to you three or more times, it is likely that it is also happening to other people. When you design a process around it, your goal should be to minimize the burden on any particular individual and promote a process to protect everyone.

Let's use on-call tasks as a case study. Some weeks of being on-call are going to be quiet and easy; some weeks will be hell. How can we deal with all the tasks that rain down to the on-call? Here are five different approaches.

A. The on-call engineer for the week when the task was created has to own the task and make sure it is resolved even if they need to keep working after their week of on-call.

B. The on-call engineer for the week resolves as many as they can, but the remaining ones are transferred to the next on-call.

C. There can be an on-call task folder that all the tasks are assigned to, and whoever has time can pick them up.

D. Similar to C, there can be a folder, but the tasks get assigned to engineers during sprint planning.

E. We can combine C and D and expect the on-call engineer to do their best during the week. The team prioritizes whatever is left over during the sprint and assigns it to engineers. In critical circumstances, prioritization and distribution can be done mid-week as well.

These are just five examples of how to distribute the tasks to engineers. There can be many more ways to achieve results, but let's focus on these five to understand the importance of the processes.

If we go with option A, it is the simplest to implement, but it will be the most frustrating one, especially for engineers who had an unlucky week. If the on-call load is light in general, we can go with A, but when problems start hitting the fan, this will lead to unhappy engineers quite quickly.

Option B has a bit of an improvement so no one engineer is left with two months of tasks from an unlucky week. However, if there are too many tasks, it is not a good start to the rotation with a giant list, and in time it might teach people that it is okay to ignore the tasks.

Option C starts utilizing everyone and promotes being proactive rather than forcing people to work on the tasks. However, what Option C is lacking is accountability. If the engineers already have too many things to work on, the on-call tasks will keep getting deprioritized.

Option D still promotes being proactive but also puts the tasks into prioritization during the sprint. This is a good solution, so you do not get assigned enough tasks to keep you busy for the whole sprint and then end up with on-call tasks. During sprint planning, you will be assigned fewer non-on-call tasks to accommodate the on-call tasks.

And finally, option E combines all of the above. The on-call engineer is in the driver's seat but not responsible for everything. They try their best to mitigate and prioritize things, but then you distribute the remaining work to the team and make sure they do not have too many tasks during the sprint. This way, no one engineer feels isolated or slammed with too much work while on-call tasks are still being taken care of. Also, this teaches people to have good habits as likely the bugs will be visible to everyone and get assigned to the project owner who introduced them.

As I mentioned, if you keep staying quiet and doing everything assigned to you, you will burn yourself out. And management will never know they existed in the first place. If you start complaining, you will push people away and look like an engineer who does not want to do what is needed. But if you can recognize patterns and suggest processes like above, you will save yourself from ending up with too many tasks while protecting your team members from feeling the same way. In addition, you will increase your visibility while doing the tasks and become the savior of the long nights.

Suggestion: Pay attention to patterns and come up with processes to solve problems instead of getting frustrated.

17. Taking Responsibility & Ownership

It was one of those days where everything went down. Our service was crashing severely, and there was no solution in sight. Everyone was busy trying to figure out what went wrong. The finger-pointing and blaming, with everyone stressed and wishing it was not their change took down the whole network. After a while, we figured out what had caused the problem and took some time to restore things. After running more sanity checks and making sure everything was working, we could finally breathe normally. To this day, I remember the amount of stress everyone was going through and how everyone was trying to find a place to hide from management. No one knew how management would react to this because we had never had such a terrible outage ever. It was the kind where everyone started thinking that the responsible person would probably end up being fired.

In an environment where everyone was scared, one of the senior engineers walked into the manager's office and apologized. I was surprised because it had nothing to do with him. It was not due to any of the code he wrote or any action he had taken, and yet there he was apologizing and saying, "If I had done my job better, this would not have happened. I should have thought of this and put some checks in place to prevent it"! He did not stop with our manager but also said the same things to the higher management when we did the postmortem review (where we talk about what happened and what went wrong). It was amazing to watch both the senior engineer and our manager explaining everything that happened technically without pointing fingers. They kept blaming the process and the design of the overall system allowing this to happen rather than sharing what the engineer responsible had done that had caused it.

What do you think happened next? Do you think they fired the senior engineer? Do you think they yelled at him or threatened him? Nope! Management was amazed by the ownership and responsibility the engineer showed. To claim responsibility for something he had nothing to do with and promise it would not happen again was a big, bold move that amazed everyone in the room.

Put yourself in the shoes of the management and think about which one you would prefer: everyone lying and denying responsibility or someone with a great sense of responsibility explaining what happened and promising it would not happen again? Would you care if he had caused the problem, or would you want him to be in charge of the project?

Of course, this is a risky action. You are accepting responsibility, and if things keep going wrong, you are on the hook. But this is exactly what makes leaders separate from their followers. Leaders step up, take responsibility for all the risks that carries, and try their best to avoid it ever happening again.

I had always heard great things about this same engineer from other people, and everyone was trying to work with him. After seeing all of this happening, I fully understood why. He was like the big brother/sister who has your back, always giving credit to other people, and yet he was there standing in front of everyone when things started going wrong. These are fantastic leadership skills that unfortunately, not many people possess. I have witnessed some managers trying to do everything in their power to blame the engineers for the project's unsuccessful outcomes and painting the picture so well that it appeared they could not have done anything better.

I am not suggesting you jump in front of everyone every time and accept responsibility for every mistake. What I am suggesting is not being scared of being blamed or yelled at and establishing certain ownership and responsibility areas. It would be best if you did this even before you were asked to own something. Start thinking about some of the areas that interest you and start owning them. Take some responsibility. Think about what can go wrong and start planning around how to avoid it. And if something is broken, shield people, accept the responsibility, apologize, and show everyone that you own it. You will get a lot more opportunities to increase your ownership areas and help people this way.

Suggestion: Pick an area, start acting like an owner, take responsibility, and shield people. No better way to become an owner than starting to act like an owner.

18. How to Disarm Assholes

This one is a bit controversial, and I hope you never need to apply these tricks. But unfortunately, it is not uncommon to bump into assholes from time to time. They come in plenty of forms. Some like to micromanage your day, some like to feel they are better than you, some assume you have to do whatever they ask, some are rude, some would not let you commit anything without a few dozen comments (I am not talking about the excellent constructive comments that you should always be asking for), some yell at you, and some are just passive-aggressive all the time. There are many forms of assholes out there. Some have the personality for it, and some are just having a bad day, but it is easier to be an asshole than to be vulnerable.

You will face all kinds of people from all different cultures with different personality traits in your career. It is a dream to expect everyone to be similar to you and to get along well. And it is not even a good dream. I feel it would be boring to have a great time with everyone every day; no challenge there. And if you think you can go to HR every time someone is rude to you, you will soon learn HR is mainly there to protect the company from possible lawsuits.

This is not to say HR is never helpful. There are some great HR people out there who will try their best to help you. Still, their first goal will always be to protect the company and resolve the situation with minimum damage. They are not obligated to listen to your preferences. You can think of HR as your parents. They will want to protect you, but the other side is still part of the family, so they cannot alienate them. They will not punish the other side (unless the other side is doing something illegal or entirely against company policies). Try to follow the suggestions in this chapter before you consider going to HR.

First, try to see if you can change your perspective. You might think someone is being an asshole, but they might just be having a terrible day. Try to look at the events from their side. Have they really targeted you? Or were you just talking to them at the wrong moment? Nothing justifies someone being rude to you, but I am sure you can think of times when you were not that nice to people either. If you can remember they are a person first and a coworker later, then you can sympathize and see their point and be nicer to them. I have had many encounters with people who were rude to me at certain times. I built great friendships with quite a few of those, and it turns out they were great people having a rough time. We are all under a lot of stress. Everyone has deadlines; everyone has family issues; everyone has a personal life. If you can be there for them when they are not treating you up to your standards, you will quickly realize they are becoming friendlier and nicer to you.

If sympathizing does not work for some reason or you cannot find a way in your heart to treat them nicer, then try to figure out conflict points. Are you having a hard time talking to them in person? Try to communicate more with messages or emails. Are they short and rude in their messages? Try to talk to them in person. Are you always ending up in conflict with them during discussions? Try applying the tiebreaker trick and bring other engineers to the discussion to avoid being even numbered. An odd number of people have the advantage of creating a majority in any given decision.

Sometimes all you need to do is take a break. Take a coffee or tea break. Walk around a little bit. Let your emotions calm down. And then try to look objectively at the situation without your emotions. This advice is one of the hardest things to achieve. When your emotions are running hot, you might think they are attacking you or disrespecting you personally. But most of the time, they are just trying to get their point across. I have worked in giant corporations with many cultures, and while some cultures prefer being super nice, others prefer being direct. Figure out if they are doing something to you or if that is their personality. Coming from Turkey, I have suffered from this many times. In our culture, the more direct you are, the better. We always say if you already have a brain, why should I use mine to guess what you are thinking? Why can’t you say it to me directly? I got in trouble so many times early in my career because I would say things to people's faces, and they did not appreciate the directness. I got feedback to talk to the managers so they could relay the message in a politically correct way by removing emotions.

If none of the suggestions above work for you, start documenting the conflicts. Having written records is usually the best way to resolve them since you can refer to them later. Try to keep your discussions in comments, messages, or emails. Try finding people who seem to get along well with both of you, show them the conflicts, and ask their opinions. Nothing is better than a fresh perspective from someone who knows both of you. I find this one particularly useful. Cultural or personal differences sometimes create significant boundaries, and it is hard to see what is going on when you are emotional. When you show these records to someone who knows them well, they can point to what they think went wrong. Even though you might think the other person is the asshole in this picture, you might be surprised to hear that maybe some of your sentences did not come across well to the receiver. But, again, you can only see things from your perspective. There have been so many times in my career when I thought I was being nice, but other people pointed out how my communication had sounded and how I could have worded it better.

If none of the above suggestions have worked so far, you might be dealing with someone who does not have good intentions. At this point, you need to start thinking of minimizing the damage. If you have tried everything, and the problems are not being resolved, you need to start taking other actions. Try to find other projects that do not involve working with them. Invest your time more and more into other projects, and slowly depart from the project that is drowning your emotions.

If finding another project is not possible in the short term, you can try to ignore them. Unfortunately, it is not that easy to ignore someone (maybe with COVID it is a bit easier, but I am mainly speaking about when we were in the office and when we will be back). I cannot speak to all corporations, but the ones I have worked for require you to succeed at your projects. You do not have to take orders from anyone. It does not matter if you are doing what they are telling you as long as what you are working on is successful and recognized by others. Even your manager does not have the full power to tell you what to do most of the time; they only try to motivate and inspire you.

There are usually performance evaluations; what you have accomplished matters more than if you have followed everything you have been told to do. Start coming up with ways to contribute to the project and minimize the communication with the concerned person. This requires some extra effort on your end to keep your visibility high and make sure everyone recognizes your achievements. Especially if this person is someone you need to work closely with, losing their good feedback for your performance evaluation will be a big deal. Therefore, you need to make sure you are covering that from somewhere else.

At this point, you also need to consider talking to your manager about the conflict to see if they can help you find a better project. Some people suggest going to the manager as an early step. If you have a good manager you are close with, that might be a good idea. But you only have a limited number of get out of jail free cards. My uncle used to say, "Do not walk in muddy streets. No matter what, you will splash yourself with mud." It does not matter if you were 100 percent right; your manager will remember you had a conflict. And if you have a second or a third conflict, you will be the conflict person. This took me a few years to admit, but there are people out there who can work with literally anyone. It took years of practice to build tougher skin, but today I can handle people who treat me a hundred times worse than when I started working. Put yourself in your manager's shoes. You and the other four engineers are working with the person you feel is an asshole, but everyone seems happy. It will not look good for you.

And if everything you tried fails, it is time to look for a new team or a new company. Do not underestimate your emotions. I do agree you should grow a thicker skin and be able to work with all kinds of people. But until you can do that, you will be emotionally drained, performing worse and burning out. If you do not deal with these feelings and let them linger, you will not even be at a stage to perform a good interview. It is essential to focus on the long game, not the short one. You will have a career for a few decades, and you do not want to end up hating your job and life during your journey.

Suggestions:

1. Try your best to change your perspective and see things from their side.
2. Change the way you communicate with them. If in-person is a problem, utilize messages/emails.
3. Take a break, let your emotions calm down, and look at the conflict objectively.
4. Start documenting conflicts so you can ask someone who gets along with them to give you advice.
5. Start picking projects that do not involve working with them.
6. Ignore them at all costs.
7. Talk to your manager.
8. Change your team.
9. Change your company.

19. Adaptability

This has been one of my favorite words since I heard it for the first time: Adaptability!

Adaptability is a virtue that unlocks a world of opportunities.

Change is the only constant in our lives. Change is unavoidable. Change is everywhere, every minute, everything. Even though change is the only way to move forward, it is one of the hardest things to handle. Change is uncomfortable; change is unfamiliar; change is scary. It is easier to create a bubble for ourselves and act like nothing is changing in it. Regardless of whether this is true or not, we make ourselves believe things are still the same.

What happens when we are faced with change? What happens if something pushes us out of our comfort bubble? There used to be so many times I felt panic in my early career. It did not matter how big or little the change was; I could feel the fear ships sailing toward my heart.

What would you feel if any of the following happened?

- You have been assigned a new project.
- Your project design has been shut down.
- Your project for the past four months got deprioritized.
- You have just found out another team is doing a similar project to yours.
- A couple of engineers are quitting the team.
- Half of the engineers in the team are quitting.
- Your team is going to be reorg-ed.
- Your division is going to be reorg-ed.

- Your whole company is going through a major reorg.
- There will be layoffs happening in your team.
- There will be layoffs happening in your company.
- There will be layoffs of thousands of people.
- The company's stock is taking a dive and losing 20 percent.
- The company's stock is taking a dive and losing 80 percent.
- You are convinced the company is going bankrupt.

You will be going through all of these and much more in your career. When things like this happened early in my career, I used to panic and start thinking about the worst possible outcome to make sure I was prepared. Looking back, I missed many opportunities because I was acting on fear instead of excitement. You can hear from many different channels that you should see things as a **Challenge** and not a **Problem**, and I cannot stress enough how vital this mindset is. I had heard similar phrases for a few years before it sank into my head, and I started following them.

Why is it so hard to be excited rather than fearful? Why is it easier to see problems than challenges? Because your survival depends on it! If you are a few seconds late to a fun event or an exciting experience or miss a challenge, you will still be alive. But if you miss an existential threat, you might not be alive in a few seconds. This is because the amygdala controls fear and anxiety, one of the most primitive but important parts of your brain for survival. The amygdala is the difference between life and death. It is what we owe our lives to. It is what kept all our ancestors alive. It is the first respondent in any situation, and it keeps you breathing.

If the amygdala is so vital for our existence, why is it a problem? Because we no longer live in primitive times filled with poisonous spiders and snakes where we need to hunt deer, climb trees to collect fruit, and run away from predators. But our amygdala cannot separate the fear of work-related stress from the fear of being eaten by a predator. If we feel there is a threat, the amygdala kicks in and tries to make sure it saves our lives. What can we do to separate life-threatening dangers from others? We can start using our prefrontal cortex. The prefrontal cortex is what separates us from animals. By using our prefrontal cortex, we can decide how we want to act on the stimulants. There are many different ways to achieve this, and it takes a lot of practice to change how we respond to fear. It is an involved topic, and I definitely cannot teach you how to do that in this short space. Many excellent books teach many different techniques like breathing, mind mapping, using emotions, etc. I suggest anyone reading this book research how our brains process fear and how to use the prefrontal cortex more.

For me, the most helpful way to manage my fears has been breathing techniques, using the words *opportunity* and *challenge* again and again and again in my brain, and forcing myself to think about the good outcome I can get out of this situation. When it is not a simple one, then I try to write down the options. When you write things down, there is an almost immediate relaxation your brain feels. I find it a great way to offload what is on my brain to a piece of paper, prioritize visually, and then follow up. By following these techniques, it takes a few seconds to a few minutes to relax and start feeling excited instead of fearful.

Why is being excited about a situation so important? Because it opens you up instead of closing you down. When fearful, we tend to be protective, pessimistic, ready to fight or flee, or jump on the first solution we see. When we are excited, we start having fun with what we are doing. We get more engaged; we become positive. We start radiating confidence and make those surrounding us feel a lot better. I had a few colleagues who managed to be continuously excited and ready for a challenge, and there was nothing better than talking to them when something was going on. Seeing how calm and excited they were always made me calm and excited too. This is a great skill to lead others to make them feel everything is going to be okay.

Do not worry if it is not easy to achieve. It is always a process. But you know what they say: "Fake it until you make it." The more you study about staying calm and getting excited about things, the more you will find it is getting easier to do. Now let's go over our list to see how fear and excitement differ from each other.

- You have been assigned a new project.
- Your project design has been shut down.
- Your project for the past four months got deprioritized.
    - **Fear**:
      I do not know anything, so many things to learn, so many people to meet, what if they are not nice or I do not like them?
    - **Excitement**:
      Great opportunity to learn new things, meet new people. I wonder if I will become close friends with some of these people. I wonder what technologies they know that they can teach me. This is a great shot to prove myself, start leading this project, and get promoted. Yes!

- You have just found out another team is doing a similar project to yours.
    - **Fear**:
      My project will be shut down; I wasted so much time on this, now I need to find a new project; performance evaluations are coming up.
    - **Excitement**:
      Nice! I can stop asking my manager to assign more engineers to help with the project. I can utilize the other team. Let's see what parts I have built already and what parts they have built. We need to figure out a good way to merge these to prevent wasting time and maximize our investment. I cannot believe I picked such a good project; with the additional help, there is so much we can do!

- A couple of engineers are quitting the team.
  - **Fear**:
    Oh no. All of their work is going to land on me. So many more responsibilities. I don't even know most of the code they have written. This is going to be a nightmare.
  - **Excitement**:
    I am sad to lose them, but now I can take over the high-impact projects they were driving. Maybe this will give me a reason to deprioritize low-impact projects since we do not have enough engineers to finish everything. I can reach out to the people I love working with to see if I can convince them to join the team for the opening headcounts.

- Half of the engineers in the team are quitting.
  - **Fear**:
    Same as above plus you might **Fear**: What if reorgs happen or the team gets shut down? I might miss all the deadlines.
  - **Excitement**:
    Same as above plus you might think: I am a lot more senior in the team (assuming senior people are the ones leaving for a change as it is unlikely for a bunch of new joiners to decide to quit all together) and I can start managing up and help my manager with hiring and this might be the chance I have been looking for to become a manager.

- Your team is going to be reorg-ed.
- Your division is going to be reorg-ed.
- The whole company is going through a major reorg.
  - **Fear**:
    Too many new people to meet with. I do not know what I will be working on. I need to spend so much time ramping up. I might

    pick up a horrible on-call rotation caused by bad code others have written.

    - **Excitement**:
      I will get to own new parts of the system. I will be learning so much more, meeting so many new people. I can use my knowledge for the old project to help the new team working on it while having the chance to build something new that could work with the old system and benefit from a fresh page with so much knowledge under my belt.

- There will be layoffs happening in your team.
- There will be layoffs happening in your company.
- There will be layoffs of thousands of people.

    - **Fear**:
      I will be fired. My friends are going to be fired.
    - **Excitement**:
      It is better to see if the company values my project. If I am not laid off, so that means they believe in this project. If I am laid off, I will receive a layoff package covering a few months, and I can find another job by next week anyway. I will get paid double for some time.

- The company's stock is taking a dive and losing 20 percent.
- The company's stock is taking a dive and losing 80 percent.
- You are convinced the company is going bankrupt.

    - **Fear**:
      My income is a lot lower; everyone is getting better offers and quitting their jobs; I will be unemployed soon, oh no…
    - **Excitement**:
      This is a great time to negotiate with my manager and ask for some retention bonus. If it gets any worse, I can always find another job. If company stock bounces back, my retention bonus

> will be multiplied. I need to work harder to get promoted while everyone is panicking.

I bet you think my examples are exaggerated, and there is no way anyone can be this optimistic, or you do not believe things would turn around this nicely. Are they even real? I try to stay away from using real examples and base most of my points loosely on the facts, but let me share with you a couple of stock prices of Snapchat below, and I will leave it up to you to interpret.

On 12/21/2018, SNAP stock went as low as **$4.82**
On 9/24/2021, SNAP stock went as high as **$83.34**

That is 17.29 times! Of course, I have picked the lowest and the highest the Snap stocks have ever been to prove my point. I do not expect anyone to be lucky enough to join or get bonuses when it is the lowest and time it so well as to sell it when it is the highest. But can you imagine keeping your head on straight and working hard even in the darkest days? When everyone around you is having a hard time, and you start fearing the worst, it is not easy to keep your head straight. But if you can achieve it, things will always get better. Maybe not 17-fold as it did for Snap, but things always have a tendency to get better. And worst case, you can always find a new job. But imagine for a second if you kept working hard and got promoted during a crisis and got new stock bonuses when they were at an all-time low.

I have seen many examples of people who turned worrying situations into the best lottery tickets I have ever seen. Some multiplied their incomes; some got promoted to senior levels. And even if there is no financial gain from this, people who do not jump ship on the first storm tend to gain respect from leadership and can have a much bigger impact in the future.

Adaptability is a great strength. It took me many mistakes and a lot of lost opportunities to learn and appreciate this. Do your best not to let your amygdala cloud your judgment. Take a deep breath, change your perspective, and start counting the possible opportunities and good outcomes. Even if you do not believe in them, keep counting, keep exercising, and in time you will notice it is becoming easier and easier to be more adaptable and adapt to all the changes in your life a lot faster.

Suggestions: Master adaptability. No matter what the world throws at you, keep your head on straight and keep going forward with full throttle.

20. The Importance of Tools and Patterns

Have you ever wondered why humans are at the top of the food chain? Why do we not have any natural predators? How come when all the animals need to live their life in danger all the time, we can sleep soundly in our homes knowing nothing can get to us?

Lions can run a lot faster than us and are great hunters. Birds can fly. Fishes can swim a lot faster than us and do not even need to go above the water to breathe. Bears are better at running, climbing, and hunting than us. If we were up against any of the species out in the wild, our chances would be limited. Compared to predator species, we are weak, slow, and do not even have claws. But somehow, we are more dangerous than all other species combined.

Because we have the ability to build and use tools!

This is what separates us from the rest of the creatures in this world. Even though considering side-by-side comparison of body features, we would lose every time, we can design tools that can make us better swimmers, faster runners, better at flying, better at killing. No bird can pass the speed of our planes, not many animals can race with our cars, and no animal can kill from miles away with a click of a button. The ability to build tools let us become the most advanced creature out there and it let us overcome all our predators in history.

Tools are still as vital as they have ever been, and we spent quite a big chunk of our lives learning them. Reading, writing, driving a car, shopping online, and many more. Our whole education system is designed to teach us as many tools as possible. But when we stop thinking about this explicitly, we do not realize it. That is why I wanted to start with a bit of a history of the evolution of tools. Even when you think you have not done anything hard for the day, you have probably used a few dozen complicated tools that set you aside from animals.

In the same way that tools set us apart from animals, they set us apart from others. The more tools you know, the better and faster you can perform something. Now let's assume we are working on a project for a hospital. I hand you a text file with a couple thousand lines of patient information. Every line has one patient's information separated by a special character. It has their name, address, birth dates, diagnosis, medication, and the list goes on. And we are told that legally, we should not use any sensitive information that can help someone personally identify the patients. How do we do that?

Do you open the file in a text editor and start deleting their names, birth dates, and addresses? You start with line one, drag your mouse around, choose the first name with your cursor, and click delete on your keyboard. Then you scroll and select their address, mark it with your cursor, click delete, then you scroll and select their birth date, click delete. Great job, you have finished one patient. Now the second one, the third one, the fourth one …

This sounds like it will take you a few hours, if not a few days, right? I am sure you started getting anxious just reading and thinking, *What if he keeps writing the fifth one, sixth one, seventh one…* This kind of work is tedious, susceptible to mistakes, takes too much time, and is not rewarding. No one is going to think you are a great engineer if you have achieved deleting some text from a text file. I could have asked my little cousin to do this. What do you do when you are assigned work like this? Do you do it? Do you ask your little sister/brother/cousin to do it for you?

Whenever I have been assigned a tedious task like this in my career, I found myself researching. Researching tools that can do my job instead of me. Better than me, faster than me, and with a lot fewer mistakes than me. Sometimes learning a tool has taken me a lot longer than it would have taken to do the task. But we are in this game for life. Unless you are planning to retire in a year or two, likely you will need to do something similar at least a few more times.

Let's go back to our example. Have you ever heard of "Regex"? It stands for "regular expression," and it is a tool for writing patterns that can be matched to the text. Like if your records were something similar to this:

1987/05/21, John Doe, Male, Common Cold, Antibiotic

You could have simply written a regex for "^.*?,.*?," (Circumflex, dot, star, question mark, comma, dot, star, question mark, comma, space) and deleted the birth date and name from all the records in less than a few seconds. Without turning this topic into a regex course, here is what it does behind the scenes:

^ (Circumflex) -> Starts matching from the beginning of the line
.*?, (Dot, star, question mark, comma)-> Matches all the characters until the first comma found

And we repeated the same operator so we could match both the birth date, which ends at the first comma found and the patient's name, which ends at the second comma found. Of course, learning regex might take a few hours to a few days, depending on how advanced you want to become. Still, once you learn it, any kind of repetitive text operation can be done in seconds.

This is only one example of excellent tools out there, and there are so many to choose from. From build automation to writing scripts, from version history tools to text operations, from auto-filling code snippets to debuggers, there are so many tools to learn that can reduce the amount of time you spend on daily tasks. The next time you are assigned a task, start thinking about what tool could do this better than you. Or even better, without being assigned a task, keep researching tools online that you could learn. There are so many great websites I follow that have a giant list of things to learn, and I try to get as many as I can under my belt.

Suggestions: Always consider whether there is a tool that can do your task better, faster, and with fewer mistakes than you. Even better, keep learning tools before needing them for a task.

21. Open Source Internally

In a world where there is too much to do and not enough engineers to do it, you will always be surrounded by tight deadlines and not enough investment for your projects. You might have been asking your manager for more engineers to help you and getting turned down. What do you do if you cannot get more engineers assigned to your project? Do you work overtime to catch up with every task? Do you start pushing the deadlines back? Do you keep arguing with people and explaining why their requests will take a few months to get to?

When you are short on engineers and requests start piling up, no one wins. You will lose because of too much work to do; others will lose because you will become the bottleneck and push every request back by a few weeks to a few months. You will lose again because frustrated engineers will start giving bad feedback about you or fund their own service so they can implement what is more critical for them instead of needing to convince you every time. No one wins in this situation. We will talk about a few tricks to convince the leadership to invest more engineers in later topics. Still, the open sourcing trick is usually my first approach before going to the leadership.

If you have ever used open-source libraries, then the things I will mention in this chapter will sound familiar. How do open-source libraries usually work?

1. Code something useful.
2. Document how it works and API usage.
3. Document how to contribute.
4. Upload the code to a repo.
5. Preferably add documentation for each component. The more documentation, the better it is.

6. Preferably add unit tests. The more tested, the better it is.
7. Preferably add functional tests.
8. Keep an excellent backlog to prioritize and track tasks.
9. Advertise the project and get more engineers invested.
10. Drive the long-term vision and assign tasks to appropriate engineers.
11. Stay on top of code reviews to keep the code quality high.
12. Keep repeating documenting, adding tests, prioritizing, etc.

This is not the most exhaustive list, but successful open-source projects look somewhere along this line. What prevents you from doing precisely the same thing internally? Nothing! I am not suggesting open sourcing your code to the world. Though if your company allows it, that would be even better. You would get a lot more help and a lot more visibility. But even if you cannot open source the project externally, you can benefit from the same principles open-source projects benefit from.

If you have implemented something useful and now it is in demand, start advertising it instead of focusing on implementing everything yourself or asking for more engineers. Instead, invest your time writing documentation, writing tests, creating a backlog, and maintaining it. Then, when other engineers come to you to ask for more features, show them the backlog and explain when you can get to it. Then give them a second option. Tell them you would be super happy to onboard them and show them how to contribute.

This way, they can implement whatever is critical for them instead of waiting for your schedule. Instead of everyone getting frustrated, you get the engineers you need, and they get the features they need. This is also a great way to see how important your project is. Sometimes we get sucked into things that people do not care too much about. But as long as you are working on it, they are happy to use it. When you ask them to implement their own features, you can immediately see how important they believe your project is. If they are investing the time to learn and contribute, your project must be essential for their success.

The more engineers you can convince to contribute to the project, the more impact you can have, the faster the project will grow, and the more people will use it. The more time you get back free, the more you can focus on the big picture and make sure the project's foundation is solid and the long-term vision is planned nicely.

This is also a great way to remove yourself as a dependency from the project in the long-term. It would be best if you never buried yourself in a project so deep that it fails without you. That means you cannot take a vacation, cannot stop worrying, and cannot change teams or companies without letting the project fail. That is not a good strategy for no project. The less dependent everything is on you, the better it is.

Suggestions: Open source your project internally (or externally if possible) to get the investment you need.

22. How to Get More Headcount

We have talked about open sourcing your project internally to the company. But what if it is not possible at the moment or you do not have the time to write documentation and onboard more people? What can you do to convince the leadership to assign more engineers to help you?

To answer this question, we need to first think about reasons why leadership has not given you any headcount so far.

1. There are no engineers available.
2. Other projects are higher priority.
3. Other projects have earlier deadlines.
4. They do not believe your milestones can be achieved faster by adding more people.
5. They do not believe you are ready to lead.

These are some of the simple explanations that might be happening. Of course, there is one I have not listed: that your project is unimportant. I would hope you are honest with yourself and not pick a project that you already know is not important. If you find yourself in a project you realize later is indeed unimportant, the best thing to do is switch as fast as you can. Do not put your valuable time into things that neither you nor your coworkers seem to care about. Let's go back to our lists.

The first three reasons above are all the same but worded differently. This is usually the excuse given to engineers in order to not break their hearts. Unless your company is downsizing and there is a crunch of engineers, not having available engineers translates to them working on higher priority projects with earlier deadlines or your project not seeming as important. If this is the feedback you are receiving, there is a problem.

Either your project is not visible enough for leadership to see the value in it, or it is indeed not as important as the other projects. If it is a visibility problem, you need to go to the foundations and see how to increase the visibility. Maybe you have not advertised it well enough, or other teams are not using your product enough. Maybe it is not aligned with the long-term goals, or there is no clear profit from finishing it. You need to figure out the reason it is not taking off and fix it. It is hard to do, but you always need to make sure you are getting enough visibility into your project. Maybe you are fixing all the problems quietly. Again, if leadership does not see any problems arising from your project, they will not be incentivized to help you.

Try sending the design document around to make sure you have captured important use cases. Maintain a good backlog and use it during the team's planning meetings to show people the upcoming work. Involve your manager in discussions with other teams to show how important it will be. Look around to see if there are any queries you can run to gain data to convince others that the project will have a big impact. The only way to not be told that other projects are higher in priority or they have upcoming deadlines is to make sure your project looks like a higher priority so that your deadlines matter more.

The second possibility is that you are indeed working on a low-priority project. Then you should be honest with yourself and question what you are trying to achieve. Do you want to work on it slowly and enjoy your life while it lasts, or do you want to jump into a higher priority project? What you should not do is keep working on it and get frustrated that others do not see the value in it.

Of course, there is a third possibility: Your project is actually important, but management is not focusing on it. As we have talked about aligning with leadership before, this is a tough one. You will either need to convince them that this is important or find a different team who is excited about your focus.

Another excuse that is often cited is that your project cannot be finished faster by adding people. The best statement I have heard around this was, "Nine women cannot make a baby in a month." Certain things cannot be done in parallel. If you are receiving this feedback, you need to go back into planning to break the system down into components that can be developed in parallel. Especially if you can find customers who start asking for some of those components, management will be a lot more likely to put some people to work on those areas.

And the last one: Leadership does not believe you are ready to lead. This one is the toughest because if management does not have their faith in you, it means one of the three things.

First, your work style does not match management's expectations, so they do not believe in you. This is one of the riskiest situations to be in. We have talked about the importance of aligning with leadership. If you realize you simply do not get along with your manager or higher-ups and the things you believe in differ significantly, this is a great sign that you do not belong there, and you should find a team that believes in you.

Second, you were already given opportunities, and your manager received feedback on how you have not performed up to expectations. If this is the case, you need to seek feedback, improve anything that was preventing you from being a great leader, and try again.

Third, you are indeed a great leader, but you have not been great at advertising yourself or convincing people you are ready. This one is the easiest one to fix. You just need to trust in yourself and sound more confident during the meetings, push your team and manager to invest in you more, take risks, and prove you can do it!

In all the cases of not having a headcount, make sure you have a great mentor who knows your company and can guide you. An external perspective is invaluable in any kind of situation to stay objective and understand better. Even if they are not your mentors, find people who will watch your back. Someone who is not feeling the emotions you are feeling will be a great asset to fact check and help you figure out the best way to move forward.

Suggestions:

1. Advertise the project and yourself better.
2. Maintain a good backlog and make it visible.
3. Make sure other team requests are visible; do not fix things before discussing them during the sprint.
4. Seek feedback and improve yourself.
5. Find another team with management that has faith in you.

23. Side Projects and Knowing Your Surroundings

What if I asked you right now, "How do your projects align with the rest of the projects in the team? Or in the org? Or in the company?" How would you answer? How many projects are you tracking closely? Have you paid attention to what is being designed in the sister teams? Or were you so busy focusing on your project that you could not find the time to join their meetings or take a look at their design documents?

We have talked about the importance of aligning with the leadership in the "Align, Align, and Align Again" topic. However, there is another aspect to aligning that is just as crucial as aligning with the leadership: knowing what other teams are up to and aligning with them. You should have a high-level idea of what is going on in your surroundings at any given time. What are the others in your team focusing on right now? What are the top priorities for sister teams?

This may sound like a lot of investment and will take a lot of time, but surprisingly, it does not! There is a learning curve for sure. When you first start tracking other projects, you will be confused; you will forget what it was about; you will forget what the updates were. But the more you do it, the better you get at it, and it will take less and less time.

What are the benefits of knowing the projects going around?

For starters, you slowly become the go-to person for everyone else to check if their ideas are worth doing. Most engineers are focused on what they are excited about. But not many people want to put in the investment to stay on top of everything going around. By helping others know what is going on, you immediately become more senior. Think about what your manager does all day long. From one meeting to another, leading expectations, communicating with people, and helping them prioritize. That sounds like what I am suggesting you start doing, right?

You become the bridge between teams. You can start directly impacting projects by commenting on them and aligning them with other projects. Other engineers from other teams start reaching out to you to check what your team is up to. This gives you the power to lead expectations and convince your teammates what is more important and what they should prioritize.

You get a chance to help other teams. Knowing what they are working on gives you the opportunity to be involved in the design discussions, help them make the right choices, and learn from them. The most thinking, comparisons, and choices are made during design discussions. Every design discussion you get a chance to join will teach you so much.

Every project you help opens a new door for developing your career as well. It is tough to join a new company or a new team and start working on high-impact projects as soon as you get there. No one knows you; no one trusts you. You will constantly be tested first, and after you have proven yourself, you will start getting bigger and bigger projects. But exciting projects do not always come along. You might spend years on the same team, and there might never be a new, exciting project. Most of the engineers working for big corporations spend the majority of their careers working on already existing projects doing incremental changes. Unless you are one of the lucky few, you will not get a chance to start something from the beginning and watch it grow. And you even need more luck for that project to be something you are genuinely excited about or dreaming of for a while.

When you are tracking projects in many teams, you get firsthand exposure to working with those engineers on those projects. You spend time designing things with them, working on small tasks with them. You get insider knowledge of how working with those people is going to be. You get insider knowledge of how the system is being designed and the long-term vision for it. And if you like the team and the project, you can simply work with them. Of course, you need to figure out the logistic parts, like if you can help them from your team or need a team change, etc. But when you are excited about a project and have already been helping them, I do not see any reason why anyone would try to stop you.

This is a trick I have been following my whole career, and this is how I always ended up working on the most critical high-priority projects, learning a lot, and working with people I genuinely enjoy working with. It is like a lottery ticket when you get burnt out and jump ship to a random team. It might go great; it might go terrible. However, when you help multiple teams and pick the one you like best, you constantly find yourself working on a project you truly care about with people you like working with.

Suggestions: Always track what other teams are working on and be a bridge between teams. Not only this makes you more senior, and it also gives you a chance to join the projects you are most passionate about.

24. The Importance of Networking

In the previous topic, we talked about the importance of keeping track of what everyone else is working on. Now let's talk a little bit more about how you can actually do it. Are you going to be using tools to figure out all the documents people are creating and sniff around their documents to track what they are writing about? Are you going to check public calendar invites for design discussions and go to them? How will you even know when and where, and what kind of design discussions are going on at any given time?

The best answer I can give you is networking. People share what they are working on with their coworkers if they feel close to them. People will want you to work with them if they know and like you. They will be the ones reaching out to you if you have good connections. Networking is important, but most of us struggle with it.

Personally, I disliked networking because of my perspective of it. I am sure you have all seen movies with fancy events where you keep talking to random people trying to build connections. Networking felt like approaching someone with a hidden agenda of some kind of gain. It felt fake. It did not feel natural, and it was against my personality and ideals. The funny part was I love making new friends and have always been sociable. I love meeting people. I love learning their perspectives and hearing their stories. I love talking about design discussions. I love learning new things.

I basically networked most of the time when I did not realize it was actually networking. As long as I was introduced to someone naturally and did not have that hidden agenda of trying to gain something from them, it happened easily, and I have built so many great friendships. I have learned and grown so much, thanks to those people. I worked on many groundbreaking projects and got promoted many times, all because of the wonderful things I have learned from my network.

Most engineers invest so much time improving their technical skills. They try so hard to be great engineers. We all spend hours and hours watching videos, reading blogs or books, and prototyping new technologies to learn them. But if you do not have the network to use those skills, what good do they do? There are a few exceptions, of course. I have met with some great engineers who simply did not like people too much. They were great at the technical part. But unless you are one of the best engineers in the world who can code everything by themselves, you need people. You need people to learn from, you need people to discuss with, you need people to gain perspective, you need people to believe in you to get your project approved, you need people to believe in you to give you headcounts, you need people to give feedback so you can improve yourself.

One thing that was helpful for me was changing my perspective. Instead of thinking I needed to network to gain something from someone, I started adding it to my day-to-day life. Instead of running to someone when I needed something, I started building those relationships before I needed anything. Instead of eating lunch by yourself, try going with coworkers. Instead of the same coworkers every day, ask them to invite more people. Join happy hours or any other company event where you can meet other engineers. Sign up for conferences. Go to as many design discussions as you can. Start your own book club or design club. Set up a monthly or biweekly meeting and invite your coworkers. Tell them that each time one of you will pick a topic, and you will all discuss it. Pick a book and share the high-level learnings, pick a part of the system and redesign it for fun. The more you invest in these kinds of clubs at work, the more people will join. The more people put in their efforts, the more you will learn and increase your network. The next time you actually need something from someone, instead of feeling you are approaching someone for some gain, you will feel you are approaching a friend to help you with something. Everyone loves helping a friend.

Suggestions: Find ways to introduce networking to your day. Go to events, start your own discussion groups, book clubs, technology review meetings, or simply grab lunch with your coworkers.

25. Changing Teams—Risks, Timing, Making the Switch

I have worked with quite a few different teams in my career. Team change is never easy, yet it is one of the best things if executed well. You might be feeling excited about a project in a different team, not feeling excited about your work in your current team, having a hard time with your coworkers, or interested in learning something new. Maybe you want to be the first engineer working on a new product or one of a million other reasons we cannot cover here. It is one of the most common things in the tech industry. People love to change, but it is scary to move sometimes.

What if your manager finds out you are trying to leave? What if your promotion is not given because you are leaving? Or maybe your bonus will not be the one you deserve. You might be put into a lower performance bracket. Would you get fired if your manager learns you are trying to leave? Do you go around publicly advertising you are searching for a new team? Or do you need to be hush-hush about it?

The bad news is there is no one size fits all… The good news is there are specific actions you can take to minimize the risks.

If you ask your manager or anyone around about a team change, you will likely hear the following sentences:

- We are one big family as a company, it is okay.
- I am sure your manager will not punish you for it.
- Your manager will do the right thing.
- Your manager should respect your decision and should be supporting you.
- Your promotion should not be affected by changing teams.

These are great to hear and should be what to expect. Especially if you have a great manager. Great managers do not feed on fear; they support you no matter what, and they should always fight for what you have deserved. They should not base their efforts on the expected return from you in the future. I have worked with excellent managers who would give people bonuses and promotions and fight for them even if they knew a leave was coming. If you have a great manager like that, I suggest you think twice before deciding to leave.

Managers are people. People have egos, self-preservation, and their own worries. You should not expect your manager to think about what is best for you. They will be busy spending most of their time thinking about what is best for themselves. When we are engineers, we spend most of our time investing in ourselves, picking the exciting projects, learning exciting technologies, waiting for the weekend to come for that nice trip, and so on. But when we think about our managers, we expect them to spend their time making us successful. How selfish is that? We spend all our time for ourselves. And when we have a manager, we want them to spend all their time on us too. Managers are not that different from us.

Let's assume performance evaluations are coming up. You are up for a promotion. You are the best engineer on the team, and everyone agrees you should be promoted. But the budget for this cycle only lets your manager get two engineers promoted. And there are three people ready for it. Suppose everything was black and white, and there was a metric to stack people, and you are on the top, and your manager is an ethical person. In that case, you should be promoted as you are the top deserving engineer. You have already launched the project successfully and proven yourself. I have seen this happen. I have seen people who left their team and still were promoted by their old manager even though they were leaving.

Unfortunately, it is not black and white in real life. And there is no agreed-upon metric to stack everyone and know who is best and who is worst. You brought the most profit, but another engineer implemented the system you are dependent on. You have commented the most on code review, but another engineer answered a lot more questions for other engineers in chat threads…

Put yourself into your manager's shoes. What would you do? Three engineers are ready for promotion, and only two can get it. You already know one of the engineers is going to be leaving the team. Would you promote the engineer who is about to leave so they can develop their career in the new team while you spend a few months dealing with a disappointed engineer in your team who believes they deserved the promotion, and yet you picked someone else?

And sometimes it is not even up to your manager. For example, some companies have committees who decide on promotions. Your manager needs to prepare a package for you to be presented. Again, a great manager should put the same amount of effort into your package as other engineers'. But not only engineers are overworked. Managers do not have too much free time either. They are running from one meeting to another, catching up with dozens of chat threads, and so on. And preparing your package takes a lot of time. It is not easy to go through months of achievements, read dozens of feedback comments, and prepare a package. Even though great managers should still put in the time, sometimes it is easier to create a lighter package for engineers who are leaving the team.

What is the best course of action you can take? First, get familiar with the company performance deadlines. Unless you are miserable and need a change fast, learn when the performance cycles are coming up and by what date bonuses and promotions are finalized. Your manager might be great, your manager might not be great, but if you are following the deadlines, there is simply nothing they can do about it at that point. Better to be safe than sorry.

You are planning to change teams after the results are finalized. But when and how do you start picking new teams? Again, there is no great way to handle this. In my experience, if you reach out to some other manager and ask a few questions, they will not go running to your manager to tell them. Especially if you tell them you are simply exploring and would like to keep it discreet for now. Unless they are best friends with your manager, I cannot think of any reason why they would not respect your request to be discreet. Even though there are some risks, it is comparatively low, so I would suggest not worrying so much about your manager hearing it from other managers. At the end of the day, the new manager wants you in their team, and it is not smart to screw someone over by disrespecting their request before you even start working for their team.

You can even do better to help the team you are interested in with some of their tasks in your free time. This will let you familiarize yourself with the project and the team. And this will make it look like you are just helping because they are in a time crunch.

Regarding fears of being fired, I think that is unlikely—unless you start performing badly because you think you are leaving. Not a good idea to slam any doors because you are leaving. Always keep putting 100 percent into your project until your last day. Suppose your manager is not too happy with you. In that case, they will be happy to learn you are not going to be their responsibility anymore. If they like you, then they will try hard to keep you around, not fire you.

It is common knowledge that once you change teams, you lose some of your progress. This is another thing you need to be careful about. We have talked about being ready for promotion and not getting it. But what if you have worked in the same team for a year and finished big projects, but you are not fully ready for a promotion yet? What if you need another six months to a year in the current team to prove you deserve the promotion? Do your achievements carry to the new team? Unfortunately, only some of it transfers with you. If you talk to the leadership, they will likely assure you will not lose any of your achievements. But the reality is not that black and white. A good manager will try to help you carry your achievements over. But it is tough.

The new team will not have the context. Unless you have published a service used by millions of people or brought a lot of profits, it is hard to measure someone's success in another team. The new team will not have visibility into how hard you have worked, how many people you have helped, how many tasks you have finished, or how many times you have saved the production from crashing. Were you woken up at 3 a.m. multiple times while you were on call? There is simply too much information for the new team to know about. They will try their best to learn who you are but on a superficial level. You should be prepared to lose some of the achievements you have made if you are changing teams.

Therefore, most engineers change teams right after promotion. To capture all the achievements and not invest in the team you know you are going to leave soon. If you can wait to get your promotion, that would be best. But if you do not see it coming on the horizon and you have found your dream team, I suggest you close your eyes and jump in. There is nothing better than being excited about your project to develop your career. You might think you have missed a promotion. Still, if you are working on a project you are excited about, and learning so much in the new team, you can always change employers to get your level matched to your skills. Doing what you love and what teaches you the most is always going to be the best investment in yourself.

Suggestions: Familiarize with performance deadlines. The best time to switch teams is after your results are finalized, especially right after a promotion, if possible. But if you are excited, close your eyes and jump on it!

26. How Fast to Respond to Emails/Chats

We have mentioned in a few different topics the importance of people's perception of you. One of the easiest ways to build that perception is how fast you respond to emails or chat threads. Especially group chats. People will build an image of you without even realizing it. Who is the go-to person in the team for any questions? Likely the same person who is always the first one to write an answer.

On this topic, I have gone back and forth a few times in my career. Because answering people requires time. It takes you out of your focus while coding. Sometimes it takes a while to find the best response. Sometimes it is hard to word things, so you keep writing and deleting… I have received feedback for not being fast enough on questions. I have also received feedback that I am too fast at responding and that it might reduce my productivity. There were days I tried to focus on responding faster. There were days I tried to only respond to people at certain times during the day and not check notifications for the rest of the day.

I cannot tell you how often you should check your notifications because it depends on how much they distract you. For example, some people are great at context switches. They can keep going back and forth without losing context. On the other hand, some people need half an hour to focus again after losing their context. You need to find your sweet spot to decide how often to check the notifications.

But there are certain things you can do to be more responsive when you do answer people.

Do not wait for a complete answer to type something. I used to try to understand the context, put together my thoughts, write a complete answer, and click send! But many times, I would forget to hit send because I would be pinged in a few different threads. I would put together my thoughts, and as I was typing, someone else would ask a question. By the time I got to the other chat, it had already been a while, and sometimes someone else had already answered, thinking I was busy.

The best thing you can do is to type acknowledgment messages.

1. You can simply acknowledge you have seen the message like "Ack," "Acked," "Acknowledged," or "Looking."
2. If you are not free and need a while to look into it, you can just type so. "I will respond in a minute," "Have a meeting but will get back to this in a half hour," or "I will answer by EOD."
3. Or simply answer by asking the priority: "How urgent is this?" "Can I look at this later?" or "Can I get back to you later today?"

When you answer messages fast, people start building a perception of you always being ready to help. Always working. You can be answering as many messages as someone else. Still, if you are sending these kinds of acknowledgments as soon as someone messages you, they will think you are working harder. It is crazy how our minds pick up signals that might not be entirely true. But that is the critical part. Signal! When you do not answer, they do not know if you are saving the world or simply slacking. It is better to give them the image that you are working hard. And as you can see from the examples, you do not even need to spend more than a few seconds. All you are doing is letting them know you have seen it and need more time. Even if you have not even read the message yet, you have already set expectations.

Another thing you can do is to send pointers instead of full answers. There is nothing better than a complete answer for the recipient. Of course they would like you to answer their question in full, but it often takes too much time to do so. If you are busy but still want to help, send pointers like "Try searching for x" or "Have you talked to y?" Or send a few links to design documents. I have seen people even sending Google links with a search with the same question people asked. You would be surprised how sometimes people ask such simple questions that all they need is a Google search. Of course, I do not suggest you send Google links as it is humiliating. Still, if someone keeps asking super-simple questions instead of doing a basic search again and again, then you can try it at your own risk.

Another trick is to tag people if you are not the right owner. If you have free time, you can invest your time answering. But if you do not have the time, don't be shy. Just tag someone you believe who is a better match to answer. And you do not need to pick the right person on the first try either. You can simply tag them and ask, "@Merih, please take a look or tag the right owner." No one expects you to know every owner of every project. You are achieving two things by tagging people. You are saving other people the time of reading the same message. If I see someone else is tagged, I know it is someone else's responsibility so I can skip reading it. And if it is urgent, the owner of the question can reach out directly to the person you have tagged. Most people are slow on group chats but faster on one-to-one messages. The second thing is you look like you are working hard because you responded so quickly, even though all you did was to tag someone.

One thing you want to avoid is to answer questions with one-to-one messaging, especially if you are putting more than five minutes into the answer. This took me a few years to get used to, and I am still not great at it. But if people come to you with questions and you have invested a lot of time answering, you need to utilize that time better by sharing it somewhere. Otherwise, only one person gets the benefit of all that time investment. There are likely a lot more engineers who need a similar answer. Either they will ask you, and you will spend the time to answer again, or they will ask someone else, and now they need to waste time doing exactly what you have already done. Or even worse, they will not ask it and research it themselves and waste a bunch of time on something you already knew how to do.

It takes practice, but if you realize someone is asking a question that can be beneficial for others, ask them to make a post about it and ask it there. Tell them you are happy to answer, but a post is better suited for this. Or if you do not want to ask them to ask in a post, after answering the question, make the post yourself. Just share, "I have been asked X, and here is how you do it. Thought it might be useful for others." The best part of sharing things as posts is that it makes them searchable. The next time someone else has a similar question, one keyword search might save them a lot of time researching, asking, and waiting for answers. Not to mention this greatly increases your visibility.

Suggestions:

1. Acknowledge receiving messages by simple messages as soon as you see them. It can be just a simple "Ack" (Acknowledged) message.
2. Send pointers instead of full answers if you do not have the time. Just a link can help them get unstuck until you have some time to invest.
3. Tag people, even if you are not sure they are the right owners. This will also save others' time to not read a long message if they can tell they are not the owner.
4. Leverage posts to prevent the same questions from being asked again. This will help people find your answer later with a simple search.

27. One-on-Ones with the Leadership

The idea of having one-on-ones with my manager used to scare me a lot. *What if I said something wrong, what if I have not accomplished enough, what if I disagree with them on something and start arguing, what if they tell me I am not doing that great?* The list goes on…

I have rarely had one-on-ones with my manager and have even tried to decrease the frequency as much as possible to avoid feeling afraid and uncomfortable. I would only talk to my manager's manager if I bumped into them in the corridors or if they scheduled a meeting with me. Anything above my manager's manager, I would not even know what they looked like or how they sounded.

I hope you are at a better stage than where I was a decade ago. But what changed between then and now for me was the way I look at these meetings. You guessed it right: perspective. We have talked about perspective on many different topics. How you visualize things in your head defines the path you are going to be walking. It is crucial to have the right perspectives.

Let's talk about my current perspective on one-on-one meetings and how it can help you have a bit of a different perspective and make your next one-on-one more enjoyable than fearful.

The most significant perspective change I have had during my career was considering management's role in my career. I used to believe I was there to do what they were asking me, and they were there to make sure I was doing a good job. I would treat them as if they were my boss. Technically speaking, they might be my boss, but I am sure you have seen many blog posts about boss vs. leader. I am talking in the context that I would treat them as an old-school boss. I would go into the meetings with a list of achievements, and we would run the meetings more like a status update: the things I had done well, the timelines, the things that were not ready to be developed, etc.

What is wrong with this approach? For starters, you do not get much out of these meetings. You do not learn anything new; you do not understand your manager's vision; you do not get a good feeling about what is important. You simply give your status update. It is somewhat useful for your manager and not useful for you at all.

What other responsibilities does your manager have other than keeping track of your status updates? So many… First, they are there to make sure you are happy with your career. Second, they are responsible for making sure you are not burning out. Third, they are supposed to help you plan your career development. Fourth, they are there as a guide to help you plan your long-term goals. Finally, they are there to help you not to fail! This last one is super important. They are not there with a whip to punish you when you fail; they are there to make sure you do not! If you are failing, it is not your fault, and your manager does not get to blame you. If you are failing, it is going to hurt both you and your manager.

Each company utilizes managers in different ways. Some want managers to help you during your day-to-day job. Like if you need to set boundaries with another team, or if you need someone to drive the design discussion, or if you need someone to tell other teams about deadlines and not let them push you to work harder. Some companies treat them as more of a people manager, meaning they are there to make sure you can do your best job. They stay away from technical discussions. Some companies treat them as tech leads where they help you with the designs, etc. But all the companies I have seen so far have one thing in common: They want managers to be a resource for everyone. You can think of it this way: The managers are actually working for you, not against you!

Wait, what? Managers are there to work for you? Yes, you heard that right. And it is up to you to hold them to that standard. If you treat them like a boss and give them power over you and keep fearing them, they cannot do much to help you. They cannot be there for you because you are not giving them the opportunity. But if you can change your perspective and begin to think of them as being there for you, you will not believe how much of an advantage you are going to gain from this relationship.

As we talked about earlier, managers are busy people like the rest of us. They are not going to spend too much of their time trying to figure out how they can help you best. It needs to be a team effort. You need to decide how to utilize them. They have strengths and weaknesses like the rest of us. Some managers are great at setting boundaries with other teams. Some managers say yes to everything and then come back and start pushing you to work more. Some managers are great at seeing high-level goals and can help you not waste your time on unimportant tasks. Some managers are bad at prioritization and will keep assigning you everything they hear about. Some managers are great emotional mentors who can keep you positive and give you a good perspective. Some simply are not. Some managers will fight for you against the leadership and get you promotions; some managers will be the roadblock that prevents a promotion even if you deserve it. You need to get good at reading people. Take notes on things you see. This will help you recognize patterns. Keep noting the things they do amazingly well. And the things they are not doing that well. There is no point in asking your manager to do something if they simply do not have the skills for it.

I have seen managers invest time to do pair programming to help someone get better at coding. I have seen some commenting on code reviews. I have seen some hands-off and only giving guidance. I cannot stress enough the importance of getting to know your manager well. Tell them you do not want to use your one-on-ones as a status update. If they have high-level questions about deadlines, etc., of course, you should answer them. But never let your one-on-ones turn into a long session of status updates. There are many other meetings to do that.

Your top goal in one-on-ones should be career development. Do not be afraid to tell your manager what excites you, what you hope to learn. Do you want to work on scalability issues? Do you want to learn concurrent multithreaded programming? Do you want to pick up new technology like NodeJS? Of course, they cannot grant every wish you have. But the clearer you are, the more chances they will get when an opportunity presents itself. Then the next time they hear a project with the skills you are hoping to learn, you will be the first name popping into their head.

Your second goal from one-on-ones should be long-term vision. We all get 24 hours in a day. We cannot do everything. Everyone needs to focus on what they do best. And managers are expected to be the ones who know the long-term game. Not every manager is great at focusing on long-term vision, but they need to improve themselves even if they are not that great yet. Management is a learning process at the end of the day. You should do your best to have an idea of what your team is trying to achieve.

We have talked about knowing different projects going on in different teams and aligning with them. Well, your manager is a great resource to do that. That is one of their responsibilities: to track what everyone in the team and other teams are working on. They are the ones who go to higher management to get other big projects approved. Use your manager's strengths and ask about the long-term plans. Where are you headed? What can you do to help the most?

The third goal from one-on-ones should be getting high-quality feedback. You should not fear hearing you are not doing great. What is the alternative? There is nothing scarier than a manager who keeps feeding you platitudes like "You are great," "I am so happy with you," or "I do not see anything you should do better," but then you get a bad performance review. Which one would you prefer? Hearing how amazing you are just to learn later it was a lie? Or hearing what you are doing that needs improvement so you get a chance to actually work on it and fix it? Nothing is better than early high-quality feedback to make sure you are on the right track.

Why do I explicitly say, "high quality"? Because not all feedback is actionable. And hearing things that make it sound like you are not doing well without a clear path to do well is simply going to make you feel worse. Do not let that happen. Make sure the feedback you are receiving is proactive and actionable. If your manager simply says, "You are not performing at your level" or "Your project is not good enough," these are low-quality statements that do not achieve anything. They are still valuable to hear so you do not waste time working on them. But in this case, your manager is not helping you to become better. Keep asking follow-up questions to clarify. If they say, "Your project is not good enough," simply ask them to give a concrete example of which part does not match their expectations or to show you which project is better in their perspective. Ask what you can do to raise it to their standards. If they say, "You are not performing at your level," ask them to be more concrete. Ask them to give examples so you can better understand. Is it that you do not respond to people on time? Is it that your design quality is low? Is it that you are running late for meetings?

Do not be afraid to ask them to coach you. If they see something that can be better, ask them to break it down to milestones and work with you. Receive their feedback, write it down, summarize after the meetings, and put together a plan to move forward. Send them an email of what you understood from your conversation and what you think you can do to make it better. Tell them you want to touch base on this topic every week to see if you are doing better. Of course, they will push back a bit because they are busy, but if you do not push your manager, they might not do their best for you.

Think about when you were a baby. Would your mother feed you if you were not crying? You should always make your manager feel you are engaged, and you will keep following up until they help you. Most managers want to help, but they are simply too busy, so they will not even remember what you talked about yesterday if you let things go. You might be thinking you have talked about something important because it affects your life from a first-person perspective. Remember, your manager likely went into five more one-on-ones with other people who had just as many problems as you have in the same day. You need to work with your manager to help them as much as you can so you can be a real team.

What else can you talk about in your one-on-ones? You can ask for recommendations. Your manager is likely talking to other managers and has good visibility on other engineers' focus and how good they are for different topics. Your manager can be a great resource on who to reach out to for help. Ask them who they believe is the best person to ask about something.

We have talked about the importance of having a mentor. You can ask your manager to find you one. As we have discussed above, it is likely that your manager's network is a lot larger than yours. Use it to your advantage. Tell them what you are hoping to achieve or get better at and ask if they know anyone who is willing to mentor you to achieve that. A request coming from your manager to help you goes a long way. If you reach out to people and say, "Can you please mentor me?" they might not be too receptive. Mentoring takes a lot of effort at the end of the day. Using your manager to ask the same question can put weight on the request. And they will get credit for saying yes. If you ask someone to mentor you and they do help you, that is unofficial. If your manager asked them to help, your manager will be tracking the progress of the mentorship, and during performance evaluations, your mentor will receive good feedback from your manager. It makes it more official and beneficial for all parties involved.

Another thing you can talk about during one-on-ones is other people who need help. I do not particularly enjoy complaining about people, so in my career, I have never gone to my manager to complain about someone unless it became unbearable, and I have tried a few different approaches with no success. But this is exactly the wrong perspective to have. You should not feel like you are complaining about someone. Instead, you should feel you are helping them. If someone is doing something that sets you off, then the chances are they are doing it to a bunch of other people. If you do not give them the opportunity to hear your feedback, they will anonymously receive all that feedback during their performance evaluations. They will be set to fail. Make sure to pick your words wisely so it doesn't sound like you are complaining about someone and focus on what kind of help your manager can bring. For example, you can tell your manager person X needs a hand on external communication. Tell them you feel they are not communicating enough with the other teams. Your manager can mentor them to perform better. People appreciate this kind of sincere help.

It is also important not to be a stranger to higher management. Everything we have talked about in regard to one-on-ones with your manager, you need to also do with their manager and their manager and so on. I used to think higher management would not have the time. I was positively surprised to see whenever I set up a meeting request with the skip-level manager, their manager, or even the directors sometimes, most of them accepted, and some asked for a different time. Only a few used the "I am busy" excuse. And even then, when I asked follow-up questions like "When would be a good time?" or "How about next week?" or offered to schedule the meeting 3-4 weeks ahead of time, I cannot remember anyone who turned me down. You need to expect a few bumps in the road. Sometimes they get pulled into important meetings and need to reschedule a couple of times. Sometimes you wait, and they do not show up. But consistency is the key. If you keep asking, you can have a meeting with anyone.

Why is it so important to meet with higher-ups? Because they are the ones setting long-term goals, and they are the ones who know all the details from the planning meetings. They might not want to share every detail with you. At the end of the day, they need people working on lesser priority projects too. They will not tell you your project is not important. But if you are like me, someone who likes to challenge people, ask it anyway. Ask it point-blank when you are looking in their face. They may not share every detail verbally, but their face will tell you so much. Phrase your questions to require a complete answer. Stay away from Yes/No questions. If you ask, "Is my project important?" they will say yes immediately. If you ask, "How do you see my project aligning with the long-term goals?" or "How do you envision my project evolving in time? What is the best outcome you think we can get?" they will have to think. The more people go out of their memorized answers, the harder it is to keep you in the dark. The more they talk, the more you will learn how they think, what they see, what their plans are, and how important your project is for the long-term goals.

As you are reading this passage, I am sure you think what I used to think. *It is so hard. Why would they invest the time? I cannot imagine challenging a director*. Fear gets inside your head, distracting you when you try to visualize. Your heart rate is increasing. But believe me, it gets easier. You need to take the first step, and the more you talk to them, the more you realize they are just human beings. They are not robots working 24/7. They have hobbies; they have families; they have strengths and flaws like the rest of us.

The most important part of meeting with higher-ups is so they can put a face on your name. Have you ever chatted with a customer support agent online? I am sure you often felt that they did not understand what you were going through, they did not seem to care, or they felt like robots. Not being able to put a face to someone does that. We have evolved around reading people's facial expressions well. When higher-ups do not know who you are, it is hard for them to remember what you have accomplished. You are nothing but a name on a summary document during performance evaluations. Do not be just a name. Make sure everyone who is involved in making decisions about your career can put a face on your name. They should all know what you look like, how you smile, what you get passionate about, and how you argue for what you believe.

Also, it might feel hard to start reaching out if you have not done it before. It gets harder every day you have not talked to them. I have been there and done that. You do not schedule a meeting with higher-ups for the first few months, then you think "Maybe later," then again later, then you realize it has been six months, and you have not even said hi to them. You start thinking, *I will when I have an important project or an important topic*. Then you start feeling no project seems to be important enough. As each day passes by, it gets harder and harder.

If you are already in this loop, you need to act before you think. We will talk about how to act without thinking in a later topic, but just go ahead and schedule a meeting with them as you are reading this. Do not think about what to talk about, do not think about anything. Just find their name and click "Schedule a meeting" with one of your higher-ups. You will soon learn the power of throwing yourself in a situation.

Do yourself a favor and never end up in the same situation again. I agree with you; it gets harder every day. So do not wait. The first thing you need to do before choosing to join a team is to meet their management chain. It is a lot easier when you have the leverage. I highly suggest working for a team that has a management fully aligned with your vision of the world. If you do not believe in management, you cannot be your true self. So before changing teams or companies next time, request a meeting with the manager's manager, their manager, and even the directors. Ask them a lot of questions. Challenge them. Each time you change teams or companies, you are investing the next few years of your life. Make sure you will have great stories you can tell. The best way to ensure that is to meet the management and see if you believe in them.

Suggestions:

1. Always have one-on-ones with your manager, their manager, and even the directors.
2. Make sure you feel management is there to support you, not against you.
3. Get to know the management chain, make sure you are aligned with them, and always work in teams where management has your full faith.

28. Act Then Think

This is one of my favorite sentences. It changed my life. I used to overly prepare for everything and live inside my head. I would spend hours and hours thinking and thinking. Of course, it is beneficial to do so when you are preparing for a critical design discussion or when you are about to make a massive decision like which company to work for. But for decisions that are not that important, overthinking does more harm than the benefit it brings.

Have you ever found yourself wanting to ask a question, but you wanted to make sure you did not ask something too easy, so you kept thinking and thinking, and finally you got so frustrated you decided not to ask anything?

Have you ever wanted to schedule a meeting but wanted to write down a summary first and then got busy and totally forgot to schedule the meeting?

Have you ever wanted to meet someone and then kept thinking about what would be a good introduction and an excellent way to impress them? Then you tried to come up with enough things to talk about for half an hour before scheduling the meeting and decided you did not have enough and postponed the meeting?

I am sure you have experienced some of the examples above and many more in your career. Sometimes we overthink. And it starts doing more harm than any benefit it brings. There are many coworkers I have lost the chance to meet because I was too busy planning what to talk about with them and then got too stressed and decided not to do it. Many times, I wanted to reach out to higher management, and the same thing happened. Sometimes, I wanted to ask a simple question but spent too much time researching and gave up asking in the end. It happens to all of us.

The worst thing is when you start getting the feeling that you are silenced. In many design discussions in my early career, I would think and think and think before opening my mouth. By the time I decided how to phrase my question, we would be on a completely different topic. I often wanted to say, "Let's go a few minutes back" or even invent a time machine so I could go back in time and speak up. Everyone goes through similar things, but you will get frustrated and feel silenced if you keep letting your thinking get in the way. There was a time in my career when I would not even bother trying to speak up because I knew I would be too late anyway.

What is the alternative? You can act before starting to think and then figure it out on the way. What do you lose when you do this? You may suggest an idea that you believe is obviously stupid. Have you ever heard your coworkers saying, "There is no such a thing as a stupid idea"? I hear it all the time. Why do people feel the need to keep saying this? Because this happens to all of us. All of us choose to stay silent sometimes, even when we have important things to share.

Let's measure the pros and cons of each scenario. Let's assume you have spoken before thinking thoroughly. And the idea you suggested was pretty bad. Or was it? I cannot explain how many times I thought it was a bad idea and people were amazed by my suggestions. Of course, there were times when what I suggested did not make sense, but what I realized is that most of those times, those suggestions made us think. You might be thinking the presenter has already thought about all these side choices. But you will be surprised to see how many times people do not think of every possible scenario. It is not how our brain works. When our brain is convinced that something is going to work, it stops searching for other answers. Our brains do not spend a lot of time thinking about what other ways to achieve this. So even when you suggest something that sounds too simple, it might spark a conversation that leads to important gains.

What is the other possibility? You are suggesting something amazing! Then you definitely should not stay quiet. How can you know which side you are on? Of course, you can stay quiet and keep thinking even longer to decide if your idea was good or bad this time. Or you can simply speak up! You do not need to build your whole phrase before saying anything. I have met amazing engineers who just started by making sounds like "Umm, umm, umm." And not even small team meetings. One time in an all-hands meeting with over a hundred people, someone did that. Everyone suddenly went silent to listen. All that matters is grabbing people's attention to stop them from jumping to another topic. You can start with "Just a second," or "What about, mm …" Just buy yourself some time and keep making filler sentences until you prepare your phrase. What is amazing about this is how your brain works so much faster when you are on the spot.

Suppose you start thinking about what to talk about during a presentation. When that is the case, you might spend days and days planning and perfecting every sentence. But have you ever been pulled into a spontaneous discussion, and you were able to talk non-stop for the whole duration? Let's call this active and passive thinking. When you are not on the spot, you are thinking passively, slowly, with the security of your silence. You know you can always give up on what you were about to say and save yourself the embarrassment. But when you start making sounds, you start thinking actively. You put yourself on the spot, and now you need to complete a sentence, so your brain starts working a lot faster to pick the best sentences for you.

One thing to note, though: You still have the safety of giving up. Most people feel once they start a sentence, they have to finish it. Think about how many times you have noticed someone start talking, and a few seconds later, they cut themselves off and say, "Never mind"? Would I or anyone else think less of them? No! You can always give it a try, and if you cannot put your thoughts together, you can tell people to give you a few more minutes to think.

What about a meeting like the one with your director? You are curious about meeting them but do not know what you want to talk about. A 15-

minute meeting sounds weirdly short, and 30 minutes feels like forever. What do you do? Do you give up? Do you spend the next few hours planning it before even knowing if they will have time to meet you? No! Just go ahead and schedule the meeting. Put it on your calendar. Switch your brain from passive thinking to active thinking. Put some pressure on yourself. Suddenly you will find yourself thinking crystal clearly.

I do not know if this is common for other people, but I find myself sleepy when I am doing passive thinking. I get tired and start feeling slower and sleepy if I am busy with voices in my head trying to figure things out. When I switch it over to active thinking, I find myself energetic and ready to go. This might be coming from evolutionary feelings of doing passive thinking in safety and active thinking when our life is in danger. If you are like me, then why not put millions of years of evolution into action? If your body already knows how to handle it, just test it out.

The worst part of staying silent, other than feeling not heard and not valued, is educating yourself to stay quiet. Pay attention to your surroundings. You will realize some people always talk. About everything, small or big. You will also realize some people are always quiet. The more you do something, the easier it gets. The more you put yourself out of your comfort zone, the more you get used to it.

Nowadays, when I start speaking, I do not even feel the pressure if it is not a meeting with at least a few dozen people I have not worked with before. And this is coming from someone who would stay quiet most of the time. There was a time I was fighting myself internally, judging myself and asking *Why didn't I say this? Why didn't I say that?* Of course, self-confidence has a lot to do with it. But it is a chicken and an egg problem. If you do not have self-confidence, you do not feel comfortable speaking up. But if you do not speak up, how will you learn how to trust yourself? I have never gotten better at trusting myself by staying silent and not saying what I wanted to say.

Is there a middle ground until you build the confidence to speak up whenever you want? Yes! You can start by exercising this with small groups of people. Or you can send your ideas as an email. I started by promising myself that I would never let myself be quiet when I had something important to say. No matter how early or late it was, it had to come out. Sometimes I would stop the discussion after the topic had already changed. I would apologize and explain that I wanted to understand it better. Sometimes, presenters had to go a few slides back so that I could talk about it. Sometimes, I could not speak during the meeting, so I would write it to the chat groups or follow-up emails. It is never too late as long as you have the will. Start testing out what you wanted to say in a follow-up email. This prevents you from being put on the spot and gives you a chance to think more and correct your words so they are the best they can be. This works great as long as you do one thing! You must promise yourself that you will actually send it out. There were times I waited too long, and when I wrote the email, I felt bad sending it. You need to promise to yourself you will do whatever it takes to speak up. No matter how late it is, no matter if it has been a week. Just speak up, send a follow-up email, or ping people directly.

Another suggestion from my personal life—put your promise out in the world. Tell your partner you are promising to do this or set a reminder of the promise on your phone. I have gone ahead and printed a poster with "Act then think" on it and placed it on the wall behind my monitor with many other self-motivating posters. It was easier to fulfill a promise when I constantly stared at it and remembered how important it was for myself.

So please go ahead and sign up for that presentation you wanted to make, go ahead and ask the question you wanted to ask, go ahead and suggest the idea you wanted to suggest. Talk, send emails, send messages, print posters, whatever it takes for you to develop the confidence to do it without reminders. You will change so fast that you will start asking yourself why it took you so long to get out of your head and start living in the active world!

Suggestions: Stop thinking too much and start acting. Start with the simplest step to force yourself. At the very least, start making sounds like "Umm" or send out the meeting invite, and then you can figure out the rest later.

29. Do Not Be Scared to Take the Time Off!

Are you feeling bad about taking time off? Do you have a lot of vacation saved up but do not have a couple of weeks without too much pressure from projects? Are you scared of looking like you have not done enough? Are you having a family situation but do not want to fall behind at work?

Do not let any of these ideas or more stop you from taking off the time you need. We are not machines. When you need a break and do not take it, your performance and productivity go down. You are not as excited, not as motivated, and definitely not producing as much as you used to. I said we are not machines, but even machines need to take a break. Can you imagine driving your car non-stop from Los Angeles to New York? Can you imagine overclocking your gaming PC and never giving it a rest, and running at high temperatures for a long time? Can you run your oven for the whole day without ever turning it off? If even machines need a break, how do you think you can keep performing without one? You simply cannot!

You need to remember your career is your life. We spend too much time trying to pick good friends or an amazing partner in life. The idea of spending the rest of our life with the same partner scares all of us, and we measure every aspect of it heavily. Do you know what we spend more time with than we spend with our partners? Our jobs. Even more than the time you will spend with your children. On good days, it is six hours; with deadlines coming close, it might go up to sixteen hours a day. This is not even counting the time it took us to get ready for work and the time we wasted sitting in traffic. We invest at least 50 percent of our day at work, 30 percent sleeping, and 20 percent plus the weekends to do everything else.

That is the most important thing to remember. We will be doing this job for a really, really long time. Burning yourself out will not help anyone. Hopefully, you have a good manager who tells you the same. But even if you are in a team that does not take a vacation, and everyone seems to be working all the time, you need to remind yourself you are in it for the long game. When it is time to work, you need to give your 100 percent. When it is time to rest, you should use it 100 percent.

Especially if you have a sickness or a family emergency. Using vacation is comparatively easier than taking time off for family issues since most people believe they have earned it. During COVID, it was even suggested by multiple peers to take a sick day off for mental health. I used to avoid taking time off as much as possible. If I felt sick, I would still try to work. If I had fallen behind a few hours, I would work whenever I felt better. Sometimes I would add time from the weekend to cover it up so that I would not take a sick day. The upside of doing this? Almost none. I would be feeling sick and not be able to be productive. Because people thought I was still working, they would still expect output. If you fall asleep and do not respond to messages fast, you will create the impression you are not working. Making up lost time is even more challenging. With so many deadlines and so many projects to drive together, when you fall behind on the schedule, it adds extra stress to catch up.

I learned the value of taking a sick day a few years into my career. If you are not feeling 100 percent, just take the day off. Make sure you are well-rested. Take a few more days if needed so everyone knows you are not working. This will justify pushing the deadlines, and when you are back, you are rested, motivated, and ready for a challenge. With the added gratitude to the company for providing this opportunity, you will be even more motivated now.

The hardest time for me to take off was for family leave. My father was diagnosed with a serious condition. I was shaken to my roots. He lives back in Turkey, which is over seven thousand miles away. We would see each other for a couple of weeks every year. I thought I could keep working. I thought there was not much I could do for him anyway and did not even realize how hard this was on me. I forced myself to keep working, but my focus was gone entirely. I would start coding just to realize my mind had wandered off. I would grab a coffee and start again. Same thing. I would try to respond to chat threads but without any motivation.

I was late to meetings and was not answering messages on time. I had to skip many meetings so I did not cry in front of people. Even when I joined, my camera was off, and I would not be able to pay attention to questions. People would repeat questions a few times. One of the hardest parts was sharing my father's situation. I did not want people to know because then it would become real. When you are thousands of miles away, it is easier to believe it is not real, and everything is still normal. It was not long before I started getting feedback about my camera being off, not answering questions, and not performing like I used to. I kept pushing and pushing and pushing.

My manager knew what I was going through, so he insisted that I should take family leave. I refused for a couple of weeks. I thought the more time passed, the less sad I would feel, and I would be able to work better. I talked to my friends, and one of them said he went through a similar thing with his father and refused to take the leave. As a result, he ended up with a not great performance cycle. Unfortunately, there is no exception for your father having a serious problem on your performance evaluation.

Corporations have policies to help you. They provide you the time to recover, but they expect you to be working if you say you are working. My friend's outcome was a clear signal to me that I should indeed take the leave.

After taking the leave and spending time with my family, I realized how hard it was for me. I was feeling a brain fog most of the time and had no motivation to do anything. I visited the doctor, and all my tests were normal. Then the doctor asked me if there was anything significant going on in my life because too much sadness can trigger this kind of problem. I could not even believe it because I thought I was doing okay. It is easy to brush your feelings under the rug and act like they are not there. But no matter how good you try to hide them, they are there. Some things get to you, and there is nothing more important than spending the time needed to help yourself. If you have a family emergency, talk to HR. Learn about programs available for you to use and do not hesitate to use them. They are there to help you recover. Put all your focus on making sure you are 100 percent. When you do that, you can go back to work and catch up with everything later. Nothing is more important than your and your family's health.

Suggestions: If you are sick or have a family situation going on, or have vacation saved up, use it! Never hesitate to use the time off available to refresh yourself. You should put your 100 percent at your job. If you cannot, take the time off until you can!

30. Biggest Regret

This was one of my interview questions. The interviewer asked me, "What is your biggest regret?" I kept thinking and thinking and thinking and could not give any answer. He was surprised and asked again, "Is there nothing you have regretted? No project you have designed ever had big failures?" I was getting more and more confident, and I replied, "Not really." I kept thinking more but could not remember even one time any of my projects failed miserably. I have always done my due diligence, started sending design documents early and treated them as evolving documents, got as many eyes on them as possible, wrote unit tests, wrote functional tests, prototyped, and tried different aspects. None of my projects ever had any major issues, and nothing has gone wrong badly enough to be a regrettable memory. I thought the interviewer would be impressed by my skills and how solid I approach my projects and say, "Wow." Instead, he looked at me with pitying eyes and asked, "So you have never taken a big risk?"

I was shocked. I didn't even know how to respond to that. It was almost like he had grabbed a knife and stabbed me in the heart. I did not even care what he thought about me at that moment. All I could think of was what was the biggest risk I have taken. I kept thinking and thinking.

In order to understand why this bothered me so much, I should explain a little bit about my personality. I love taking risks. There have been so many times I have risked my life for adventure. I have skydived, paraglided, dived with manta rays, and even walked over a couple of days-old lava to watch fresh-flowing lava. I have cooked marshmallows inside lava and eaten them. You can imagine how it felt for me to realize that while I take so many risks in my life, I have never really taken a significant risk in my career.

After a little more thinking, I remembered some of the big risks I have taken in my career. I was lucky enough that those risks paid off well; there was nothing regrettable. But these risks were mainly about changing jobs or teams or other career-related moves. I could not remember a time I had taken a risky project that I was not sure would be a success.

I spent a few months fixated on this. Because I had hand-picked most of my projects on my interest topics, I learned things before signing up for them publicly. I always had a clear picture of what I needed to learn and what I needed to develop to make the project successful. These are all great skills, but I could not remember the last time I had felt the anxious rush in my heart wondering if my project was going to work or not.

This might sound like a good thing, but it is not. The rush in your veins and that excitement of achieving something is what keeps us motivated. And yet, I could not remember a time I had taken a significant risk or signed up for a project without knowing a lot about it and working hard to bring it to life. I realized I was too scared to fail, so I kept playing it safe. It is funny how I could go out and risk my life, but I could not do the same when it came to projects. Death might sound scarier to you than failing a project, but for me, it was vice versa. I finished my school as the valedictorian. I have worked on software development since high school. I was always a step ahead of my peers along the way and never had a big failure in my life. That one question the interviewer asked me literally made me realize how scared I was of failure.

Most of us keep focusing on getting promoted, being recognized, and achieving things; we stop asking ourselves what projects we would be excited to work on if we were not scared of failure. What excites us the most? Where do we see ourselves in the next ten years? Do we just want to get one of those fancy titles like a distinguished principal engineer or director of some sort? Or do we want to work on healthcare to help cure people? When I asked these questions to myself, I simply had no idea. This turned my life upside down. That one question was powerful enough to make me question my whole life. I have been thinking about this question for a long time, up to today. It started with what type of project do I want to work on? What is something hard and risky that I can take a shot at?

Later it evolved into what kind of specialty do I want to work on? Drones? 3D printers? Nanobots that can cure cancer? Or stay in software development and focus on the backend and scalability? The more I thought about it, the bigger the answer became. I started questioning; do I really want to be an engineer? I never chose to be an engineer. My whole family is filled with engineers. My father would force me to code before I could play games on our computer. I learned how to code when I was in primary school, and since then, I have known nothing else but coding. My whole life was planned because I was good at it. But was that really what I wanted to do?

I have spent years trying to figure it out. I have gone through career websites, bought career books, and explored many lists for different jobs I could do. I removed my blinders and started looking at everything with a fresh perspective. I have considered everything from being a comedian to becoming an actor. Becoming a musician and just playing my flute and singing. Going back to college to become a professor. Switching into a program manager role. My all-time favorite has been opening up a restaurant or a bar. Even though I have zero expertise in the hospitality business and I would probably quickly go bankrupt, the idea of talking to people all day long sounds so amazing. I love interacting with people. If you look me up and realize I own a bar or restaurant by the time you are reading this book, do not be surprised!

So far, I have not changed my profession. I realized that I do enjoy what I do day-to-day. Being an engineer is engraved in my veins at this point. But I started improving myself on other things that excite me to be ready to switch one day, if I decide to do so. I spend time playing the flute and singing on a regular basis. I regularly meet with business owners to learn more about owning a business. I follow a lot of entrepreneurs to learn more about how to start a side business. I have launched two start-ups. I keep trying things on the side. The more I ask myself this question, the more I find myself loving what I do day-to-day.

When I was a kid, I wanted to be a flute player. My father asked, "Why?" I answered, "Because I love playing it." He told me that I love it because it is my hobby. If someone were to pay me for it, I would have to play even when I did not feel like it. He asked if I was okay with that. I couldn't picture myself playing the flute just because I had to. So I never became a flute player. I still apply the same technique, and it makes me realize how much I like being an engineer. People have been paying me for so many years, and I can still do it every day. On the good days and on the bad days, it does not matter. I can always keep coding something because I genuinely enjoy it.

What did I gain by doing so much thinking for all these years if I really wanted to become an engineer? Especially since my answer has not changed after so much thinking. I became aware of my choices. Now it is no longer the only thing I know how to do. Now it is what I choose to do. And even though your day-to-day looks exactly the same, your feelings change a lot when you realize you really want something. Not only will I not have regrets in the future realizing this was not what I wanted, but also I now get to pick projects I am actually interested in. I spend a lot of time thinking about what I want to focus on. I remind myself of the question, "What is the biggest regret you had?" "What big risks are you willing to take?" and only work on projects that satisfy these questions. I do not want to have any regrets in the future, so even if the projects I work on fail, I want them to be my choice.

What is your answer to "What is your biggest regret?" Or "What is the biggest risk you have taken?" Have you really made the choices in your life yourself? Or did you just follow the path in front of you without realizing it was not your choice? Have you given up on something you really wanted to do? Keep asking yourself these questions again and again and again. And do not worry if your answers are changing, because we keep changing. You will not be able to answer these questions today and follow them for the remainder of your life. What is important is making sure you have made the right choices today and asking yourself again tomorrow.

And if you have never failed any project or at least came close to failing, consider your decisions again. Work on projects you are proud to talk about, not the ones you know will succeed. It is hard to take risks when we all have financial obligations. Still, one of the great pieces of advice I have received was the following: It is safe to do one for yourself when you do two for the company. If you want to play it safe, pick two projects that you know will succeed and pick one risky one. If you fail, everyone will know it was the project and not you personally, since you have successfully demonstrated your skills. If it is a success, even better. You can keep taking risky projects until one fails. If it does, reset. Two for the company and one for you again.

Suggestions: What is your biggest regret? Or the biggest risk you have taken? If you cannot answer these questions, that means you are playing it too safe. Try to work on two safe projects but then jump on a risky one. Work on things you will be proud to mention in the future.

31. Perfectionism—Your Biggest Enemy

I hope you are thinking to yourself, *What is perfectionism?* If this word is not familiar to you, you are one of the lucky ones. But for most of us, perfectionism is embedded deep in our brain. Our society is designed around perfectionism. You are encouraged to be the best in the class, keep getting better scores. You need to study hard, achieve a lot, find good companies to work for, keep working harder to get promoted, and it goes on and on and on. And if you fail any steps of the way, it might be hard to recover from.

I started a start-up to build apps I strongly believed in. But I could not resist my internal pull toward perfectionism. Who would not want to build the best app and have millions of users, with everyone talking so highly about both you and your creation? You are walking on top of the clouds, and every time you look at your app, you feel your ego going above the clouds with you.

One of the apps was to send photos to friends and collect anonymous feedback from them. I bought the domain for the app. I hired a designer. I worked non-stop to design every surface of the app. I spent so much time making it perfect. It had to have everything. I came from years of experience in building software at the end of the day. I would not be able to just publish an app that did not work properly. We built messaging, friending, following, different options for anonymity. I hired engineers to work on it and spent a lot of time leading them. So much time was spent on every aspect of it.

Do you think it was a successful app? It could have been the top app in the App Store. But instead, what happened is we ended up being too late. A couple of students published the same idea with only one functionality: to send photos and receive anonymous feedback. They became one of the top downloaded apps in the App Store in almost no time…

All these investments, all those all-nighters, all that time perfecting every pixel of the app went down the trash. Why? Because of perfectionism. If only I had sat down a few nights myself and put the idea into a prototype and published it, I would have immediately gotten feedback and would have known what users were asking for and what was not really needed. And if my app had started getting famous, I could have always hired excellent engineers to add any functionality it needed.

The sad part was that this was the fifth time an idea I had became popular, except that it did not get famous through my apps. You might think I would have learned my lesson from this and started publishing things fast, right? Not immediately. We started working on the second app. Once again, so much work to make it pixel perfect. It took over a month to just submit it to the App Store. What if people downloaded it and it crashed? What if it did not work in different-sized phones? What if we got a few bad reviews? What if, what if, what if. Clicking the "Submit" button was one of the hardest things I have ever done. Guess what happened afterward. The App Store only took a few hours to launch our app. After a month of postponing, testing everything, and making sure everything was perfect, they just launched it without complaining about anything in no time. I was confident they would send back the app for us to fix some stuff, but no. Our app was live.

Do you think this second app was a success? It was not. Because we spent so much time perfecting it. We did not even realize how much stuff we had put inside. Working at big corporations has taught me so many things. We were collecting crash logs, analytics, event tracking, logging, A/B testing, and so on. We built so many modules for the app that were not needed. We could not reach enough users we were hoping to reach and got frustrated with all these features we did not even use. Then some packages we did use got updated and broken backward compatibility. A lot of time went into just keeping packages working or removing them. Then other projects started consuming most of my day, and we put this app on the shelf.

Good news? I have learned my lesson this time. It is hard to let old habits go. But believe me when I say this: You will never have a bigger enemy than perfectionism. There are times you need to make sure things look as close to perfect as possible. But that is usually when you launch an app or a service and customers heavily use it, and you want to keep supporting them. But at that point, you should already have a team of engineers making sure of this and many processes in place to keep it up and running. Until you get there, just start small. Do not let your ego get the better of you. Not everything needs to be perfect for them to be useful. Half of the apps I use on my phone have bugs. They crash on me. They do not do what they are supposed to. Even the biggest products from the biggest corporations in the world have downtime. You keep hearing that the biggest networks are down sometimes. So why should you spend so much time perfecting everything? Simple answer—you shouldn't!

Focus on the most valuable product (MVP). What does your project/service/app provide to users? Pick your selling point and make sure that is working, and nothing more. Do not spend time on too many details, too much pixel perfecting, adding too many features. Just do it!

What better way to live life than to make mistakes? What is the fun of being a human being if we are not going to make mistakes? It takes a long time to relearn how not to be a perfectionist. But believe me, it is worth to relearn as best as you can. Mistakes are the colors of life. Mistakes are when you learn the most. Mistakes are what put you one step ahead. I have made many mistakes in my life, and I owe where I am today to those mistakes. Stop being so scared and letting your ego control you. Just do whatever you want to do!

Suggestions: Stop worrying about what others are going to think and letting your ego take over your actions. Mistakes are your best friends, and perfectionism is your worst enemy. Do not let that sneaky ego ruin your life.

32. Finding an Ownership Area

Finding an ownership area is one of the best ways to establish authority as well as job security. In our industry, it is easy to find many engineers with similar skill sets. Skills are one of the easiest things that can be replaced. What cannot be replaced is the knowledge of a system. Knowing how a system works, especially if you know its history, is the most valuable thing you can offer. If you have been working on a project for a while, just knowing the other engineers who worked on it and their thinking style, you can pinpoint any problem without even seeing the code. Especially projects that have been running for a long time. There are so many little additions here and there over time. Knowing how to add new features and make sure things do not get broken is super valuable.

I know some strategically brilliant people whose first action when joining a team is to figure out ownership areas. Pay attention to who owns what. Who is in charge of infra, who is in charge of projects A, B, C? How do each of these projects support each other, and what is the dependency structure like? Is there any old system everyone is scared to touch and avoids working on? That is an excellent sign for taking ownership. If you can find a piece no one wants and learn it, you immediately make yourself valuable. This will come with job security, as hiring someone new and getting them to learn the system would be hard. After gaining ownership, you can slowly start replacing pieces with better designs. Even if you do not need to change any parts and it is working by itself, just knowing the system and being able to help people onboard or figure out what causes a problem when things start crashing is valuable.

Of course, this is not an easy route. New and highly impactful services will have many people trying to stay as an owner. You can also join those projects, but then it becomes harder to become an owner. Either the project lead needs to be promoted to something bigger, so they leave the project to you, or they need to quit for you to take ownership.

A quick example would be infra. Check out to see if anyone owns infra. Most of the time, people stay away from infra because it is a lot of leg work with heavy on-call. But the bigger the problem, the bigger the rewards. If you take over and start learning the system, you can easily become the owner. If you can own infra, the possibilities are limitless. You can convince management to give more engineers to support the infra. Infra is the backbone of everything. If there are no machines, no code will be able to run. Deployments, health checks, and so on are essential to any system. In a short amount of time, you can grow your team size and start having a significant impact.

If you get the chance to build a new system—jump on it. You can learn so much by designing something from scratch. Not to mention there is no backward history you have to ramp up on. There will also not be any on-call for a while, and if you design the system well, there should not be any on-call burden. This is usually the dream. Suppose you are the main engineer who designed most of the new system. When that is the case, you immediately become the tech lead and an essential part of the system. Again, this will come with job security as well as authority and the respect of other engineers.

Suggestions: Be on the lookout for any system without clear owners and try to take them over. If you can design a new system from scratch, that is the best. If not, look for complicated and heavy on-call systems like infra, as most engineers will want to stay away from problematic systems.

33. Do Not Redesign a Working System

This is one of my favorite sayings that I learned the hard way. Engineers love new projects and redesigning something from scratch. If you ever find yourself going through an old piece of code, it will gross you out. So many if-else checks, so many one-time solutions, so many hacky codes. Things are not going to be designed with nice interfaces; you will see people have added unrelated variables to every part of the code. Data types will carry unrelated fields, a lot of copies are going to be made of each object instead of designing nice getters and setters and passing by reference. So many things are going to be wrong, and you will be tempted to fix them.

The next time you find yourself excited to butcher a system and redesign it, please remember this: "Do not redesign a working system!" No working code is bad code. I have seen many projects in my career where engineers thought it was so ugly and could be made a lot better. You collect the requirements and start way ahead of the previous project. Probably a lot has changed after their initial version, and they had to keep adding things. Your new design is going to be a much better fit for all the scenarios. Then why would you not do it?

Because systems keep changing. Even though you are going to design a perfect system, it will be a short-lived one. If you are lucky, sometime after you finish developing the new system, but generally while you are implementing it, use cases will keep changing. People are going to keep coming to you asking for changes here and there to support their scenarios. Especially since it is a new project, it will be a lot easier to add functionality than changing an ancient system. Everyone will keep coming to you for new features. Before you even realize it, your code will start getting ugly.

Remember those if-else checks and hacky solutions added to all kinds of random places? They are there for a reason. Most likely, something started crashing, and they had to add those to fix it. There will be so much business logic living in those lines. Unfortunately, when you redesign a system, there is no way for you to be able to catch all those decisions and bugs they discovered. The only way to reach that state is to launch your service and wait for things to start crashing. The next thing you realize is that everyone is adding if-else blocks or hacky solutions to your new system to fix things.

Not to mention the extra cost of running two systems together until the migration is over, all the headaches of migrating data, and migrating users to the new system and keeping them in sync. And as I mentioned earlier, before you even realize it, your system is going to be not very different from the old one with many hacky solutions and unrelated data fields being added to random places.

One of my mentors used to say this: Junior engineers think they can fix everything. Senior engineers know when they should not touch a working system. I couldn't agree more with this statement. One thing to note, though, is that the motivations of junior and senior engineers are different. For a junior engineer, even if the new system ends up in the same place as the old one, it teaches a ton to the engineer building it. So there is still a benefit for the engineer. Since seniors already know a lot of the systems and will not learn much by redesigning an existing system, it is all about adding extra work to them that will not bring much real value.

You need to be vigilant about this. You should always weigh the pros and cons of redesigning a system and know that if a system is working, simply leave it alone—unless you are willing to go through all the trouble in order to learn a lot, and you are okay with taking the blame when everyone starts complaining about how your system is not doing any better than the old one. To top it off, it will be missing a lot of the core functionality that was added over the years.

Suggestions: Do not touch a working system thinking you can make it better. Every project starts super clean and nicely designed. Then life happens, and each line of hacky code you see is actually fixing a business logic for something important.

34. The Importance of the Culture

Culture is one of the most important parts of our lives, and naturally, it is the hardest thing to build. Once it is built, it is hard to change. Most big corporations out there are known for their unique culture. Culture at work will have a big effect on your life. It will change the way you think, the way you react, the way you take risks, and the way you resolve conflicts—every aspect of your life will change depending on the culture.

Culture at work will decide if you feel like a slave or like you are hanging out with your family or something in the middle. Each corporation tries to establish specific ground rules for building a culture for the most critical aspects they care about. And then each division inside the company adds their own culture on top of the corporate culture. Then each team inside adds something of their own, and you will be affected every day by the combination of all these cultures set in your team, in your division, and in your company.

For example, how much do you feel encouraged to break things? I have seen many different cultures on this. I have seen cultures where breaking things was considered awful. I have heard horror stories of people getting fired because they took down a system or introduced a nasty bug to the code. But I have also heard amazing stories about when things start breaking down; management comes, and kudos to you for finding the problem in the system!

The culture around encouraging mistakes will change your whole perspective and approach during your day-to-day life. I have met people who took down the whole system and then told management they should have invested more in testing or frameworks that prevent such failures. Their culture was based on the idea that processes should protect the system, not engineers. It makes sense, doesn't it? You cannot expect an engineer to be vigilant 100 percent of the time. I have seen no blame for postmortem cultures. When things go down, you need to have a postmortem to talk about what went wrong and how this could have been prevented. But no one in the meeting points any fingers. It is forbidden to judge or tell someone they should not have done something (unless it was obviously malicious and intentional). Instead, you focus your whole energy on making the system foolproof, so it never goes down no matter what people do.

Some companies encourage very long design discussions. You design every bit and piece of the system in advance, go into discussions, get it reviewed by peers, get it reviewed by legal, write a ton of tests and tools to check the health of the system, and only then you are allowed to launch everything to users, and you do it very cautiously. The goal is for the services to be up 100 percent.

Some companies encourage cultures where they promote going fast. They think there is no way to catch the bugs until you see them, and the best kind of testing is the one you do on users. Do your best and design the working prototype as fast as possible. When things start breaking, and users start complaining, fix it even faster.

Which one sounds better to you? The answer will change depending on the person, and that is precisely the point. There is no right or wrong way of doing things. If you are working at a plane manufacturing company, you would not expect a go-fast type of culture. Can you imagine a world where you are asked to prototype as fast as possible, and when there is a plane crash, you go back, collect the data, and fix it later? Not possible. For certain companies, mistakes are simply not affordable. So they build a culture around implementing so many different processes and long design discussions followed by long testing sessions to make sure the system works as expected in any condition.

But suppose you are working in a start-up. In that case, as I have mentioned in the perfectionism topic, you will be digging your own grave if you are spending months designing something and testing again and again and again. Start-ups are known to be fast-paced environments where you are encouraged just to do whatever you can to get the product out there.

The most important question is what kind of culture do you prefer at work? Do you want to be encouraged to code prototypes and ship as fast as you can, hoping things will work? Or do you want to spend a lot of extra time on designing, testing, and making sure everything is working? You should pick carefully because the culture you are exposed to day-to-day will define you in no time. It gets harder and harder to change yourself later. For example, if you have grown a feeling for security and making sure your system does not crash, and then joined a fast-paced company, you will not be able to start pushing code out all the time. Even on your best day, you will do many sanity checks before clicking "Submit."

You will hear some companies out there providing lunch or even breakfast and dinner sometimes. Some companies will reimburse you for lunch if more than two employees are joining you. Why do they do this? Besides many other reasons, it is because they want their engineers to socialize with each other. We have talked about networking and its importance. The more people you know, the more productive you can be and the more opportunities you will get to help. Food is one of the best ways to achieve this. Some companies will throw happy hours, parties, or many other team-building events to get their employees to network with each other.

Not only do cultures like this promote networking, but they also build the feeling of a family. When you have fun coworkers that you get to see often, spend time together, grab food, or go to a party together, you no longer get as much of a corporate feeling. Instead, you start loving your workplace and feel as if your coworkers are members of your family.

One of my teams in the past had a pranking culture. Any time someone went for a vacation, we would make sure a surprise was waiting for them when they got back. And they were not the quick and cheap kind of pranks. They demonstrated true engineering design skills. When our manager went for a vacation, we changed the locks on his door and hid the key in a different building. We left breadcrumbs for him to go looking for the key. There were puzzles he needed to solve, little signs he needed to realize, and people he needed to talk to. Still to this day, I remember it like it was yesterday. This was a great team-building activity as everyone came up with a different piece of the strategy and helped implement this prank.

Another time when one of the managers left for vacation, we bought double-sided tape and filled all his office floors with it. His solution was even more creative than our prank. He found cardboard boxes and just put them over the tape instead of removing it. For months to come, we laughed every time we walked by his office.

Another time we bought a pool and filled it with sand and little toys like ducks and other things for a teammate. What was my prank? I found my office turned into a boxing ring when I got back. I never realized how much attention people pay to you until then. I loved playing soccer and kickboxing. I did not even think people at work would know this about me. When I got back, there were gloves and a punching bag, and the room was surrounded by ropes for the ring. Still to this day, the memory of walking into that office makes me smile, and I miss my teammates so much. It always felt like a true family.

It has been almost a decade since those pranks, and I have changed a few teams and companies since then. But that feeling you get when you are fully bonded with your teammates is unforgettable. After all these years, we still talk to each other, and we still start so many sentences with "Oh, remember that one time we …?" I did not pick this team knowing how amazing they were; I just got lucky. But if you want to create your own luck, start asking questions about their culture. Ask the team members what they think about the culture. What do they like the most about it? Do they feel like they are with family, or do they feel they are a workhorse running non-stop?

You should also work hard to bring your own culture to work. The more colors at the office, the better it always is. We had a Middle Eastern coworker, and her favorite thing to do was cook some delicious Middle Eastern desserts and bring them to us. My mouth is already watering as I remember this. I would always take the longer path to my office to walk by her office to see if there were any desserts. We became close friends, and she would let me know what she was going to bring sometimes so I could get a bite before it was all gone.

Remember to be creative and put extra time into bringing some of your culture with you to work. Ten years from today, when you look back and think about it, you will not remember the little details, you will not remember the arguments with people, you will not remember much except for the unique stories you have collected. And when you start building a family culture at work, you will soon realize that more and more people are joining you. So go ahead and schedule a happy hour, plan some contests, maybe prank people, cook something unique to where you are from, or any other way you can think of to build meaningful experiences.

Suggestions: Pick companies that are aligned with your cultural expectations and keep adding something of your own to the culture until you feel it is not work anymore but more of a family.

35. Ambiguity

What is ambiguity? Something that does not have a single clear meaning. In the tech world, it refers to dealing with unknowns. For example, let's say you are about to start a new project. Your team has decided to provide a new in-memory data store. Is it going to be fully in memory? Or will it write a copy to the hard drive for the purposes of not losing data? Will it keep replicas to protect data or increase read speed? How many replicas? Is there going to be a master partition? How will it know which machine should become master? Is it going to be a consensus-based algorithm? If there is a master, will there be only one? Or will you choose a partitioning algorithm to partition your key space? Will there be a different master for each partition? Will masters synchronously write the data to the slave partitions before returning back to the user? Or will they acknowledge the changes and then put it on a queue to write to slaves later? What if the master is not accessible? Will another slave become master? Or do you have a waiting threshold hoping the master will come back online? If one of the partitions died, are you going to initiate a new machine to start copying all the data? What happens if your partition comes back online after you have already copied data to another machine?

So many unknowns, right? Maybe these unknowns scare you and your team, so you have decided to use something already built. Are you going to go with Redis? Or Memcached? Or are you going to start using a cloud service provider like Google Cloud? Even when you do not implement something yourself, so many choices will appear all of a sudden. You will start the discussions. Everyone will have some input. Everyone will walk away with different understandings. Let's say you believe the agreement was to go with Redis, but other involved teams thought you were going to go with a service provider instead. Or you have talked about the abstractions you are going to be building on top to provide value to other teams, and they do not even know their requirements fully yet. You talk to them one day, and they say they need it to be super-fast at writing new data, so you go with writing to a master and master syncing data later to the slave partitions. Later they came back and tell you consistency is more important than everything. There can never be different partitions storing different information. Or you have agreed with eventual consistency, and all of a sudden, they started receiving millions of queries per second. They cannot afford to wait for eventual consistency, and they start pushing for strong consistency.

Do not worry if you do not get all my references. These are specific to distributed data stores, and you would know this many details if you have actually worked on implementing one in your career. But I am sure you get my point: so many unknowns. How do you navigate the waters when you do not even know which direction you should be heading? This is one of the hardest parts of being an engineer. I remember when I was in high school, I would read open-source project codes and think, *Wow, it is not that different from my code*. I was wondering what being a senior engineer meant since the code they write still consists of if-else blocks. Of course, there is much more to being senior, but the ability to drive ambiguity is one of the top ones.

What makes an engineer senior is not necessarily their code looking so much different than the juniors' but more on knowing how to approach problems, the ability to see the potholes you are going to be falling into if you choose a specific route. The more experience you have under your belt, the more you will know what parts of a system people might misunderstand. You will be better at clarifying requirements. You will be good at chasing after dozens of people, asking the right questions, and bringing everyone to a consensus. The ability to do research, provide different solutions with pros and cons, and help the team come to a consensus is going to be the hardest part of your life. Senior engineers are not even expected to write too much code. Instead, they work more on influencing people to make the right choices. But how can you make sure the right choices are being made if you don't even know which one is the right one?

You cannot ever drive the ambiguity so well that the risks go down to zero. But the better you are driving ambiguity, the closer your project's risk is going to be near zero. Shall we talk about a few tactics to drive ambiguity and not end up on the wrong path?

Let's start with what you should not do. You should not keep spinning the same circles again and again and get stuck. If you notice you are having the same discussions repeatedly and end up in a freeze and cannot decide how to move forward, that is the worst outcome. A system working for half the scenarios is always better than a system not working at all. You can start by following my previous suggestion of starting an evolving design document. What if you don't even know which designs to mention, and there is nothing you can put in that document yet? Well, figure out a name for the project or at the very least a description and start a blank document and share it around. The nice thing about evolving documents is they do not need to start at an advanced level and keep evolving. You can literally have a blank document and call it a design document. Name the headline for the first section ***Requirements***.

Start involving tech leads from the other teams who will be your users. Ask them what they need and how you can be helpful. Do not worry if the requirements change later because they always do, but this is your starting point. This is where your project gets the first drop of water that is going to help it flower later. Collect as many requirements as you can. Feel free to ask them if there are any other services they currently use to satisfy these requirements, what is good about them, and what is not working for them. Document all of these next to requirements. Comparative analysis will help you come up with a complete list of things you can provide to people and avoid implementing a service that does exactly the same thing as the other ones out there.

Stick to writing as much as possible. Getting together in a room full of engineers and discussing things will always produce more fruit for your project. But if you do not start writing them down, you will forget, and the other people will forget. You will find yourself having the same arguments again and again, and you will not remember why you have chosen a certain path. So have as many meetings as you like but always send the summary emails after the meetings and include these discussions in your evolving design document.

You have started having some idea of what you should be building. But now you are headed toward conflicting requirements. There are some requirements that simply cannot work with the other ones. If you end up in this, you need to start working with the involved parties to figure out which ones are must-haves and which ones are nice to have. By eliminating nice-to-have features, you will be able to resolve most of the conflicts. There might be a few conflicts still left that teams are thinking they are a must have.

What can you do if you find yourself in this situation? You need to use your expertise and get help from other teams, coworkers, or mentors on which one they believe is more important. It is sad to admit, but not every team is equally important in your career. Let's say the sales team is going to use your datastore to power some of the charts, and the infra team is going to use your datastore to collect health metrics from machines and use these data to decide on the health of every machine your company owns and route the traffic accordingly. This is not an easy decision, but in my experience, there are many visualization frameworks, and if it does not fully fit the sales team's expectations, it is not the end of the world. But again, it depends. If there is actually a functionality that the sales team needs that will increase the sales by X percent and the infra team already has something in place that kind of works, then the equation changes completely. You need to put together all the use cases of your service and start prioritizing which teams you need to help with first.

The good news is it is never too late for other teams to adapt to your design. Suppose you implement your service, and everyone sees it is performing a lot better than the alternatives. In that case, the other teams will try to become your customer as well and start giving up on the things they believed were must-have features before.

This is another thing to keep in mind when you are all excited and start running to people advertising your soon-to-be amazing service; people are going to be skeptical. They will still help you come up with use cases, but they will not base all of their future on your success. They will have a backup plan and wait to see if you are delivering on your promises and only onboard fully after you have proven the service. Do not believe everyone you hear and try to provide everything they ask.

Now you are at a point with requirements, use cases, some prioritization, and a few lists of the pros and cons of each decision. Your design document has already evolved to an amazing place. It is time to start comparing different solutions and which one would provide the most benefit. You need to start a new pros and cons list for each solution and, if possible, put them side by side and show the differences and how much coverage they each have over the requirements. At this stage, you should start setting some deadlines for making choices. As I have mentioned earlier in the topic, a half working system is better than a not working one.

Every day you should achieve some form of progress. If you have not made any progress, it can easily turn into stalling, and you might find yourself in a decision freeze. After setting a deadline, keep doing your comparative analysis and pick the best outcome. It does not have to be the best outcome possible, just the best from the options you have in hand. Do not go on a wild goose chase trying to find the best solution in the world. You will only lose time. If you tend to do that, please read the "Perfectionism" topic again. Perfectionism will always be your worst enemy.

Do not forget to post updates about each of these stages. Keep people informed. The more informed you keep them, the more trust they will have in you and the more they will be willing to help you as they see things are coming together.

Have you found yourself arguing over what is the better solution and unable to figure out which one to go with? Use the "Tiebreaker" trick and bring an odd number of engineers to the meeting and just go with the majority. Learning to unblock yourself in these kinds of situations is one of the greatest skills you can have.

Now you have a design in mind; some choices have been made, a good list of requirements, etc. What is next? Prototype! Do not wait until your evolving design document has evolved to perfection. The more design choices you put into the design document without testing them first, the more you will be cornering yourself for a possible failure. Maybe you have built the best design ever and coded everything perfectly, but then finance did not approve your request to use a few thousand machines. Now your project cannot perform at the scale needed from it. How do you even know how many machines you will need without prototyping? You will not. The best thing to do at this stage is to write a bare minimum prototype that just works.

Once you bring the service up and running, people will trust you more. And you have just built your first test framework. It is never a good idea to keep coding things without testing them first. Once you have a fully functional prototype, now you can start adding features one at a time and keep testing them as you finish implementation. Especially if you have more team members working with you on the same project. Providing a platform where they can test their changes is invaluable.

You might have the whole design in your head and keep coding without needing to run it, but most other engineers working with you either will not have the whole picture in their head, or even worse, they will have a different picture than the one you have in yours. Providing a working prototype prevents all kinds of ambiguity conflicts you might face down the line.

This also gives other teams an opportunity to cross-check your promises. For example, it is easy to say, "I will provide a service that responds to requests under 15ms," but if you finish the prototype, and it is taking 200ms to respond to requests, you will not be able to bring it down to 15ms later—unless you have made a terrible mistake like forgetting a "Sleep(150)" or something like that in your test code, which I hope you never do.

You have a working prototype, a good idea of the requirements and use cases, a nice backlog with clear features that need to be implemented, what is next? The good news: you have driven most of the ambiguity away. Now you just need to repeat the steps above. Implement more features, test them out, let people use them, advertise, and keep repeating.

It is time to look at your achievements and enjoy diminishing ambiguity and start getting excited about more ambiguous projects. Ambiguity is something you get scared of in the beginning. The more you get used to driving ambiguity, the more you will start feeling excited about it. Because even though driving ambiguity is one of the hardest things to do, it is one of the most rewarding and the most fun. No one enjoys reinventing the wheel again and again. The more ambiguous the projects you are working on, the more cutting edge they are.

Suggestions: Keep fighting the ambiguity. Start an evolving design document, collect requirements and use cases, prioritize, find solutions that fit the requirements, do comparative analysis, prototype, and repeat until ambiguity disappears!

36. Hold That Question

My first team at Microsoft was the Content Delivery Network (CDN) team. If you have never heard of CDNs, they are geographically distributed services put in front of other backend services to do essential things like load balancing, caching, service health checks, and protection against unwanted things like bots or DoS (denial of service) attacks. Let's not dive too much into details. However, it is important to know they need to perform in minimal latency and are expected to be working 100 percent of the time. When they go down, everything goes down.

As a new grad who had just started working in a different country and had never worked on such a complex system, my days were driven mainly by fear. Most of the time, I felt I would click the wrong button and take everything down. This led me to double or sometimes even triple-checking every action. Without exaggerating, I would check every decision multiple times a day and ask my senior peers to confirm things. This quickly led to people not wanting to work with me as they had to spend too much time answering my simple questions. I was lucky to have good mentors who were great at giving constructive feedback. After a couple of weeks, I started to implement some rules for myself to follow before asking a question.

1. Do a Google search (sounds obvious, but you would be surprised how many questions engineers ask each other that could be answered with one Google search).
2. Do another search in internal documentation.
3. Search code examples. Make a search in the codebase, and if you find anything similar, investigate the version history to see how previously engineers have implemented similar features. You can also see the notes they added to their commits.

These steps might seem easy, but give it a try. You will be surprised how many of your questions can be answered quickly. I have seen people respond with a link to a Google search, which is humiliating. And the best part of having a process like this is it gives you the opportunity to mention your process before asking a question. People are a lot more receptive to questions when they know you did your best, and they like spending time helping someone in need vs. someone using their time instead of their own. Not to mention you learn a lot more than you originally intended when you do your own research.

Suggestion: Hold your questions for a second, build your process, and do basic searches before asking questions. If you cannot find answers, mention your process, and ask for help. You will like the answers you get a lot more.

37. Put it on My Tasks—Unblock Yourself

Have you ever found yourself spending many iterations on a code review because someone feels strongly about changing something, abstracting some part, or simply changing styling? Have you ever finished coding, spent so much time testing it in production, and right before merging the code, someone asked you to do a small change? If you have found yourself getting blocked regularly by other people's requests, you are not alone.

It is not only code reviews, either. Sometimes you end up in similar situations during design discussions or even in regular meetings. Someone thinks something should be done differently, and you keep thinking about how to say "No" so you can focus on the topics you care more about. But you do not want to disrespect them or make them feel that their point is not valuable. You do not want to be the guy always pushing back or disregarding feedback. Especially if there are many more people in the room. Saying "No" to someone discourages others from sharing their opinion. Why would anyone want to bother giving you feedback if you are not interested in it? They will get disengaged and start dreaming about something else and not even listen to what you are presenting.

At the end of the day, you are doing the design review, regular meetings, or code reviews to get people's input, right? It is never a good idea to make someone feel that their input is not valuable or wrong. But you also do not want all input to stop you from what you are interested in focusing on. If you keep ending up stuck in a discussion you do not believe is worth the time, what do you do? Do you simply keep scheduling follow-up meetings to figure everything out? Or do you keep turning people down?

Before sharing a trick to help you with this kind of situation, let's turn the tables. You want to focus on what you believe are more important parts, and they seem to get stuck at a small detail (in your perspective), and they hijack the whole conversation until they hear, "Yes, I will do it." If I was the one presenting something, would you waste your breath on something that you thought was not even important? Would you speak up in front of everyone and make them think your suggestions are terrible? I do not think so. When people speak up, they believe what they have to say adds value. This is where the conflict starts: They believe something is important, and you simply do not.

What would you feel if you were the one giving me feedback and I simply made you feel it was not valuable? Would you like to hear that what you believe to be important is actually total garbage? Would you like to hear, "No, I will not do that"? What is the outcome you would expect when you speak up? You would want to be heard and acknowledged. Even if it is not done the way you hoped, at least you would want to know it was considered, some time was spent thinking about it, and then some better way was chosen. Of course, you would not expect all your suggestions to be amazing and all of them to be done 100 percent of the time. But at the very least, you would want people to show you the courtesy to care.

Believe it not, it is no different for other people. You should never shut someone up if they care enough to try to help you. That shows they are trying their best to be there for you. Of course, I am not talking about someone you recently pissed off. If someone is hijacking the conversation intentionally just to make you look bad, that is a different story. But I would still suggest never assuming that someone is doing something intentional with malicious intent. That happens so rarely that I do not believe you should ever worry about such a thing (again, I am assuming you are not an asshole to people every chance you get). And even if they are trying to hijack the conversation, fighting with them makes both of you look bad. Even if you are dealing with a malicious person, the best thing you can do is to hear them out and show them you care. The more you do that, the more they will start liking you and not doing anything malicious.

Since I believe you should treat both nice people trying to help and malicious people the same, let's talk about what you can do to handle the situation. Because yes, I strongly believe you should value everyone's input, but that does not necessarily mean spending a lot of time discussing the ideas of everyone in the room. Or spending days trying to agree to every comment made on your code review and doing everything the way people ask. If you go this route, your design discussion is going to get ten times harder, and you will not be finalizing anything. You will not be getting the feedback you were hoping for on the parts you deeply care about.

How can we achieve not letting people hijack the meeting or code review while making them feel heard and valued? Write it down! Sounds simple, right? Never underestimate the power of writing things down. If someone has a suggestion during a meeting, simply write it down in the meeting notes and thank them. Show them you care. Writing it in your notes means, at the minimum, that you will have to read it one more time. Even if you think it is not important and delete it later, you will have to read it one more time to know which part to delete. But that is the minimum bar; you will likely read it again with a fresh mind later and understand more what they actually cared about. Especially if you have slept on it. It is amazing how your brain understands things a lot better when you think about it again after a good night's sleep. There have been many instances where I thought people were talking about parts that I didn't think were important the first time I heard it, only to find out they had incredible suggestions. When your mind is under a lot of stress and trying to accomplish a successful meeting, you will not be too receptive to ideas you have not thought of before. But when you read it with a fresh mind, you will understand it a lot better.

What can you do better than writing them down in the meeting notes? You can write it down in the design document. Either as a suggestion like "Consider doing …." or as a to-do like "TODO: Consider adding …" Design documents are a lot more formal and shared with so many people. Writing something into your design document to consider later will mean a lot more than writing it in the meeting notes that you may never open again.

Can you think of any other way? Do not cheat by looking at the title. Yeah, you have guessed it right. Create a task with the suggestion you have received. It is common for engineers to create tasks for everything they want to follow up on later, or for the things they do not have the time for right now. If someone is asking me to change the data structure to be an interface and implement the interface in a different file, that is good advice. But it is not urgent enough to implement immediately. Unless you are planning to share your code with other teams and they will start using it immediately, you can do it tomorrow or the day after. It completely depends on the priority of the thing being asked. But for our purpose, the most important part to note is that it is not something that needs to be fixed immediately. By creating a task, you are giving your promise to follow up and make the other person feel heard and valued. And you are unblocking yourself as well.

What happens if you created a task but could not get to it tomorrow because something higher priority happens? Nothing! This is our life. At any given time, we have way too many tasks to work on and not enough time to work on them. If you legitimately are busy with more important tasks, no one will blame you. If you are kind enough to follow up and explain what happened and why you cannot prioritize it right now, they will like you even more. They will see the effort you are putting in and that it is not something against them personally.

Be cautious about abusing this. This trick is somehow common in corporations, so whenever someone does not want to do something, they make a note or create a to-do to unblock themselves, knowing they will never look at it again. You might get away with this a couple of times, but engineers are smart people. If they realize you are abusing this to ignore what they are asking, next time, they will keep arguing during the code review or the meeting to make sure you have actually listened to their suggestion.

Go ahead and give it a try the next time you feel someone is side-tracking during a meeting, design discussion, or code review. Unblock yourself while respecting others and focus on what is important in your head right now. Write it down in the notes or design documents, or simply create a task and follow up later.

Suggestions: Make sure people feel valued when they share suggestions by writing them down to notes, design documents, or simply by creating tasks. If you cannot get to the task soon, it is always a nice gesture to reach out to the person and explain why you cannot prioritize it right now.

38. Do You Have Some Time? Ask My Calendar

Hopefully, as you are reading this topic, you are already thinking, *Oh yeah, the calendar is my best friend*. If not, let's talk about this. Using the calendar is like having an executive assistant without actually paying for one. I cannot even imagine what my days would be like if I were not using a calendar.

Calendars have evolved a lot in the last decade. They used to be a place to set some reminders to notify you at a given date and time. But with the collaborative features added to them, now they are a true lifesaver. They have many features like booking a meeting room, seeing other people's availability, syncing everything to your phone, and some even have suggestions on what time to book meetings to disrupt other people's day as little as possible. There are even plugins for the common IDEs to sync your calendar and make sure you do not miss any notifications.

Especially in big corporations, they also come with sync tools to update messaging applications and other internal tools. Like if you are currently in a meeting, people will be able to see you are not available when they try to message you. Instead of hopelessly waiting for an answer or not knowing if you are available, they can just see your schedule. What is the alternative? You ping people every time you need to sync about something, ask when they have some time, check if you are available at a given time, and repeat until you find a time that works for both of you. Now I just open up my calendar app, see their schedule, and book something at a time when we are both available. I add a note on the description on what I would like to sync about, and not long after, I hear back if they have accepted the meeting or requested a different time.

Calendars have been my best friend for a long time at work. But I did not utilize them as much for my personal life. What is the problem with this? Sometimes you have a doctor's appointment; sometimes you need to attend an event after work and need to leave early with the traffic and so on. Conflicts would arise whenever I looked at my work calendar and accepted an event only to find out I could not make it. The more I started incorporating calendars into my personal life, the less stressed I was about conflicts.

You can even use some tools to sync your personal calendar to your work one, so when you have a doctor's appointment in your personal calendar, it blocks the time on the work calendar, so no one tries to invite you to a meeting. If you cannot find a good tool to sync all the events, it only takes a few seconds to do it manually. Most phones today support duplicating an event on the calendar app from one calendar to another. You can just duplicate the personal events into your work one in a matter of seconds.

I might have even gone too far with using the calendars by convincing my partner to use them too. Our plans would always fall apart because she would talk about what she wanted to do without picking a day or time. She would be like, "I want to check out that restaurant." Then on a random day, she would be like, "Do you want to go to that restaurant I have been talking about?" and I would find myself going through my calendar to give her an answer. Now I tell her, "My calendar is up to date. Feel free to grab any time."

If you are planning to try the same, just be ready for a small argument. Partners do not seem to like the idea of booking time on your calendar. But insist on trying it a couple of times. When they start noticing the convenience and the amount of time you save from talking about when you both are available, etc., they will love it. Now both of us just look at each other's calendars and book whatever we want to do, and in a few minutes, you have already gotten a response if they accepted or not.

There is no more forgetting the plans you have made, no more arguing about when to do something, no more long discussions about what to do. Instead, you see it coming as soon as you check what is next on your calendar. And you get a friendly life-saving reminder to remind you in case you get too busy.

Suggestions: Get better at using calendars. They are like having an executive assistant without actually paying for one. Try incorporating them into your personal life as well.

39. Setting Focus Blocks

Do you have a hard time finding time to focus? Are people messaging you, and you feel bad if you don't respond right away? Are people scheduling random meetings all over your day? Or even worse, are there meetings with one to two hours in between, making it hard to focus on something knowing you have to snap out of it before you really get started?

These problems are common for all of us. We have talked about mastering calendars and using them to your benefit in the previous topic. What can you do to make sure your whole day is not randomized? Keep using your calendar to create events with one difference. Do not invite anyone else but yourself. Put some time in your calendar that you are hoping to complete some work and do not want to be distracted. There is no point in getting frustrated by people scheduling things randomly. They do not have a lot of free time either, and they schedule things for wherever they see an empty spot on your calendar that fits their schedule.

Some calendar tools automatically support functionalities like starting business hours, ending business hours, lunch break hours and focus block hours and take those times out of your calendar so that people do not invite you to something when you are not available. Check out your company's calendar tool to see if it supports this. But even if it does not, it is super simple to do it yourself. It is nothing but some sugar coating around scheduling meetings. Book a meeting with only yourself from 8 a.m. to 10 a.m. to repeat every day. If you want people to know you are out of work, change the privacy to public and make the title something obvious like *Out of Work* or something funny like *Busy Snoring*. If you do not want people to see the details, then mark the privacy setting as private, so all they see is a dark-colored meeting that you are attending, so you are busy.

Do the same for the focus blocks you are interested in. If you are an early riser, block 9 a.m. to 11 a.m. as a focus block; if you are more productive after lunch, book some time then. Again, you can make it public or private, depending on your choice. This will help people know they should not book some time from your calendar during your focus block. We have talked about chat applications with calendar sync features and how they can automatically update your status on a chat application, so whoever is messaging you can see warnings like *In Focus Mode* or *In a Meeting*, so they do not expect an answer from you because they will either see your status directly or receive a notification letting them know you are not available after they message you. Do not be afraid to use the tools to your advantage.

One thing to be careful about is not booking the same times every day. I suggest booking different focus blocks for each day for two reasons. First, so people can pick a different day to meet you at a different time. You might be an early riser, but someone else might want to have their meetings in the mornings to leave their afternoons empty for focusing. This gives them the ability to choose different times of the day. Secondly, to make sure it is not too obvious. If someone sees a two-hour block on your calendar repeating daily, they will know you are not actually busy, and even though some people will respect this, some people will think the meeting is important, so it outweighs your focus block and they will ask you to come. It goes without saying, if you set up a repeating event, make sure to unmark Saturdays and Sundays. It looks funny when you see someone's calendar and the same time is booked seven days a week.

Another thing to be careful about is to make sure you are leaving enough free time. Finding times that fit everyone's schedule is already hard as it is. If you block most of your day for focus blocks, people will have a hard time inviting you to meetings, and they will get frustrated. I did this without noticing once. I set up my business hours to start late because I wanted to make sure I had a good sleep in the morning and no surprise meetings showing up before I was at work. Then I decided to wake up early to finish my work earlier to enjoy the sun before it was completely gone during the short sunlight days of the winter. I updated my business hours to end early but forgot to update my starting time. With the added focus blocks in my days, people started complaining that I was never available. I checked my calendar and was shocked. It looked like my business hours were something like from 11 a.m. to 4 p.m., and there was a lunch break and focus block in the middle. Make sure there is always enough time left in your day in case people have questions or need to set up a meeting with you.

Suggestions: Utilize the calendar tool to set up starting and ending business hours, lunch breaks, and most importantly, focus blocks, so people know when not to schedule meetings or ping you directly.

40. Make People Think It Was Their Idea

You go into a discussion, you keep suggesting some ideas, but someone else keeps suggesting another one. You keep defending yours; they keep defending theirs. It seems like no one is happy with how the meeting is progressing, and it is getting harder to reach a consensus. You are getting tired and starting to question whether it is worth wasting time doing what they have suggested. Finally, you start considering if their suggestion would work just to get it over with, and even if it is not up to your ideals, you agree to it just to end the discussion. Has this ever happened to you? I have noticed this happening to so many people in so many discussions I have joined.

Why is this happening? Because people let their egos get involved in the decision-making process. We have talked about how to reach consensus in conflicting situations in previous topics. But all other tactics still do take more time to apply. There is a tactic that prevents you from ever ending in a conflict in the first place. What is the reason people's egos start getting involved in the process? Have you noticed something in the above paragraph? We have said "your" idea and "their" idea. It is not idea A and B; it is not the better or worse idea. It is simply yours vs. theirs. When people start feeling they are on a side, they want to support it no matter what. They sometimes stop thinking which one is better, or sometimes even though they realize it is not as good, they still defend it because if they do not, they are afraid people will think their idea was not that good.

What can you do to prevent the discussion from entering *us vs. them* territory? You can simply not suggest any idea of your own and lead people in the right direction and help them suggest the idea you are trying to achieve. It is easy to describe but hard to achieve. Unless you get good at this, you cannot simply pull this off on the spot. But the more you practice it, the better you will get.

How can you help someone reach the same idea as yours by themselves? To figure that out, you need to ask yourself how you reached that idea. Engineers are smart people, but none of us suddenly arrive at an idea without any reason. If you think an idea is good, it is likely because you did research and know certain things that led you to believe it is a good solution. Even at times when you have not done so much research, you are still using your intuition, which is mainly based on your past learnings. It is still backed by experience gained from somewhere.

Suppose you did enough research to come to the conclusion that an idea is good. Why should someone else not reach the same conclusion provided the same data? Most of the time, they do. And if they do not, you should consider thinking about their idea more in detail. Because if they have access to the same data you do and reach a different conclusion, either they know something you do not or you are missing something. In both cases, you should take a step back, and instead of pushing your point of view, you should try to find that missing piece. Regardless of which idea you end up choosing, you will learn something new.

Since we established that if you feed the same data to a smart person, they are expected to reach similar conclusions, how do you lead them there? By asking good questions. Simply avoid suggesting any solutions and keep asking questions. But make sure they are directed toward the ultimate goal.

Let's build a quick example to make it more concrete. Let's assume you are trying to decide which kind of servers to rent from the cloud for your team. These are the choices:

A. 1 processor, 32GB of memory for $5 a month
B. 2 processors, 32GB of memory for $6 a month
C. 1 processor, 64GB of memory for $6 a month.

Do not let prices make you think this will be an easy decision. You will need around 5,000 servers to be able to run your service for all your users.

Now, suppose you go into a meeting and suggest renting server type A. In that case, I am almost certain someone will think B or C is a better choice because by paying $1 extra, you either get an extra processor or a whopping extra 32GB to keep stuff in memory. Let's question why you want to pick A when the additional hardware is not much different in price. If you are planning to run a single-threaded service on the machine, then there is no need for a second processor. You will be simply paying for hardware that will not be utilized. How can you make someone come to the same conclusion as you? Here is a quick conversation:

- You: Hey, Henry, we are trying to pick a server type for our project. We will be needing around 5,000 machines, and I would love your help. Here is the pricing chart. I cannot decide which one.
- Henry: What are you planning to use them for?
- You: We are planning to run our in-memory cache with them. Do you know if we have run any performance analysis on how it performs on single vs. multiple cores?
- Henry: I believe it is single-threaded, so I do not think there is going to be any difference.
- You: Oh, that is a good point. If that is the case, it does not make too much sense to go with option B, does it?
- Henry: I do not believe so.
- You: Nice. We are planning to keep the top-visited pages in memory for faster user access. I think the top 100 pages should be enough, I have checked, and we limit page size to be less than 1MB. What do you think?
- Henry: 100 might work. But we need to make sure we plan for the future in case we decide to increase it later. Since page size cannot be more than 1MB, if we go with option A, we can support at least up to 25,000 pages with what is left after operating system usage.

- You: That is a great idea! It is the cheapest one too. We will be saving $5,000 a month. I cannot thank you enough for bringing up the single thread. You have just saved us a lot of money.

Yes, the conversation above is a bit exaggerated to make a point, but I am sure you have got what we are trying to show here. Have you realized that you have not made any suggestions above? It was all Henry's idea. But you already knew he would reach that conclusion provided the data. All you had to do was provide the same data that made you reach the decision to use A and let Henry come to the same conclusion. You got the choice you needed while Henry feels he contributed. If someone else were to come and say, "Let's do option B," Henry would start arguing with them even before you do, because now, he owns the idea, and he will want to make sure everyone knows why. The best part? No egos are involved because you both are on the same side.

Now, what if you wanted to go with option B? All you have to do is ask one more question like, "Hey, we are also planning to run a service to check user passwords. Do you think we should go with option B to get more processors for a small price difference? Or keep it separate and buy different machines?" Henry would jump on this to say, "It would cost a lot more to buy more machines; let's run them together."

Now, if this is such a good tactic that helps everyone come to the same conclusion and prevent conflict, why does everyone not do it? Because most people's egos do not want to let Henry take the credit. Most people fear that if you let someone else suggest an idea and you go with it, people will think it was their solution and not yours; hence they are a better engineer. But this cannot be further from the truth. At the end of the day, no one cares who suggested all these small decisions. The most important thing is how well you have executed and driven projects. Not if you were the one who has made every decision in the process. We have already talked about the importance of giving credit. The more you give credit, the better it reflects on you.

You should always think like you are driving a car. Just because you are in the driver's seat does not mean you need to be spinning all the wheels yourself. You do not need to worry about how gas is pulled from the tank, filled into the cylinders, burnt, and generated power, and how that energy was transferred to the tires. Instead, you are responsible for stepping on the throttle, ensuring the direction is set correctly, and letting the car do the rest of the job.

The next time you are trying to convince someone to agree with you, do not start with telling them what you hope to achieve. Instead, give them the same data that made you reach that decision and keep asking questions that lead them in the same direction, and wait for them to be the ones suggesting the same idea you had in mind.

Suggestions: Do not tell people what your solution is. Give them the same data that helped you reach that solution and ask leading questions until they reach the same conclusion.

41. Being More Active on Interviews

How often do you interview candidates? Never? Once a year? Or regularly? Hopefully, your answer is regularly. Because interviewing is an essential part of our job. It is one of the easiest ways to have a huge impact on your team (if you are working for a company where hired people join your team) or it is a huge impact for the company (if you are working for a company where hired people get assigned to other teams). It does not matter if it is for your team or not; interviewing has many advantages for you.

First, you get to say yes to people meeting your requirements. Everyone has different requirements. Some people prefer technically strong candidates, some people prefer great communication skills, some people prefer high-level system design, some people prefer hiring leaders, and some prefer hiring coding machines. There is no right or wrong choice here. Everyone is good at something. In a perfect world, you would want to hire someone who is great at everything. But we do not live in a perfect world, and it is not possible to be great at everything. The more involved you are in the interview process, the more you can speak up and make sure people who you think deserve a chance to work with you and your team should get hired.

Secondly, work culture. We have talked in many different topics about how important work culture is and how it is so hard to build. Everyone hired to the company comes with their own work culture. If your company is hiring at low rates, this might not become a big problem because engineers are great at adapting. If your team likes moving fast and fixing things when they break, and you hire someone who likes spending three months in design discussions, they will either soon adapt to your team's culture, or they will feel like they do not belong there and leave. But when the hiring rate is high, your team's ratio of new hires vs. old ones is going to be a lot higher. If you are working at a well-established corporation with a solid work culture already set, they might still adapt to align with the company.

But the more the number, the harder it is. You might find yourself in months-long design discussions soon if most of the team starts believing this is the right way to go. Again, there is nothing wrong with months-long discussions. It is all about the pros and cons. But you will end up feeling you are the one who does not belong there and start looking for a new team who has a faster pace.

One of the golden outcomes of interviewing people is getting better at interviewing yourself. Turning the tables will help you see what people who interviewed you in the past have seen. When your mind is busy solving a question during an interview, you do not get a good chance to focus on how you are perceived from the outside. But when you are the one doing the interview, you will quickly start picking up on things.

For example, in my early career, I would focus on solving the problems interviewers were throwing at me. I thought the more questions I could solve, the better the results were going to be. After I started interviewing people, I realized this is not true at all. When you are interviewing someone, you are looking for a teammate, not a know-it-all machine. You get dozens of signals from every word they use, every choice they make, every question they ask. You basically get a speeded-up version of working with them on a project.

Assume you have a teammate who is super smart. You talk to them and mention the problem, and they disappear for a month and bring back the perfect solution. Does that sound like something you would like? What if they misunderstood you? What if they have not clarified anything? What if they are condescending and start telling you this was not worth their time? These are the things you cannot get by having someone quietly code something. The more they talk, the more you realize how good of a team player they are. The more they ask questions, the more you figure out how good they are at clarification.

Since I started interviewing people myself, I stopped treating my interviewers like an interviewer. I started treating them as if they were my teammates who were there to help me, who are supposed to work together with me, building a solution that works by helping each other. The less you treat them like someone you need to please and start treating them more like someone you enjoy working with, the more likely they will want to work with you in the future.

You learn all these signals by simply interviewing people. Do not worry if you do not get that many signals initially. This is a skill you build in time. In your first interview, you will be stressed, not knowing how long your question is going to last; you will keep fearing what if they are too smart and burn through all your questions in fifteen minutes? What are you supposed to do in the remaining time? The more you interview, the more you will learn what signals to watch for, which topics to choose from, what questions to ask, and most importantly, what behaviors to observe.

One thing I also like is when candidates start questioning you. They are curious to learn what you are working on, how much you like it, how much chance you get for self-development, what your day-to-day life is like, and many more questions. The more questions they ask, the more you start thinking about these questions. Normally, you talk to your coworkers, who already know what you are doing, so simple explanations are enough. But when you are talking to an outsider, you need to phrase your sentences both not to share any internal secrets and to make sure they can understand without knowing everything going on internally. This gives you a unique opportunity to see how you perceive your life at your company and get better at seeing your projects from an external perspective.

Suggestions: Be more involved in the interview process. You will help your team/company to hire great candidates, preserve the work culture, and learn so much by doing it.

42. What to Expect from a Brand-New Manager

Managers are one of the most important people in our lives. Good ones will lead you to success, while bad ones might ruin your career. If you are changing teams, you usually get a chance to talk to the manager and get a feeling of what working together will be like. But what if an engineer gets promoted and becomes your manager? There are many reasons why this might happen. Your team might be growing, and your current manager might start smaller teams under them and promote a few engineers to become managers. Your manager might quit his job or transfer to a different team, and instead of hiring a new manager, one of your peers might be promoted either temporarily or for the foreseeable future to be your manager.

What should you do if one of your peers becomes your manager? Should you try to run away? Or should you be happy about it? Well, it depends on many factors. I have seen examples of this turning into a great relationship with the new manager. If it was someone you were getting along with, it is great news. You might form a much closer relationship than the one you had with your previous manager. Since they will know the internal dynamics of the team, it will be a lot easier for them to know who is who, who does most of the work, who spends more time in design discussions, etc. They will have more insight into your team than your previous manager since they were one of you not long ago. You might feel more comfortable taking problems to them. You might find a chance to be more open. You might feel more secure knowing they have got your back. Or you might feel completely opposite to these things. It will heavily depend on the person and the current situation.

Regardless of how good they were as an engineer, you will run into a few issues with them becoming a manager. Do you remember being new to engineering? Do you remember how hectic your first project went? Do you remember feeling that you had no idea what you were doing? Exactly! Your new manager has never been a manager before. They will need time to learn, develop themselves, and become a solid manager.

The most challenging change a new manager will face is perception. Before becoming a manager, they had a sense of control, a sense of ownership, and a sense of ability. When they thought something was important, they could simply work on it. When they thought something was broken, they could simply fix it. They were able to manage their time to their liking, start their day whenever they wanted, finish whenever they wanted, pick up any projects they would like, etc. Once they become managers, the most challenging part is going to be getting used to not having any of these powers anymore. They cannot simply fix something (technically, they can, but the more they do it, the more they are postponing becoming a real manager); they need to convince someone else to fix it. They need to prove to someone why it is more important than what they are working on. When things are falling behind, they cannot pull an all-nighter and catch up with everything. They need to convince people to catch up on their projects. And if they are leading five people, they need to do this for all five. Think about how stressful it was to be behind on your deadlines. Now what if five projects you were working on were falling behind and you had no time to work on them? How would you feel about that?

New managers will struggle with letting go of the feeling of control, not being able to manage time, and not being able to control everything. What can you do? You should find ways to help them. Do not let their panic scare you. It is just a transition, and they will soon be much better at it. But until they do, you need to make sure you are helping them. The more you help them with prioritization, the more precise your timeline is, the more you become explicit with them about your estimates and let them know how much padding there is, what is your backup plan if things go wrong, if you are planning to take a vacation or not, they will feel a lot safer. They will have one less thing to worry about.

Another hard change for a new manager is going to be not being technically involved. If they were promoted to a manager, they likely showed good technical leading skills. Being a technical lead is quite different than being a manager. As a technical lead, you are responsible for technical choices, design discussions, breaking things down to tasks, pros and cons analysis, and prioritization of those tasks. As a manager, you are responsible for setting direction, aligning with other teams, keeping people happy, and motivating them to do the rest. They will need a new technical leader to fill in what they used to do. During the transition, most new managers will spend a lot of time trying to be the technical leader as well. It takes time for them to get used to and realize it is not their job anymore. The more they see decisions are being taken against what they believe, the more they will micro-manage. What you can do is to remind them that they are a manager now and they should trust in the abilities of others and let them do all the technical driving. It will be a bumpy road, but after a few projects being successful, they will learn to trust the team more.

Learning people's passions and motivating them is going to be another area they will need to learn. Being technical lead, they used to work on high-impact projects, design the system, and people would work on the parts they were motivated about. Your old manager would help other engineers get motivated. Once they joined the project, the technical lead did not need to do too much to motivate people. They would be ready to work on it. As a new manager, now it is their job to figure out who likes what and which projects they are interested in. They will need to learn ways to keep people motivated. Sometimes everyone in the team will want to work on the same project. It takes great skills to learn how to motivate people to work on other projects without telling them, "No, you have to work on this." No one likes to hear what they have to work on. All engineers need to be motivated to like what they are working on. Again, you can help your manager by taking some of this responsibility off their shoulders. The more you focus on motivating engineers around you and distributing the projects and the tasks, the more time your manager will have to focus on all the other fires that need to be extinguished.

Visibility and expectations are going to be yet another area your new manager will struggle with. When you are working on a project, it is easy to become visible and prove your value. You have either written the code, or you have not. Your project has either brought impact, or it has not. There are well-defined metrics to measure your success. When they become a manager, they will have a hard time learning what they need to do. If everyone on the team is performing above expectations, what does this mean for the manager? Does that mean they can go on a six-month long vacation and still get a great performance rating because the team did so well? If the team consists of engineers who are in conflict with each other and every project fails, does that mean your manager needs to be fired? What is their job? How do they prove they actually did good work? How can they show the upper management they have done their job? These are similar problems senior engineers and technical leaders are facing. But as a manager, this will be of a lot higher importance because they are no longer coding. They are no longer working on projects. They are just managing people. It will take quite some time for them to align with upper management and know what metrics they should be focusing on.

Unfortunately, there is not much you can do to this aspect to help your manager because you simply will not know. It is hard for engineers to know the list of things managers have to accomplish. We all know that your manager needs to keep you happy, but how do you measure it? How does the upper management measure it? There is nothing more you can do other than help them as much as you can and try to keep projects on the right track so they can find more time figuring this out themselves.

While they are so busy learning how to manage expectations, prioritization, letting go of control, and a bunch of other new areas, they will also be pulled into many new meetings they have never been to before. One of the most important things managers are responsible for is going to the planning meetings. These meetings are filled with skilled managers. Your manager will have a hard time with these. They need to convince other managers to claim ownership of important projects while trying not to promise too much to the point the team cannot keep up with the deadlines. We have talked about how they will have a hard time knowing how much time each engineer will need for a project and also that your new manager will not know who likes working on what. While figuring that out, they will need to join these planning meetings and do their best to grab projects that are best aligned to the team. There will be many times people will think the projects are not important enough or they do not provide the skills they hoped to learn. This will take a while for a new manager to get better at.

What can you do to help them? You can follow the suggestion in the "Side Projects and Knowing Your Surroundings" topic and be better at tracking what projects are going around and helping your manager be aware of them. If you can do some of the management jobs for your manager, it will make their life a lot easier. If you can follow different projects and have a good grasp of who in your team would be interested in working on it, just pitch it to your manager. Keep giving them feedback on which projects they should try to own and who in the team is a good match for it. The more data you can give them, the better they can achieve making sure everyone in the team is happy.

Lastly, learning how to grow engineers is going to be another challenge. Great managers are good at seeing what you are struggling with and assigning more opportunities to you, as well as mentoring you to make sure you get better at them. Being a new manager, this skill will take time to develop. Your new manager might think you are good at something when you actually are not, and they might set you up for failure. What you can do to prevent this is to be vigilant yourself. Let your manager know what areas you feel insecure about and ask them to help you. The more you can take the guesswork out of the equation, the more chance you and your manager will have to succeed together.

We have also talked about the importance of mentors for yourself. Guess what? The same is true for your manager. If there are great managers you are amazed by, suggest to your new manager to go after them to make them their mentors. Or even if you do not know who a good mentor would be, just keep asking your manager who their mentor is and if they are getting the help they need. Ask your manager's manager to make sure your manager has good mentors. If not, ask them to find some. Nothing can teach them faster about everything we talked about above. A great mentor will make sure your new manager becomes a good manager in no time.

What is great about having a new manager is you will learn so much about management. Try not to see the things they are learning as pain points. See them as opportunities to learn management yourself. The less your manager knows about dealing with situations, the more they will ask for your help. The more you help, the more you learn. If you can build a good relationship with them and help them become a good manager, they will share this with their managers. The next time there is a spot for a manager promotion, your name will be mentioned. And you will suffer a lot less than your manager since you have been going through all these problems together and have been helping them.

The most important part to remember is that becoming a manager is hard. No matter how good they are, they will struggle. If you have the patience and energy in you to help them, it will be great for your career. But I need to say this: If you do not have the patience or energy, it might be best for you to flee the ship before things start getting crazy. Having a new manager requires a lot of time and energy investment. If you cannot afford it, more quickly than you realize, your career is going to take a hit. As I have mentioned, perspective is the most important thing. If you can see your new manager as an opportunity and a great challenge, you will excel. If you cannot and start seeing things as problems, run as fast as you can before you hurt your career or your new manager's.

Suggestions: Always keep in mind that your new manager has to learn a lot, pick up several new perspectives, give up on old habits, and let go of control. This can be a great opportunity for you if you have the energy to help them along the way. You both might excel in your careers if you can work together. Do your best to help them with every challenge they face. But if you realize you do not have it in you or you do not want to take any risks, it is better to change your team as quickly as possible.

43. Collecting Feedback

A long time ago, one of my manager friends told me that he believed if someone has not improved something during the latest performance cycle, they should not get a good rating. I was surprised and followed up with "What if they are already good engineers?" He said, "No one is perfect, and the worst kind of engineer to work with is one that thinks they are. Everyone should be able to find an aspect that needs improvement, and if you cannot, you are not paying enough attention."

His process was quite impressive. He would ask engineers to write down the things they believed needed improvement at the beginning of the performance cycle. After writing down those things, they needed to document what they had done to improve them through the cycle and show their progress at the end. If they could receive confirmation from other peers that they had seen an improvement, that was even better.

This process was impressive but yet hard to implement. Why? Because no one likes to admit they are not good at something. And to do it publicly and share it with everyone is even worse. I have seen a few different examples of people writing good things about themselves as a bad thing like "I overwork to make sure my projects succeed. Need to learn not to do that." Or things like "I spend a lot of time helping my peers. I should invest more in coding." Even though these sound like they are something you can improve on, they are not bad at all. Who thinks it is terrible to hear someone is ready to overwork whenever needed or that they are doing their best to help the team? As I always say, engineers are smart people, and they always find a way to make the situation work for themselves.

After working as an engineer for over a decade, do you know the most impressive observation I had? The people who are truthful to themselves and to their coworkers end up achieving the most. Yes, it is hard to advertise what you could do better publicly. But the upside is, now you have the motivation to do it. When you know everyone is aware of it and that you will be measured against it, you do your best to improve yourself. Most of us focus on short-term gains like getting a bonus or getting promoted, but we forget that we will be an engineer for the rest of our life. Which one would you prefer? To get one promotion a year earlier and then get stuck over there? Or work on improving yourself and guarantee the next ten promotions you need to get for the remainder of your career? I think when put into perspective like this, it is easy to see you are only hurting yourself by not being truthful.

What is more important to realize is that even though you might not want others to know you are not great at something, they already do. You can accept it or not, but they are good at observing, and they will quickly pick up on it. If anything, by accepting it and working on it, they will appreciate how humble you are. At the end of the day, we are all human beings, and we are not perfect. Hearing someone accept their challenges only makes me respect them, not think less of them.

If your team has a process to help you improve yourself like the one I have mentioned above, you are lucky. But what if you do not? What can you do to achieve similar outcomes? Meet one of your best friends—feedback!

Receiving feedback sometimes feels like your enemy. Especially if you are not self-confident about your job yet and scared of others also noticing this. Every time you receive feedback, you might feel someone is complaining about you, and they were the reason you missed a bonus or promotion. But the more self-confidence you build, the more you will feel like you can find a job anywhere whenever you are interested, and you will realize that feedback is actually your best friend. When you change teams or your company, your level will not follow you. People's perspectives of you will not follow you. Your failures will not follow you. The only thing that will follow you to your new job is your personality and your knowledge. If you have invested heavily in improving yourself, when you change teams or companies, you are gifting them a great asset. If you have invested your efforts into hiding your flaws, you have nowhere to run. Your current team will know all these flaws, and even if you change teams or your company, it will only be a matter of time for your new coworkers to start picking these up. You can delay the suffering, but you cannot avoid it fully.

If you are still feeling like feedback is your enemy, don’t worry. I have been there. We have all been there. But what you should focus on is gradually changing it to be your best friend. I used to not like seeing feedback on what I needed to do better. Now I do not like when people write short feedback about me like, "He is great. He is a good engineer. He drove the project well." Even though it makes my ego feel good when I read it, I soon start wondering what I could have improved on. Is there nothing for me to be better at? Did I not make any mistakes? Or did they secretly complain to my manager and ask to only share the good parts with me?

There is no better friend than receiving constructive feedback. Why do I say constructive? Because as non-actionable as the feedback "He is great" is, it is also disheartening to hear "He is terrible" or "I do not like working with him" or "He was hard to work with." These are also just complaints and not really feedback. The best kind of feedback you can give and receive is the constructive kind where there is an action plan inside. It can be as short as "He complains a lot." Even though this is not detailed feedback and not super actionable, it is still telling you that you should complain less. It is still valuable. But think about this feedback for a second: "I liked working with him. He was easygoing, fast to ramp up, and started helping the project in no time. He was willing to help with whatever was needed. I noticed he was also working on two other projects with different teams. He tried his best to keep us informed, but we did not have too much clarity on what his assignments were on the other projects and how much free time he would have for us going forward. If he puts together all the things he is planning to do for the month and shares it around, all the teams would have a better understanding of what to expect." This type of feedback makes you feel good, tells you what you did well and shows what you could have done better. When everything is clearly outlined, it is easy for you to improve yourself. This is the kind of feedback you should be aiming to give and receive.

One thing you can even do better is to seek this feedback all the time instead of waiting for the end of the performance cycle. If you receive feedback at the end, even though it gives you a chance to improve yourself going forward, you are missing the chance to do it early. Also, it might affect your performance rating if there are more than a few things you have not done great. In order to make sure you are improving yourself non-stop, and not waiting till the end, it is always a great idea to start asking for feedback early. It can be as informal as talking to your peers and explaining how much you appreciate receiving feedback and asking them if they could write a few things you could have done better once a month. Some people might feel uncomfortable sharing their true feelings, and if you think there is a chance of hitting these barriers, there are feedback tools out there that help keep comments anonymized. Make a search and find one. Send feedback requests to multiple people, put them all in a group, and tell them you will only be able to see the feedback if there is more than X number of comments received to keep it anonymized.

By doing this, you can get more honest feedback from your peers and start improving from there. Now instead of hearing things you should have done better at the end of the performance cycle, you will start receiving feedback on how you have improved on A, B, C from your peers. Engineers appreciate someone who is self-aware and trying their best to improve themselves. And this will help you bring your A-game to work without punishing you for seeking feedback. It will make receiving feedback your best friend, and when it is time to change your team or company, you will be taking a much better self to the new one so you will have no roadblocks in front of you.

Suggestions: Become best friends with receiving feedback. Always think of what you can improve about yourself, share it with peers, and work on it. Instead of waiting for the end of the performance cycle, ask your peers to give you actionable feedback throughout the cycle. Find and use tools to anonymize them if needed to make your peers feel more comfortable opening up about their honest opinions.

44. Dogfooding

Have you ever heard this word before? Not sure how common it is used, but I really like it. In case you have not heard it before, it refers to eating your own dog food before sharing it with your best friend at home. Of course, it does not mean real dog food. It is the process of using or testing your own services.

How often do you use your own service? You would think if you are working on something, you would use it regularly, right? Not really. A lot of the time, engineers are working on a small part of a big project. They test and use the parts they have developed. Still, I have seen many instances where engineers often do not use the main product, do not know half of it, or are not familiar with who the users are.

Why is this so important? Because you cannot align yourself to the app or the service without knowing who the users are and how the general flow works. Let's assume you are working on a note-taking app. You are building a new feature that lets users draw shapes. You implemented a nice pen style, brushes, erasers, and everything. You keep testing it again and again, and every component you have built seems to work. But then you send your changes to production, and no users seem to be using it. You start thinking about why your project is such a failure. You open the app to see it in action only to realize your toolbar is missing. Someone enabled another toolbar that pushed your toolbar down, and it does not fit into the page anymore. Do you think this is not possible? You would be surprised how many times this happens. Of course, this is an easy example to catch what is wrong. Most of the time, it is not that easy. There are many different screen sizes, many different phone sizes, many different technologies like iPhone and Android, etc. If you have done all your testing in one device and then productionized it, likely it is failing for some of the devices.

Or a more common problem is you have tested the brush tool, the pencil tool, and the eraser, and they all work well. But when you draw more than twenty shapes, the app crashes. Or when you draw a shape with a pencil on top of the shapes drawn by the brush tool, the app crashes. Most of the time, we introduce bugs to the production when we test the feature we are building right now. As we have said many times, engineers are busy. No one has the time to test every scenario again and again and in different combinations and in different devices etc. It simply takes too much time.

How can you avoid introducing such tedious bugs to your service? By being a user of your own service. It is not always possible to use everything you build on a regular basis. But the more you use them, the fewer chances of something going wrong with them. The more you use them, the more you will learn different parts of the system, and you will have much more creative ideas that will complete the rest of the system and make users feel like the app is flowing instead of every part of the service acting like they were designed for a different one.

Can you do better than just dogfooding your service yourself? Yes! You can schedule dogfooding sessions and invite other coworkers to use them with you. You can just ask everyone to use it randomly for ten minutes and tell you everything that they did not like or everything that caused problems. It is even better if you can write a list of things to test or break it down into different components and assign each one to at least one coworker. This way, you will make sure everyone is testing some part. For example, one person should play with the pencil while another one is testing the brush tool. Do you want to make it even more fun for your coworkers and find more bugs in your service? Incorporate some pizza and some games or some prizes. Tell everyone there will be three choices of pizza for people joining the dogfooding session. Tell them that whoever finds the most bugs will get a $20 gift card for their favorite coffee shop. The best part is it does not need to be an expensive gift. Engineers like to be challenged and appreciate any kind of small gift to make them feel rewarded. The more challenges and games you incorporate, the more bugs will be found.

A lot of companies already offer incentives to make sure their engineers are using their products. Most of the teams I have worked for had some kind of gift cards to use on the service, or they implemented policies for reimbursements. Like if you use your note-taking app and purchase a nice plugin to test it, the company pays the cost. Or if you are working on a ride requesting service, you can enjoy free rides to test it out. Win-win! The incentives paid by the company compared to your salary is almost nothing. But they get their products heavily tested by the engineers who are building them. At the same time, you get to enjoy some free benefits to do what you already should have done in the first place. Test the service!

And if you are working at a company that does not offer incentives, just talk to your manager. I am sure they will be happy to grant some spending amount to use as an incentive. If you are planning a dogfooding session, ask if it is okay to use a corporate credit card to purchase the pizzas and the rewards. I do not think they will say no.

Dogfooding should be an essential part of your day. I would even go one step further to suggest working for projects that you are actually interested in using. You will be more engaged, you will be prouder, and you can enjoy the benefits of every feature you have ever built. And you will know what is missing for the rest of the users. By implementing features you would use, you will be helping all other users who wanted that feature but had no way to reach out to you.

Suggestions: Eat your own dogfood before anyone else. Keep using your service as much as possible, and plan dogfooding sessions. Make them organized, assign some parts to everyone involved, and make sure there is some food or prizes involved for the audience. It will make sure they find everything that needs some improvement.

45. The Importance of Knowing Terms—Office Jargon

Our industry is evolving so fast. Not a single day goes by without some new developments. You start following speeches, and most of them use certain terms; if you are following leaders, new terms start popping up here and there all the time. Like the words *leader, entrepreneur,* or *philanthropist,* which have become so popular in the past decade. Or the words that we have all gotten used to, like *sprint planning* and *stand-ups*, etc. There is also a trend to find acronyms for everything we use. Engineers love using abbreviations that mean different things like AFK–away from the keyboard, AFAIK–as far as I know, OOO–out of office, or LMAOROTF–laughing my ass off, rolling on the floor.

And this is only the tip of the iceberg. Every project name gets abbreviated, every algorithm we use gets abbreviated. Technologies, business metrics, and pretty much everything else eventually finds its way to some kind of abbreviation. Do you feel dizzy sometimes just trying to understand a sentence like, "AFAIK, he is OOO today, but I have checked DAU and ARPU and everything LGTM"? You are not alone. I used to think everyone but me got every acronym being used. It turns out they do not either. It is best effort most of the time. Sometimes someone uses an abbreviation, and I look it up and ask in the group, "Is this what they meant?" The group will split into a few groups, everyone with a different understanding.

There have been times that the dictionaries I have checked had over five or six different meanings, and all of them fit the sentence. Good luck figuring it out…

But no matter how confusing they might be, acronyms and terms are a daily part of our lives. We communicate by using them. It is important to keep up with the current terminology used day-to-day. How can you do that? There are a lot of online dictionaries that can help with looking up well-known acronyms. You just need to make a Google search. Most companies I have seen also had internal dictionaries where you can search for acronyms. The next time you hear an acronym, do not wait; just look it up on the spot so you can better understand the sentence. The more you learn, the easier it gets to communicate with people.

What is the alternative? If we do not use acronyms or names of algorithms, how can we communicate with people? More than a decade ago, I was interviewing. My English was not great, and I did not know the algorithm names in English well. My interviewer asked me a simple question: "How would you design an algorithm to find the shortest path to escape from a maze without being seen by the security guards?"

If you have ever been to software engineering interviews, then you know this question is simple. All you need to say is the magic words *breadth-first search algorithm*. Most interviewers will believe you can solve this question easily and move on. Suppose they do not believe and follow up and say, "What are the steps?" In that case, all you need to say is, "I would put the root node into a queue and then loop pulling from the queue one at a time and adding their neighbors if they have not been visited and checking if I have reached the destination." It takes a few seconds to form the sentence, and your interviewer already knows that you know the answer. Why? Because you have used important terms like *root node, neighbors, queue,* and *loop*. These are the terms used by software engineers. By using them in the right order, you have just described how the breadth-first search algorithm works.

What if you do not know these terms, but you know how the algorithm works? You will end up with a long misery of trying to code it to a whiteboard, test the steps, and make sure you remember how to code each piece. Which is what I ended up doing. This is still comparatively an easy roadblock to recover from. Maybe you will waste more time than a few seconds, but it is not going to be impossible to get around.

Now think about senior engineers. They design systems that consist of many moving pieces like load balancers, DoS protection, health checks, watchdogs, round-robin distribution, optimistic hashing, Kubernetes, master partitions, slave partitions, replica policy, round trip times, backend, frontend, VPN, third part service, encrypted request, SSH, SSL, HTTPS. The list goes on for hundreds of pages, if not thousands.

Suppose you do not know the term *load balancers*. How many minutes do you think you will be spending to explain "The service that takes a request from the user and checks the health of your service while keeping in mind how many current requests are going toward it, not to crash the machines, and only after making sure it is not some kind of hacker traffic, it sends it to the service. Oh, the thing that also provides caching." Even this explanation used so many terms like *user requests, crashing a machine, caching*, etc.

There is no way you can survive without knowing a lot of terms and what they represent. If you do not know the terms of your profession, it is like being a tourist in a foreign country where they do not speak your language. You will still be able to get around by pointing your fingers or making noises, but how much are you going to enjoy not being able to communicate with anybody? When I was traveling through Italy, my throat was hurting, and I just wanted a cup of tea. I thought, how different can it be in Italian. I sat down in a café and asked, "Can I get some tea?" The server responded, "Tea?" I said, "Yeah, tea please." He said, "No, no, no, no. No tea." But I had seen other people drinking tea, so I kept insisting, "Tea, tea, tea" (changing the pronunciation a bit each time) while making shapes with my hands like I was drinking something and pointing to tea on another table. He kept saying something that I did not understand. Finally, some nice person said "tè per favore" and told me the pronunciation of tea means *you* in Italian, spelled as "ti," in case you are curious. So the whole time I kept telling him, "you, you, you" while making shapes with my hands.

Unless you want to feel like I did in Italy, you need to learn the terms. Unfortunately, there is no golden book with every term you will ever need. And even if there were, it would become outdated in a year or two. Also, I do not believe you can learn and memorize all the terms in one sitting. You need to make this part of your life. Every time you hear an acronym, look it up. Every time you hear a keyword you have not heard, look it up. Go through essential algorithms online and learn all the names. Open cloud providers' web pages and click on the Products link and just look up all the words you see, like what is a load balancer, what is a database, what is a cache. Repeat, repeat, and repeat.

The more terms you start learning, the easier it will be to communicate with others. The more you learn, the fewer times you will be looking things up because even though there are thousands of terms to learn, it is quite easy to get to most of them in a short period of time. I am sure you already know most of them. Even if you do not know the term for it, you must know what they actually mean. Especially if you have graduated from college or worked for a year or two, you have already been exposed to all of these. Maybe you just did not pay attention to what they were called. Start paying attention, and before you realize it, you will pick most of them up.

Suggestions: Every time you hear an acronym or a term you do not know, look it up. Create a list of terms and keep noting them until you memorize them. Terms are our language to communicate with each other during design discussions. Never underestimate their value.

46. Knowing Popular Frameworks and How They Work

I wish someone had given me this advice when I was a junior engineer. As we talked about the importance of knowing terms, there is another thing that is just as important, especially if you are on your way to the senior track. That is to know the popular frameworks and how they work. You do not need to know all of them or all the details; that would take too much time. But to learn some basics takes almost no time, and it is super beneficial.

Let's assume you are trying to speed up your service's response time to users. What can you do? What if you have never heard the words *cache, Redis, Memcached,* or *in-memory store*? If you do not know the basics of what a cache is and how it works, it will not even appear in your thoughts to use it. Caches are complicated software that comes with many great functionalities. You can study how to design a cache yourself for a year and still not be great at it. They do so many magical things like automatically discovering your servers, scaling themselves, copying data around, replicating the data for data loss, partitioning the data so not every server has all the copies of the full data, bringing most recently used data to memory, offloading not often used data to disk—and the list goes on and on and on.

The good news is, you do not need to know any of these unless you decide to build a cache yourself. In which case, you can learn all these things then. All you need to know is the word *cache* and have a simple idea that they allow you to keep calculated information to prevent the same calculation from happening again and just respond to users fast. That's it. Learning all the details of caches might take months to years. But learning what a cache is and maybe learning a couple of famous names like Redis or Memcached and their pros and cons will just take you a few minutes.

What can you do better? If you have the time, write a small prototype. Most of the famous frameworks even provide a playground on their website. Like if you have heard about Golang, you can just search their website, and they have a playground right then and there. You do not need to install anything, and you do not need to do anything. You just read their hello world tutorial, type it on their website, click the "Run" button, and that's it. Of course, not every framework comes with a playground, and some require extensive installation investments, but you can leave those to later times.

Pretty often, I find myself opening all the cloud providers' websites and checking to see if they have added a new product. If they have, I start researching it to see what exactly it does and how it is designed. Or I find myself searching Google trends or GitHub trends to see which projects are trending. Even though I will not spend too much time reading the code of the projects themselves, I try to understand what the main purpose of the project is and the main working logic behind it. I take a note of it for later in case I need it. If I have the time, I just run a small prototype. If not, I just take a note and move on. The next time I am designing a project, I seek opportunities to use the new knowledge I gained to put them into action.

Start a new page in your favorite note-taking app and fill it with all the names of the technologies you wish to learn. After each one you learn, take some small notes on what it does and how it is designed. Next time you need it, it will be a quick lookup away. If you need some starting points, I suggest looking at Apache's website. They have so many open-source software of great quality. There are a lot of websites explaining how each of them works. You can learn the main design if you are interested; if you are interested in more, you can read their code and learn more; if you want to use them, you can use most of them for free since they are open-source. And next time you go into a design discussion, you can even throw around some of those names and tell people how they handle a certain situation. Like if you are working on an in-memory database and tell people Redis is mainly single-threaded, and it is world-famous and working like a charm, you gain huge leverage to push for single-threaded design and avoid all kinds of mutexes, semaphores, and learning low-level programming of concurrent system design.

Suggestions: Learn about popular frameworks and how they work on a basic level. With minimal time investment, you can prepare yourself for great design discussion and have an idea of what are the available solutions out there and how they solved similar problems you are facing.

47. Never Keep All Your Eggs in The Same Basket

How is your current project distribution? Are you mainly focused on one thing? Or are you working with multiple teams? Are you helping a few different projects or leading a big effort?

Before we start diving into what I mean by *Never keep all your eggs in the same basket*, let's mention what I do not mean by it. I am not suggesting you go out there and sign up for a bunch of random projects and randomize the heck out of yourself. Randomization is not a good thing. Everyone has a limit to how many different things they can drive at a time, and I suggest you do not push your limits too far to figure out where that boundary is. Because if you do that, when you realize you have too much on your plate, everyone will be upset about your performance, you will be burnt out, and you will not be doing anyone any good.

Since we got that out of the way, what are we trying to say here? It is simple: Do not bet all your money on one lottery ticket. If you have ever received any financial lessons, they would tell you the same. Do not invest in one stock, do not invest in one currency, do not invest in one house. Diversification is the key to surviving through the toughest crises around the world. If financially, all everyone is constantly talking about is diversification, why should we not apply the same principle to our projects? Easy answer: We should!

Again, I am not suggesting attacking everywhere and signing up for everything at the same time. You should do one thing at a time, for a while. You should do your research and pick the project you believe in the most. You should focus on it until you are ramped up and put together a good plan to bring it to success. Then start onboarding other engineers and focus more on leading them and offloading your time to set the direction. When you feel the project is not too challenging anymore or not taking all of your time, it is a great sign that you should start looking for another project and do the same. You need to find another project while driving the first one and start helping the second one. Repeat all the steps above. Focus on it until you are ramped up, put together a good plan, and make sure everything is ready to go. Start implementing some features, then onboard more engineers and start leading them and taking yourself off the project slowly. Now you are working on two important projects and achieving important goals toward both. If you start feeling you are not being challenged enough, start helping the third project and fourth and fifth, and the list goes on. The sky is the limit.

One thing to note here is, you will hit a limit at some point. If that is happening, that means you have not offloaded yourself well enough from the project. Because once everything is set up correctly, the projects should be driving themselves, and they should only be reaching out to you when something unexpected happens. There are going to be unlucky weeks when a few of the projects you have helped have some crisis. Still, hopefully, if you have put together the important stones nicely, it should not happen that often. And other than those crisis moments, people should not feel the need to come to you.

If you can achieve this, not only will you keep collecting the impact from all these projects' success, you will also keep yourself challenged and learning new things while making yourself harder and harder to replace. Why do you need to do this? Because the direction of a company changes all the time. The direction of the divisions and the direction of the teams changes all the time. What if you were working on some kind of client-side load balancer, and suddenly there was an open-source version of it that came out with great features? Do you waste your time implementing things they already implemented just to keep your job safe? Or do you switch it over and prove to everyone you do not have much value for the company anymore since your contributions were replaced by an open-source version?

Tough spot to be in, right? Well, it happens more often than you think. I cannot count the times I have seen people realizing another team is building something similar, or an open-source version is built, or the company decided not to pursue the project anymore. This happens because of budget cuts, direction shifts, or simply profit expectations from the project not being satisfying anymore.

Regardless of what might happen, you should always try to expand your horizon and your influence. This is the number one way to increase your seniority level. The more influence you have, the more senior you are. The more people you can help, the more projects you can help succeed, the more your value to the company keeps increasing. This directly relates to the risk of losing your job decreasing and the chance of you getting promoted increasing. Learn to leverage others and invest in making them successful so that you become a great leader while they become great engineers.

Suggestions: Solve the hard challenges, put things in order, onboard more engineers, and focus on becoming a great leader. Once you feel like you are not challenged enough, pick a new project, and do the same, and again and again. Never keep all your eggs in the same basket.

48. The Importance of Positive Surroundings

This topic directly correlates to your level of happiness. How positive are the people surrounding you? How supportive are they? How comfortable do you feel making a mistake? Do they complain all the time, or are they great at seeing the positive side of things?

The importance of your surroundings has taken me quite a while to learn. I used to not like picking my friends. It felt like it was a terrible thing to look at someone's qualities and decide if they deserved to be my friend or not. I was friendly with everyone, from random people I saw on the street to anyone who talked to me at work. From all my friends to my family members, I never separated them and never treated anyone differently. I have never pushed someone away because they did not fit into my requirement list. I would hold them close and treat everyone like we were close friends. I would listen to everyone who had something to say. If any person on earth decided to talk to me, I would spend the same amount of time I would give anyone else.

One thing to note here before we talk more on this topic is that there is nothing wrong with being nice to everyone. Treating everyone equally and with respect is a great value that I still hold to this day. You should never discriminate against people based on their life situation, how much money they make, where they work, etc. And I strongly suggest you remember we are all equal beings and treat everyone with respect. The more you learn to respect everyone, the more you learn to respect yourself. I want to make sure you do not walk away with the wrong understanding from this topic.

One day, one of my friends told me, "You are the average of the five people who are closest to you." I started thinking about it, and it was scary to realize he was completely right. I have always been social and loved making friends. Throughout my life, I have had friends from all kinds of religions, all kinds of political beliefs, all kinds of craziness levels. Even to this day, I have at least one friend from pretty much any background and any belief. But back when my friend made that comment, the closest friends I had were pessimistic and obsessed with work politics and getting promoted.

Even though I still believe being friends with everyone is a good thing, now I believe you should be careful about who you accept into your inner circle. You should be careful about who you spend more time with than the rest. Why? Because the people you spend time with will change you. The more time you spend with someone, the more alike you become. Their beliefs will start rubbing off on you, their life view will start changing yours, their expectations will slowly become yours, and you will start doing the same things.

It is only natural to be like the people we love. If you love someone, anything you are exposed to regularly becomes a normal thing to you. That is how our brain works. We have a special kind of neuron in our brains called ***Mirror Neurons.*** Guess what their role in our brain is? They are designed to copy other people's neurons you are exposed to. Why do we do that? Because they help us understand each other. It is how we have empathy. Think about your best friend now. You can finish their sentence right after the first word they say, right? That is your mirror neurons doing the work for you. You do not think like them; you actually become them every day. The more time you spend with someone, the more your brain mirrors them.

This is one of the reasons we like hanging out with people who are like us. Most of us get uncomfortable when we are surrounded by people who are very different. It is harder for your brain to copy their neurons, it takes longer to do so, and until you do it, your brain cannot predict their actions, and this causes stress.

I hear you: Enough with the biology lesson. Going back to our topic, if our brains are designed to mirror the people surrounding us, then it is easy to see how important it is to surround yourself with great people. And if you do not believe me, give it a try. Find five pessimistic friends who complain about everything, spend a month with them, and see for yourself. You will start hating the sun for how hot it is, wind for how cold it is, food for not tasting good, work for being boring. I have been there.

Going back to my pessimistic friends, around the time my friend commented about being the average of the closest friends you have, I indeed had close friends who were pessimistic. They were engineers, making six figures, having a life most people would call a dream, but they were unhappy. They were missing things in their life they cared deeply about, and it was affecting them heavily. Which was understandable, and I always thought they had every reason to. Just because you make more than others does not mean you cannot get unhappy. I tried my best to be there for them, reminded them how life has so much to offer, and tried to change their perspective.

What do you think happened next when I was hanging out with them all the time? I started becoming more problematic. My life goal turned into getting promoted; everything became less joyful, everything became about achievements, and my life became unbearable. I became one of them. I was making six figures, living in a nice town very close to the beach, and had many hobbies and so many places to go to. Yet I was unhappy. I was fixated on my level, putting all my efforts into getting promoted—because that was what all my closest friends cared about. We talked about work politics, who got promoted, the unfairness of the system, etc. This lasted for a while.

One day, I remembered who I used to be. I remembered how a tiny bit of wind would make me smile. I loved the morning breeze while driving my car. I would open the window and feel it on my face. I liked putting one of my hands out the window and feeling the air. Somehow, this was enough to make me happy. I used to be so happy when the first sunlight came through the blinds. I would work out all the time and never cared about how much money I made or what level I was at in my job. I was a strong believer that you only live once. I would go on adventures and road trips, pick up new hobbies, and enjoy every second given to me in this life. From such a positive person, I became one of the most negative people you could possibly meet.

I realized it was time to update my values. Letting just anyone become close friends with me was not a wise choice. I still believe in giving people a chance or maybe two. But you need to measure what they are adding to your life. I still become friends with anyone who cares to talk to me. But I limit my exposure to them unless they meet my criteria.

It is an amazing skill to be able to get along with everyone. No one will be able to damage you in a short exposure time. You will get diversified views of the world and learn so much from the different perspectives. But if you are going to see someone more than once every couple of weeks, you need to be careful. There is nothing wrong with not wanting to hang out with someone too often. No one deserves the opportunity to make you miserable.

I changed my values from treating everyone the same to treating everyone the same but paying attention to their personality. I still do not separate people depending on how much money they make, where they live, what kind of a job they do, or what kind of car they drive. These are materialistic things that simply do not matter to me. I would not judge anyone if they want to network with the rich. It is just not my style. But what I learned to care about is their personality. Life might affect everything someone has. Life might go easy on them and give them a great job or be tough on them and take everything away one day. I have seen a few wealthy friends end up with nothing one day. That is not something we can control. But our personality is completely our choice. And it is not a single choice. It is the accumulation of every choice we make every day. I have decided personality is something fair to judge people for.

If you have a positive personality, we will get along great. If you are adventurous, we might become best friends. If you have a wonderful personality but are going through a hard time, I will be there for you every step of the way. But if you choose misery over joy, if you decide to get fixated on things that I do not find valuable, then I will not allow you into my inner circle of friends. Again, I see no harm in being friends with those people too. You can talk to them, try to help them to cheer up, and definitely treat them with respect. But do not let them become a close friend; if you hang out with them too often, they will start making your life miserable.

The best kind of friend you can surround yourself with is the kind you enjoy talking to, the ones you walk away from with a smile, who you do not feel ashamed to tell your biggest failures to, and you feel their support regardless of what you choose to do in your life. Just make a small test. Think of something you have never thought of before. Like would you like to write a book? Would you like to give conferences? Would you like to build a start-up? Something that requires some effort and some risks. Call your friends and pitch the idea to them. See how they respond. Do they share the real risks with you but tell you that you should go for it? Or do they start laughing at you, telling you will fail for sure? Do they make you regret opening your mouth in the first place? Or do they believe in you so much that they want to take part in it?

Remember, you only get one shot at this life. You will not be given a second chance to fix your mistakes. You will not be given extra time to make up for the days you were miserable. If you were miserable in your life, when you die, it is game over. If you were happy in your life, when you die, it is game over. Make sure that from now until the end, every second you have counts. And make sure you are surrounded by people who make you feel happy to be alive for each of those seconds.

Suggestions: We are the average of the closest friends we have. Make sure you pick your close friends wisely. Make sure they support you, they are there for you, and you feel comfortable sharing any failures in your life with them.

49. Being the Happy Coworker

Engineering is a stressful job. We have repeated this in many topics. We also talked about the importance of your surroundings in the previous topic. Now it is time to turn the tables. While it is important to surround yourself with positive, supportive people, you must be the same way. You also need to focus on building healthy habits to keep yourself positive and happy.

I have worked with all kinds of engineers. Some are always stressed, some always enjoy what they are doing, some are crystal clear at work-life separation, and some work non-stop. But between all these people, there have been a few I could never forget over the years. What did these people have in common? The joyful look on their faces at work.

When you go to them with questions, they always welcome it. When they had to design something, instead of being stressed, they took their time talking to different teams and learning from them. They drove design meetings and let everyone be involved. Instead of pursuing their own designs, they put together all the suggestions they received from other teams with their remarks on them and let the team design things as a team. They were not strongly opinionated and were not always trying to prove they were better. They listened to everyone with open ears and tried to understand them. I cannot remember a time they were pushing people to design something quickly. I have never felt their ego when I was working with them.

This is so hard to achieve with all the deadlines lined up. But when you can build the kind of environment where you enjoy your day, your job stops being a source of income and becomes an enjoyable part of your life. But what about the deadlines and how people perceived their joyful actions? I cannot remember anyone ever complaining about them. I cannot remember anyone thinking they were not fast enough. Sometimes all those deadlines are in our heads. Your manager asks you to give them a deadline, and you give a tightly squeezed timeframe hoping to impress them. Then they say, "Can we do it in this much shorter time?" You keep thinking, and you decide if you push yourself a bit, you can do it. You say yes, and you resign yourself to unhappiness until the deadline. Then you do it again, again, and again.

This was a great reality check for me. I picked my deadlines to be competitive with what I could do best. I tried to get better and faster at building systems to achieve more. I thought the system required that. It is true that we are expected to perform greatly and produce as much as we can, but our industry is a little bit twisted. What do I mean by that? Let's give some examples. Suppose you were to make a small app to try on some clothes over the weekend, but ten million people decided to use it every month. In that case, you would be looking at six-figure incomes in no time. On the other hand, if you worked eighteen hours a day for five months, but no one is using your app, you are looking at zero income…

Fortunately, and unfortunately at the same time, our industry is not like other ones. How much you work does not necessarily get translated into what is your impact. There have been a few times I found some low-hanging fruit during my career. One of those times, the change was just one line. One line change in a configuration file. It took around five minutes to write it, a bit more to deploy and test. The outcome—a seven-figure cost savings every year. Yes! Not once or twice but every year. So even if I were to be paid for the rest of my life without doing anything else, I was still bringing more money to the company than I would receive in compensation. There have been times where I would put in extra hours for months, and my project would not bring in remotely what I achieved with that one-line change.

I am not suggesting you go on a wild goose chase to find low-hanging fruit all the time and spend your days relaxing at your beach house. But sometimes, you need to do a reality check. Do you need to be stressed and working all the time, or is that what you think you need to do? Changing jobs in our industry is quite simple these days. If you are not scared of going through a round of interviews, maybe you should test your reality. You should lower your pace and start focusing on what you find joy in doing.

Do you know what the tricky part is in this equation? If you actually start working on projects you deeply care about, and if you are happy most of the day, without realizing it, you start becoming more productive. The code you write becomes better, your impact becomes higher, and your teammates like working with you, so the feedback you receive becomes better.

We are not designed to be emotionless coding machines. Find something that makes you happy and work on things that increase your drive. If your coworkers remember you the way I remember those people in a few years, you will always be able to find a job. If any of those people were to reach out to me today looking for a job, I would do anything in my power to make sure they ended up on my team.

Try your best to be happy and make your teammates happy. Become the always happy coworker everyone is searching for.

Suggestions: Find ways to love what you are working on or find projects you will love working on. If you can achieve being happy during the day, your teammates will love working with you.

50. Working at a Start-up

I would like to share some insights about working for a start-up, especially coming from a big corporate perspective. When I decided to join Snapchat, there were a lot of things I feared.

- What if I was too slow? I kept hearing start-ups go at insane paces and always wondered if I would be a good fit.
- Shared office space was something new to me. Coming from Microsoft, I had enjoyed having my own office. I was scared of not being able to focus on coding with so much going on around me.
- I feared layoffs. Start-ups being tight on cash, you never know if they will let people go unannounced to avoid going bankrupt.
- Tech stacks. Big corporations have many internal technologies to solve most problems. Working at a start-up meant either implementing everything yourself or finding an open-source alternative. It was not something I was used to.
- Long work hours. Start-ups might require working long hours since there have not been well-established teams running things for decades.
- Income. Start-ups usually offer stock compensation, but because they are not public companies, it might make you rich, or they might go bankrupt before you can sell even one share.

How was my experience working at Snapchat? Unbelievably amazing. Going fast was something I had always wanted to try. I like to design as well as I can, prototype it, and then fix the remaining things later. Working at a start-up gave me the chance to design some of the biggest systems I have ever worked on.

The shared office was so much fun. This is based on how lucky you are. I ended up in great teams, and it was fun to collaborate, chat, and laugh all day long. It turns out I am good at context switching. Especially if it is to listen to a fun story. I have no problem laughing at a joke someone else is telling and going back to coding.

Even though the tech stack was scary in the beginning, I have learned so much. Best part? Building new things from scratch or using open source teaches you skills you can take away with you. When you join another company or start your own start-up, all the open-source libraries you know how to use are still available to use whenever you like.

People were quite respectful of work-life balance, so I never suffered through long work hours personally.

There have been scary days when the stock was not doing great, but overall, it was a great ride.

My favorite part was our offices were at Venice Beach in California. Can you imagine walking out the door to the beach? Having an ocean view while coding? Especially Venice being the center of the art community, every day was a different journey. Every day was fun. We would use beach walks for having meetings. Can you imagine chatting with your manager while taking a walk in the ocean breeze? I still wish I could find a time machine and relive those days sometimes.

Not every start-up experience will turn out as good as mine. And depending on what stage the start-up is in, the risks might be a lot higher. I cannot speak for all the start-ups out there, but one thing I can say for sure is you will learn so much. If you are wondering about working at a start-up, I would highly suggest giving it a try. If you are a good engineer who is liked by your peers, I would not imagine that going back will be too hard if start-ups turn out not to be your thing.

If you do decide to go the start-up route, make sure you meet some engineers beforehand. If you can have a quick conversation with them, you can learn about all of the things above and get some insights.

Suggestions: Working at a start-up teaches you so much. If you are thinking about working for a start-up, I suggest you give it a try. If you decide to do so, make sure to connect with engineers to get some insider information on what they like and dislike.

51. Office Politics

This one is one of my least favorite topics to talk about. I believe an engineering culture should not pay attention to who you know or what political connections you have. But at the end of the day, we are all human beings, and having the support of leadership does make a difference.

The long-term vision is driven by the leadership. Project distributions are influenced by the leadership. Hiring, firing, and most of the important decisions are made by the leadership. Being close to leadership will provide you with some political power. I suggest you do not try to use this to your advantage to build mediocre services. If you design good services, most engineers will respect you, and it should be easy to move forward with your projects. But needless to say, politics exists everywhere in the world, including inside companies.

There were times I noticed people would push back on a design until they heard someone with strong connections was in favor of the project. People simply did not want to argue and look bad to those connected to higher-ups. They have the advantage of being able to give feedback about you directly to the higher management.

There was this one time when an engineer was pushing back hard. His stance was, "This is too complicated and not usable. We need to redesign it." But as soon as he heard the idea had come from someone who was known to be well-connected, he immediately changed his words to "On the second look, I think I overreacted. I see what you are trying to achieve here. It kind of makes sense and lets us be ready for the future. I knew there was something I was not seeing. You are trying to cover future scenarios; that is why it is a bit complex. Nice!"

It was heartbreaking to see this happening in person. But unfortunately, it did. Sometimes people do not listen with an open mind to see why you designed a system in a certain way unless they fear your connections. As I said before, this is not a good strategy. And it only works when you are strongly connected. Management tends to change regularly; positions get realigned all the time. If you use your political standing to get things approved, people will take note of it. If you then lose your influence or if you join a different company, you will start hitting the wall every time you realize people are against your opinion. It is a lot wiser to use other strategies and build a strong arsenal of tactics to convince people or find ways to come to a consensus without using any politics.

But still, it needs to be said: The more connected you are, the more powerful you are. Even if you are not planning to use political influence over people, it is still a good idea to build connections with the leadership. It will help you increase both your visibility and your sway over long-term projects. You will also get the side benefit of using those connections if things are going against you and you strongly believe it should be done your way.

Suggestions: Build connections with the leadership but use it more for influencing long-term vision and not to bully people. Always prefer other conflict resolution tactics, but if you have no other choice, having strong connections is always better than not having them.

52. How to Help Your Manager Work for You

Managers are a great resource, especially if you can figure out how to use them correctly. We have talked about the importance of having one-on-ones with them. We also talked about dealing with a brand-new manager. Now let's talk a little bit more about different ways they can help you perform better.

The first thing you should work on with your manager is your expectations document. There is nothing better than having clear expectations on what you are trying to achieve for the performance cycle. Your manager is the one who will be going into the performance evaluations and supporting your package of achievements. They have great insights into what is being talked about in those meetings and what they value the most. We have mentioned how our industry is a bit twisted, and working more does not necessarily get translated into more impact. Your manager will have a great idea of what higher management will view as an impact, and they can guide you in the right direction. As most managers are busy, they will likely not want to spend too much time on this. But it is critical for your career success. You should be consistent in asking them to give you an expectation document with a clear outline.

Secondly, the things you need to improve on. Feedback is your best friend. Who is better qualified to give you direct feedback than your manager? That is their job at the end of the day. They need to make sure you are performing better. Be direct with them. Ask them what areas they would like to see you improve on. Do you need to be better at communication? Should you increase your visibility? Is your code quality not up to their standards? Do you need to work on higher impact projects? Is there any area they see where an early investment of your time can turn into big achievements? These are all the great questions your manager can answer.

Use your manager as a work-life balance enforcer. When people come to you with requests, it is not a good idea to push back or tell them you do not have time. Instead, create tasks for what they are asking you to do, add some details in the description, add your insights on how valuable you think it is, and ask your manager to review. If they do not believe it is a high priority, request them to communicate this to the other team. Your manager telling people you are busy with other higher priority tasks is always better than you sharing this. Have your manager do the hard conversations.

Tag your manager in your design reviews. Even if they do not have the time to fully review and comment on your documents, this increases your visibility. In addition, your manager can tag other people if they believe they should review your design document. People will be a lot more receptive to take the time to review your documents if it is coming from your manager.

Use your manager as a central people database. Managers are great at knowing who owns which area. You can figure this out by starting a chat thread and asking people to send you in the right direction, but it might require a few hops before you reach the right person. A quick question to your manager can save you a lot of time and unnecessary interactions with people.

Start writing your performance evaluation document early. If you wait until the end of the performance cycle and send your document along with all the other reports your manager has, they will likely not have too much time to review it. Start it as early as you can and make it an evolving document. Keep asking them to review it and give you feedback. Not only will they have a lot more context on your achievements during performance evaluations, but this will also give them a chance to guide you early on. There is nothing worse than investing your time into something only to find out that your manager believed it was unimportant.

Ask for more interviewing opportunities. Show them you are interested in helping with growing the team. This will not only increase your impact but also help you build a stronger relationship with your manager, so it will be easier to get the headcounts you need later.

Ask them for the promotion template. There might not be an official template to use, but if your manager has taken a few engineers through the promotion process before, they will have a good idea on what are the important parts that need mentioning. And they likely already turned these into templates to save themselves some time. Most of the time, those templates have important information on what the management cares about when promoting someone. By seeing the template, you can start filling it out early and align your projects to make sure they look good on the promotion document.

Ask your manager to find you mentors. They have a great network that can be at your service. My managers introduced me to many great mentors that I wouldn't have been able to find by myself. Especially since managers are great at reading people's strengths and weaknesses, they can align you with a mentor that will be beneficial to you.

Build a personal relationship with your manager. They are human beings, and likely they will favor the people they like on a personal level more than just the professional ones. Grab coffee with them or go to a happy hour. The more you get to know each other, the better your relationship will be. Or you will figure out this relationship will not work, and you can save the time of investing in it and start looking for a different team with a better-aligned manager.

Take your long-term vision to them for a review. Share what the other teams are working on and how you believe your project will set your team in a privileged position in the future and get their feedback. They can help you make sure both you and your team are hitting the goal by having a good direction.

Have your manager resolve conflicts for you. If you notice someone is doing something that they need to improve on, help them by sharing your observations with your manager. Tell them what you believe is missing and how they can perform better. This way, you will not be taking the risk of people getting offended when they hear feedback from you. Your manager can anonymize the feedback, make sure it is objective, and deliver it successfully. And they do have the authority to follow up and make sure the necessary improvements are being made.

Overall, your manager is not your boss. They are a great resource. Use them as much as you can. And if at any stage you notice they are not a resource for you, that is a good sign you should be looking for a manager that is a better match for your expectations. A great manager will help you excel in your career.

Suggestions:

1. Use your manager as a resource for any conflict, any prioritization, any design discussion, or any time you need more authority.
2. Get them involved as much as possible.
3. Especially ask them to write you an expectation document and to share a promotion template.
4. If you feel your manager is not meeting your expectations to help you, that is a good sign you need a different manager.

53. The Importance of Intelligence

When you look throughout history, you will notice most powerful nations were the ones who had the best intelligence. If you enjoy watching spy movies, you will see how capable CIA or MI6 agents are. You cannot do anything about things you do not see.

Companies are no different. Having an intelligence network will serve you in many different ways. Have you ever heard about discretionary bonuses? These are bonuses usually distributed at the director level to certain individuals to keep them happy. If someone in your team is highly valuable to the company, and they get an offer from a competitor, your director might grant them a discretionary bonus to keep them around. Why is it discretionary? Because companies cannot afford to spread this bonus to everyone; otherwise the next thing you know, everyone on the team is looking for a new job to force the management to hand out bonuses. You will not see any explanation of what these bonuses are in the company benefit wiki. You will only hear about them if someone you are close with is granted one. Sometimes it is not even at the company level. Teams get certain budgets to make sure things are being run smoothly. It might even be a team giving out these bonuses, and no other team has them. Without intelligence, you simply will not know.

Same with retention bonuses. It is common for engineering compensation to be given as a base salary plus stock compensation vested over three to four years. What happens after three to four years? Your income declines. Companies like their engineers to move around. Fresh blood always brings in a lot of value. But if you are valuable for the team, they might grant a retention bonus to you, hoping you will not start looking somewhere else to increase your income.

There is more to intelligence than just financial outcomes. For example, maybe a layoff is coming, and managers are already marking which employees should be let go. If you have a good network, you will see this coming way ahead of time. If you do not, you will be surprised to hear either you or some of your friends are being let go.

What about important projects? Have you noticed most of the important projects in the company end up in similar hands? That is your proof they have a great intelligence network. When some people hear about the upcoming requests from the leadership weeks before you do, they can align their team to make sure they have the capacity to take it. When finally everyone hears about the projects, they were already perfectly aligned to take it, and you are stuck with what is left over.

Timing of the promotions, how many people are going to get promoted, which teams are getting headcounts for promotion, which teams are being given headcounts to hire someone, which teams are going to reorged—the list goes on and on and on. Without intelligence, you will always be left in the dark and hearing things last.

If you believe me in the importance of intelligence, how can you build your intelligence network? First, you need to be good at networking. We have talked about the importance of networking. The better you are at it, the more friends you will have in a different team sharing the gossip they hear with you. Secondly, remember the connections giving you political influence? They are the same people who are in charge of these decisions. If you have strong connections in leadership, you will be the first one to hear most of these things. What else can you do to increase your reach? You also need to learn to give back. The more you know and share, the more you will hear things because people will keep coming to you to learn what is going on. So many things I have learned from my friends were simply because they knew I had a great network, and they were trying to get information from me. You will see them coming with "Hey, have you heard anything about blah blah...?" The more people think you know things, the more they will come to you. How do you know which ones are real and which ones are just misinformation? You count how many people come to you with the same gossip. If three people who do not have much in common came to you with similar information, it is pretty safe to assume it is happening.

How do you build these relationships in the first place? By being open and showing people you trust them. Trust is a two-way street. If you try to use people and not give anything back, they will quickly learn not to trust you. If you open up to people and share your salary or your performance rating or the bad feedback you received or anything that requires trust (saying this to make sure you do not go and show off to people with the promotion you got and hope they share their secrets with you), the more comfortable they will feel sharing their secrets with you. But be careful not to violate any laws or agreements you have signed with your company. I am not a lawyer, so I cannot suggest if it is okay to share your salary or not. Try to read those agreements to make sure you are complying with them. But feel free to share anything with people you trust as long as they are not against something you have signed or the law. That is one of the best ways to build trust with others.

There were a few times when even my managers would ask me about what was going on around the company, thinking I had access to a good amount of gossip networks. This is a powerful tool to have your manager need something from you. Information is power, and an intelligence network is one way to gain it.

Suggestions: Build your own intelligence network. Remember, trust is a two-way street. If you are hoping to gain someone's trust, you may need to go first. Figure out what private information you can share about yourself without violating any agreements or laws and show your friends you trust them with that information. Soon they will return the favor.

54. Comparative Analysis

We have mentioned comparative analysis a few times in previous topics. What do I mean by comparative analysis? It is the simple act of putting things in a pros and cons list and comparing them to each other to make it easier to make a choice. It can be as simple as a list, but if you can come up with a table highlighting the differences, it will be much easier for your audience to compare them in place instead of reading different lists and trying to remember. For example, if we were to follow our in-memory data store example, it would simply look something like this for comparing strong consistency to eventual consistency:

Note: This is not an exhaustive list but just a simple one to give you an idea.

Feature	**Strong Consistency**	**Eventual Consistency**
Consistency	Immediate	Eventual
Writing latency	Slower	Faster
Reading latency with stale data	Same	Same
Reading latency with requesting newest data	Faster	Slower
Scalability	The more replicas added, the slower writes will be	Increasing replicas will not be noticeable by writer

As you can see from the list above, all we did was to add pros and cons to an easily comparable table. But within seconds of you looking at this table, you can pick whichever works best for your requirements.

Suggestions: Use comparative analysis to visualize the pros and cons and make a choice to fit all requirements easily.

55. Power of A/B Testing

Do you find yourself having a hard time convincing people you have a great idea? Do people doubt if it will work or not? How do you prove the impact of your project? If you are a front-end developer, it is comparatively easier to see if your project is adding value as users will either use it or not. But even then, how do you measure how much it brings in terms of metrics? How do you pick the best design? How do you make sure it is performing at its best?

Let's talk a little bit about A/B testing. In case you have never run an A/B test, let's start with defining what it is. The concept is simple: you add the functionality or feature you want and put it behind a switch that controls if your feature is shown to the users or not. Or a switch that enables your code or disables it. Hopefully, you work at a corporation that already has the A/B framework set up, but if they do not, there are so many frameworks out there that make running these kinds of tests easy. You ship your code to the production, and by using the A/B testing framework, you assign each user to a group. You can test many features together, and there are too many details, but in the simplest form, you just divide the users into two groups. 50 percent of the users get assigned to Group A, and the other 50 percent get assigned to Group B. Generally, A is called the control group, and it is the old code without your changes (or new code with your changes disabled by the switch), and B is the treatment group that enables your change. You run this test for a while depending on how many users are exposed and how significantly metrics are moving. After running it long enough to get some insightful data, you compare metrics to see how each group performed. If people in Group B kept using the app 30 percent more than the people in Group A, voilà! You have just proven the feature you have implemented and can claim that it brings 30 percent more usage.

Of course, this was a simple example with only your change being tested, and you were able to move the metrics by 30 percent. If only this would happen in real life. Unfortunately, unless you are working with a small team or an isolated project that can be tested by itself, this never happens. 30 percent is a great number. For established products, the changes are usually not more than a few percent increase. And even a few percent are a huge improvement. If you work at a company that has 50 million users, a one percent increase means 500,000 new users. That is huge.

Now let's talk about real-world scenarios a bit to give you some suggestions on how to run the A/B test more successfully. Assuming you are working at a big corporation, there will be dozens of changes being tested on a good day. How can you make sure your change was the one that brought the dough to the table? Meet the science of data analytics. It is out of our context to go into too much detail, but just think about this.

What is an average user like? Of course, there will be many cohorts like teenagers, elders, males, females, etc. Each cohort will have certain common behaviors. No one person is the same as another, but most of us go to work at similar times, come home at similar times, and do not work on the weekend. This is because we are heavily influenced by society. And we only get to choose from the options presented to us. If you have three job offers and all give you the weekend off, the likelihood of you working the weekend is almost zero. How can we utilize this? By uniformly distributing the experiments. Let's start simply by thinking there are two changes being tested concurrently. Our group distribution will be something like the following:

A - 25%	B - 25%	C - 25%	D - 25%
Control for your change		Treatment for your change	
A - 25%	B - 25%	C - 25%	D - 25%
Control for other change	Treatment for other change		Control for other change

Group A – Do not see any change, control for both experiments
Group B – Do not see your change but do see other change
Group C – Sees both your change and the other change
Group D – Sees your change but do not see other change

A good A/B testing framework makes sure of the uniform distribution of the experiment over other experiments. Remember how we talked about the average user? Since most of the users are expected to perform similar actions, they will do similar things in Group A as well as B, which will give your control group the power to average out the effects of other changes. Same with Groups C and D; since half of the users will see the other treatment and half will not, it will average out their actions. So in total, you have Groups A and B vs. C and D with the other experiment's effects averaged out, which in return will be able to detect what your change brings to the table.

The most important thing to note here, though, is that this is only true if you have enough users to achieve a uniform distribution of average users into different groups. If you are building a traveling app and have only a thousand users, and one of them makes a purchase in Group D, it will look like group D is performing way better. But in reality, that is only because it was not uniformly distributed, and the only user who was ready to make a purchase ended up in Group D. There are a lot of great books out there talking about variance, user count, metric significance, and how much do you need to see statistically significant (In other terms, provable changes in metrics) change. If you are interested in learning more about this topic, I suggest doing a deep dive into A/B frameworks. Most of the time, looking into previous experiments and seeing how other engineers picked the user count and asking around should suffice.

You will also hear the word seasonality a lot when people are talking about metrics. Let's say you have tested your change, and everything looked great, and you launched your change to 100 percent of the users. A month later, you look at metrics, and the sales have tripled. Can you go ahead and claim that your change brought 200 percent more sales? Not so fast. Suppose you launched your change right before Thanksgiving, and people started shopping for Christmas gifts. If that is the case, you will be seeing the effects of seasonality. There are important dates that will affect things even if you do not change anything. What is a good way to fight against this? You can incorporate previous years' data into the metric charts. You can see if there was a jump in previous years, or if this is something specific to this year. This somehow works but still misses out on specific things happening this year.

For example, COVID. COVID affected our lives in many unimaginable ways. So if you compare any pre-COVID metrics from previous years to post-COVID metrics, you will see a lot of unexpected moves in your charts. You will not be able to figure out if the metric moves came from your change or from seasonality. What can you do to fight against this? You can set up a back test. It is the same thing as running an A/B test, but you choose a smaller percentage of users and run it going forward for a while. Let's say you never enable your feature for 5 percent of the users for the next few months. It will not detect small changes in metrics, but if there is a big change like sales doubling or tripling, you will see how it performs against your control group to see if there was a similar increase with people without your change or not.

Make sure you become good friends with A/B testing. It is one of the tools that will make it hard for anyone to argue with you. There is no fighting against data. Yes, people might suggest things like bias, seasonality, or random shifts in user actions, but if you have strong metrics to prove your contribution, they cannot say much to that. You can even use the theory behind A/B testing in your day-to-day life. I apply it to many situations. I always try something with some people and see how they react to different situations. It is a fun game, and you learn a lot about what people appreciate more vs. not.

Suggestions: Learn how A/B testing works and use it on most projects you do so you get data to prove exactly how you have helped the critical metrics. Data is hard to argue against!

56. Logs, Events, Metrics and Dashboards

If you have been working as an engineer for a few years, you might already be familiar with logs, events, and metrics. They are the essentials of any system, but they are overlooked most of the time. In case you are new to the engineering journey, let's talk about what they are.

Logs are informative statements written by engineers to be able to debug a system later. For example, simple logs are like log("Service started"); or log("Established network connection"); they write this information somewhere when the line they are written is executed. They can just output these to the console, they can be saved to the disk, or they can be aggregated and sent to a log collection server. As you can see in the examples above, they are just statements. You can also add the value of variables inside logs.

What do we use them for? To see what happened during execution. Let's say the service crashed, and you were wondering what exactly it was up to before crashing. If there are a good number of logs written, you will start seeing something like:

1. Service started
2. User request received with id xyz
3. Fetched information from the database for request xyz
4. Error: Could not parse the fetched information from the database;
5. …… Long lines of the call stack showing which file and line threw exception like main.cpp#12,54

As you can tell, there was a problem with the data received from the database. If you have uploaded symbols with the executable, it will even tell you which file and which line caused the crash, like "main.cpp line 12, character 54" in our example. By looking at these log statements, you can easily tell that either someone manually changed the data in the database to an unsupported format, or there is a newer version of the service that wrote some data that the current version could not read. By looking at the logs, you can see exactly what happened. Same thing for distributed services or apps used by users. Logs are generated and later sent to a server so you can see exactly what was going on in the user's journey.

Events are similar to logs, but they usually mark certain actions being taken. Like the user clicked a button or the app was opened, the app was closed. You can log events as statements too, or you can keep them as structural data so they are easier to process. What is cool about events is the ability to aggregate them. To be perfectly correct, you can also process log statements and extract information and aggregate those too, but that is too much work, and it costs more money to process a stream of strings. Going back to events, you can see how your system is performing with events by just looking at a dashboard. You can pack so much information into a small dashboard.

Why are these important? Because they are your eyes and ears for your service. Without logs or events, you have no idea what is going on with your service. Maybe your service crashed a week ago, and no one is able to use it. How will you know about this? When one of your friends tries to use it and realizes and calls you on your phone to let you know nothing is working?

And your friend might not even realize it. Because a lot of problems can happen on the part of your system. Maybe half of the machines were updated to the new version and are no longer working but keep crashing. Maybe servers in Europe had an electric outage, and you keep using your service as usual in the USA. Or maybe it is not even your service but the database service you purchased from a third party.

I cannot express how important it is to be familiar with logs, events, and metrics. I suggest always creating your own dashboard and copying all the important information from other dashboards to yours so you know exactly what is going on when you look at it.

Also, it is even more important to pick good events and metrics to be collected. When you are writing your code, you should always ask yourself which part might go wrong and how you should phrase it so whoever reads it can know immediately what went wrong. The more time you invest in making sure you are writing good log statements or picking important events to be collected, the less time it will take to fix things. Especially if you are on-call and got woken up at 3 a.m. because of an alert. If you spend all night trying to find the problem but you cannot because of bad logs or events, you will know what I am talking about. The same applies to your code waking someone up at 3 a.m. But I am sure you would never do that to a fellow engineer.

Suggestions: Always write self-explanatory log messages and add as many events as possible; create good dashboards to make sure your system is functioning correctly.

57. Debuggers and Crash Investigation

This is another topic I am sure you have heard of before. But I was surprised to see how many people I know do not use a debugger. I also was not using them until I learned how powerful they are. Until I graduated from college, my debugging skills were as good as printf() statements. Every few lines of code I wrote, I would add a printf(1), and then printf(2), … printf(20) so whenever my app crashed, I could look at the latest number on the console and know where things went wrong. Sometimes I would not put enough printf statements. After figuring out which code block it crashed, I had to add more printf statements to figure out which line caused the problem. It amazed me to see the interns still using printf statements. It makes me feel that I am still a young soul.

What are debuggers, and how do they make your life easier? Debuggers are tools that can attach to a running process and provide valuable access to information. These are some of the common things you can do with a debugger.

1. You can run your code line by line. Debuggers show your code, and you can click the step button, and it will only execute the next line of your code in your program.
2. If you want to test out a certain part of your code, you can click next to it, and it puts in something called *Breakpoint*. If you run your process with a debugger attached, it will run all the code automatically until it gets to the line you have marked as breakpoint and pause right there so you can see what is going on in that piece of code.
3. You can see the values of variables by just hovering over them. Yes, no more printf is needed! You can see everything as they are running, and you do not need to pick which variables to print. Everything is accessible.
4. Automatic crash detection. If your service throws an exception and is about to crash, debuggers catch those exceptions and pause execution

before the service is shut down. How is this useful? Remember what I told you about seeing what variables are, seeing which line is being executed, seeing what the call stack is like? Yes, you still have access to all that information. So in a few seconds, you can tell what is going wrong.

5. You can execute new code or remove some of the lines from execution. Let's say you figured out which line caused the crash. Do you go back to your code, try fixing it, deploy again, and run the service again to see if it worked? Of course not! That is way too much work. Most debuggers support the ability to execute a code you have just typed in. You can skip the line causing issues and run whatever code you want instead.
6. And yes, you can also change the values of variables. It must be obvious since you can run any code during execution, but sometimes it does not ring every bell. You can simply click on variables, and when you see their values, you can just change them as well.

But what about services that are deployed to production? Most of the time, services do not crash while testing them locally on your computer. I have good news for you. There are tunneling extensions that let you do everything I told you about on a remote machine. If you have access to the machine to connect to the ports on it, you can create a tunnel and have the exact same experience as if you are attaching a debugger to a local service. The only difference might be a little bit of a delay between you clicking buttons and the following line getting executed. The information needs to travel to the machine and back to you at the end of the day.

If you do not use debuggers regularly, give it a shot. Take a crash course or search it online, and you will be amazed by all the capabilities they add. Debuggers are going to be your best friend.

Suggestions: Use debuggers, get better at them, and make them your best friend.

58. Remember It Is Not Your Company

This is one of the most important things to remember during your day-to-day job. There have been times I felt such belonging to the companies I worked for. I would fight with people to make sure the best systems were being developed or that we did not waste any resources to save money. The problem with this? It is not your company, and other engineers in the company are excited to learn things by making mistakes. If you start trying to stop them from making them, you will quickly become the unwanted leader around.

Even though it is great to feel like a family and there is some kind of belonging to the company you work for, too much is not good for you or for the people around you, and the worst part is it is not good for the company. Companies are built around longevity. Companies make most of the money because of the creative employees they have, not because everyone is always making the right choices.

If you were to start your own start-up, you hold high risks but also high opportunities. You might go bankrupt, but also you might become rich. When you are working for someone else, you are trading the risks for the gains. You will never be as rich as you could be in your own company, but the good news is you never need to take a risk that can make you go broke. If the company goes bankrupt today, you can go to a different employer tomorrow. You should always remember you already made this choice when you walked into their doors and stop acting like it is your company.

It is always wise to not waste any resources the company is providing, but there is no point in being stingy. I met people who would go out of their way to buy the cheapest plane tickets to help the company, stay far away and not up to standard hotels, avoid visiting their team in different cities, not take sick or parental leave, and so much more. I have met with all kinds of people who feel like it is their company and try to save pennies wherever they can.

But they are missing an important point. When you save money by not visiting your team, you are missing the chance to design things with them or bond with them in person. There is a reason the company is paying for those trips. Same with flights and hotels. Companies design their policies around what they believe will bring the most impact. Companies never try to baby you or give you a first-class ticket if they do not expect anything in return. It is all about the expected outcome from you compared to the investment. If your company has the policy to pay for airfare, or get you a nice hotel, or send you to a different city to meet team members, or they are paying for a conference, just use the benefits. If they are paying for a conference and you do not go to save some pennies for the company, you are missing the opportunity to learn something great and bring in the big bucks with what you learned. I will repeat it again: Companies never do something for you if they do not have any return from it. And your well-being, your motivation, your happiness directly impacts how much you like working there. It directly impacts how productive you are. Stop worrying about every little mistake people make or spending money here and there and keep in mind that it is not your company. Let the company worry about the policies. If something starts becoming expensive, they will adjust the policy in no time.

Your focus should always be on improving yourself as much as you can, enjoying every day as much as you can, and you can pay the company back by doing the best work you can in return!

Suggestions: Always remember it is not your company. Use all the benefits, enjoy your time there, focus on learning the most and being more productive, and let the company worry about the benefits they provide. And do not ever break anyone's heart trying to do the right thing for the company. Each person the company hires is as valuable as you are. Even though you might be thinking you are doing what is best for the company, you might be surprised to find out you are not.

59. Playing for the Long Game

We have made it to the final topic for our book. I thought about writing about this somewhere earlier in case you got busy with something else and forgot to pick up the book again. Thank you for investing the time to get this far. Playing the long game is the most important advice I would give to anyone who asks me. Let's get to it.

Always remember, you should be playing the long game. It took me years to learn this and more years to apply. I hope I can save you a few years by sharing it. We are engineers; we are the designers of the modern world; we are behind every second of everyone's lives. People wake up to our products, drive with our products, smile with our products, cry with our products, relax with our products, and even sleep with our products.

There is so much to achieve and so much more to do. I have missed a few opportunities because I was not playing the long game. I was focusing on short-term gains like promotions or performance evaluations. But you need to realize this is our life. We spend more time at work than we spend with our partners, with our kids, with our friends. You need to do everything in your power to prevent yourself from getting stressed, having an unjoyful job, or focusing on short-term changes in things like stocks, news, etc.

I have seen the stocks of the companies I worked for soaring as well as taking a dive. There were days I was convinced we might go bankrupt; there were days I thought we had so much money. There were days I spent fearing layoffs; there were days I was certain my job was secure. Life throws all kinds of things at you non-stop. And the most successful people are the ones who can keep focusing on the long game. The best opportunities always present themselves in the darkest times.

If your company is doing great, the stock is soaring, and money is floating everywhere, there are going to be thousands of engineers trying to get hired, work more, achieve more to get a piece. But you will get the best opportunities of your life when things seem to be terrible. We have talked about how Snapchat's stock soared more than seventeen times after the darkest days in the "Adaptability" topic. That was the biggest lesson of my life. I had coworkers who simply enjoyed their work, and all they said was, "I am in it for the long game." And they have been the ones getting promoted, getting more bonuses, and enjoying the stock when it went up more than seventeen times.

How did they achieve that? By remembering to play the long game. The only thing you carry to tomorrow is yourself. You are your most valuable asset. You need to stop focusing on short-term gains and focus on the long game. Make it a habit to have fun at work, keep improving yourself, build great relationships, build great products, and keep repeating this, no matter what happens. You will always find yourself in a better place than you were before.

If your company is not doing great, but you can keep focusing on the long game, one of two things will happen. First, your company will get better, and you will be the one getting promoted, rewarded bonuses, and enjoying them in better market conditions. Second, your fears are going to come true, and your company will not be there. But if you have improved yourself every chance you got, met great people, and have become a great engineer, you will be able to pick the company you want to work for as well as name your own salary expectations.

The worst thing you can do is burn yourself out. Never forget you are your most valuable asset. If you burn yourself out, you will not enjoy what you are doing; you will not be motivated; you will not be learning much; you will get tired faster, and soon you will find yourself fighting with your manager and your coworkers. The worse your morale becomes, the worse everything in the world will feel. Not only will you start suffering at your job; soon, you will start suffering in your life. You will feel like you cannot catch a break and life is just messing with you all the time.

And believe me, it is so easy to fix everything once you remember to smile. Once you start having joy, it will all come back to you. You will find things more interesting; you will start learning a lot more again, you will perform better at your job, and you will surround yourself with positive people and start excelling. It is all in our brains. We pick what we want to focus on. And what we pick becomes our reality.

So please be careful to make the right choices. Keep remembering this every day: You are in it for the long game. It does not matter when you get promoted; it does not matter if you get an extra bonus or not in this performance cycle. If you can keep your head on straight and love what you do, you will always get more of everything.

Suggestions: Always remember to focus on the long game. You are in this game for life. Focus on having fun, improving yourself, building meaningful relationships, and making every second of your life count!

60. Conclusion

I would like to take this moment to thank you for purchasing this book and improving yourself. Hope you have found the content useful. We have talked about the power of giving credit. Giving credit is a great way to share love.

Please leave us a review if you found the content in this book useful. I personally read every review and there is no better feeling than seeing I was able to help someone.

If there are parts you aren't thrilled with, please give me the opportunity to improve by contacting me directly with one of the ways shared below:

- Use the Contact form at engineerssurvivalguide.com

- Email me at book@engineerssurvivalguide.com

- Connect me at linkedin.com/in/merihtaze/

- Use Discord, LinkedIn or Facebook pages shared above

Interested In More Resources?

Follow us for more resources and staying up to date:

Blog / Contact Us Form:
EngineersSurvivalGuide.com

LinkedIn Page (Engineers' Survival Guide):
bit.ly/survivalLinkedin
(case sensitive)

Facebook Page (Engineers' Survival Guide):
bit.ly/survivalFb
(case sensitive)

Discord (Engineers' Survival Guide):
bit.ly/survivalDisc
(case sensitive)

Email List:
bit.ly/survivalEmail
(case sensitive)