**Interviewer:**

Just ask general questions first, I guess, and for this part of the interview, and then I'll go a bit more specific, I would say like things related to software engineering. Okay. So thank you very much, Steve and John for having me. Yeah. Today. And I'll start by asking you sort of mentioned this before, but what is the core service that you provide briefly?

**Interviewee 1:**

So X from Startup 1 is a CTO and development team for hire the way that that differs from a normal software consultancy. I usually say that with a normal software consultancy, oftentimes the absolute best case scenario you can hope for is that they build exactly what you ask for. Yeah. And a lot of times founders, especially of early-stage startups, where all of their experiences in the industry that they're building their startup for their experience is not in software startups. And so founders usually don't realize how often they're going to be asking for the wrong thing to be built.

**Interviewer:**

You mentioned this as well. You were founded 11 years ago. I would say right in 2008,

**Interviewee 1:**

Yeah, 2007, 2008. It was somewhere in there.

**Interviewer:**

Yeah. And you were located here I guess?

**Interviewee 1:**

Actually yes, we're located in Ann Arbor when I originally founded it was when I was moving to Ann Arbor from Flint. Okay. So we moved in, I moved Ann Arbor in 2008.

**Interviewer:**

That's good. Yeah. How many employees do you have at the moment?

**Interviewee 2:**

Eight.

**Interviewee 2:**

We have some consultants, you know, for main part, we have three developers, a creative director, office manager, director of operations. And then we have some consultants here and there.

**Interviewer:**

How's the gender balance? Like women, men?

**Interviewee 1:**

We only have one woman.

**Interviewee 2:**

It's very hard to find female developers. Although at, at spark, at this tech track, I did meet someone I like. So it's a very difficult thing that we're trying to get around.

**Interviewee 1:**

There's actually a lot of really good diversity initiatives in and around like Ann Arbor in, especially as of late. We just went to intermittent last week, which was a one day conference. That was entirely about diversity in startups. Yeah. Actually it wasn't just about start. It was mostly about startups, but there's also couple of funds that some of the venture are putting together around Detroit that are focused on diverse founders. And so one of the things I actually am really proud of is we have a lot of diversity in the founders that we work with.

**Interviewee 2:**

I was just gonna say that, you know, we have a lot of women and African American people coming to us to help them out and they come out with some great ideas. So it's nice to see that happening.

**Interviewer:**

So the experience with startups is you have a lot of women coming as entrepreneurs, perhaps absolutely ideas. And absolutely. So it's not like biased towards man and doing software development only.

**Interviewee 2:**

I would say honestly, in the last couple years it's been pretty equal. Okay. You know, it's been a pretty equal number.

**Interviewee 1:**

It's getting better. I mean, the industry as a whole is still pretty biased. I think Ann Arbor and Michigan in particular are starting to have a pretty good balance. Okay. For, for startup founders, but I mean, it's, I would say it's a long way from being completely unbiased. Yeah.

**Interviewer:**

And mainly these are tech startups related. Like they have to develop some software solution,I guess.

**Interviewer:**

Very nice. The employee age, I would say your startup people that you get in touch, what is the average age that you could mention? Like, are they in their

**Interviewee 2:**

Twenties? From my perspective, I would say most of our clients tend to be as interesting, mostly 20 ish, early thirties. Okay. Or older 45 in older. We've been getting a lot of in the middle. There's not, I don't see a lot in the middle ground. Yeah. Like 35 to 45 is probably like the least representative.

**Interviewee 2:**

And you're a good representative. Cause most of them are having children, families andmaybe good jobs, stable income.

**Interviewee 2:**

Yeah. I similar to Steve, I, I did startups. Interestingly, I never heard you say that you started doing startups because you were bored. But I was working at Ford Mo I come from large business, gone, went down smaller businesses and I've had several startups myself. And my first startup was in the mid-nineties a small marketing company that I did, you know, my wife was working as a counselor. I was working until late at night. So I had a lot of free time, really bored. I worked in the plants at Ford launching systems there. Yeah. And I get home at three, so I had like five hours every night to do something. And so I just got really bored and like, okay, I'm going to start building a company. You know? So yeah.

**Interviewee 1:**

Yeah. So it's interesting. The, the age group, there's sort of a like bimodal distribution. So the average age is probably the age that we have the least number, the least.

**Interviewee 2:**

The, but what I found was that, you know, I either got advances in my career that made me very busy at that age, you know, 32, 36 years old, or started having children and, you know, helping around of the house and, you know, you know, take care of the Shelton. That's where Steve really started get

**Interviewee 1:**

Involved in. And I just, I happened to get really lucky with one of my past startups in that, you know, I, my startup got acquired right as my fiance and I were getting ready to get married that year or the next year. And so I'm like, I got really lucky that as I started building a family, got a house, had a kid, like I didn't have to go into corporate, you know, into the corporate world to, to support everything. So with it

**Interviewer:**

I would like to discuss a bit more software engineering related questions. Like you mentioned all about CTOs and how you try to provide this as coaching or as consultancy yep. For the different startups. So my questions are more like, what practices do you use? Are they agile, lean? What sort of tools and what kind of programming languages sort of are chosen by your clients or by you? I dunno. Yep.

**Interviewee 1:**

So yeah, so I would say the process is one of the things about having a, a company that at focuses primarily on startups is you're you're basically saying, okay, we're going to focus on the market. That is the most diverse market in, you know within itself. So like every startup is so different from every other startup in a lot of respects in terms of, you know, are they a B to B, are they B to C, are they bootstrapped? Are they venture funded? Are they boots strapped now, but they're going to try to be venture funded at some point in the future. And they just need to get, you know, from a to B are, is it a, a young founder who's right out of school? Is it an older founder who has a lot of, of existing inroads into the market? Do they have family and friends who are putting in money? Are they family, friends within an, you know, a, an actual investor all of those considerations, do they have an existing development team that we're augmenting or are we their team? Are we building a, a, are they at the, a stage and they need to go from that to a minimum viable product or do they have a minimum viable product? And they're trying to get more to like a version two yeah. That they need our help with and all of those things play into the process. So, you know, we, we have a lot of lean startup idea in the, the sort of guidance and feedback that we give. And we have a lot of agile in the, the actual, like project management and scoping processes that we have. I wouldn't say we're a hundred percent, you know, agile process base, or I would also so say anyone who is 100% agile probably missed the point of agile. Yeah. So we, we adapt to some regard for each client in terms of what they need and where they need to go. You know, one of the biggest values I'd say that, that we bring is we, we've had a lot of founders where we meet and, you know, they come cash in hand ready to build the minimum viable product. And we actually take a look at where they are, where they're trying to go and let them know that their next step isn't actually building a minimum viable product, you know, know they can go out with what we call high fidelity mockups, or some wire frames to customers in their target audience. And pre-sell some subscriptions to sort of de-risk you know, the, the, the obstacle of getting first customers before you sync, you know, your, your college savings or something at your kid's college savings into a startup or something like that. So a lot of times we'll kind of

**Interviewee 2:**

We might, we might tell them that they need to discover their customer. Right. Cause they're not ready for it at this point intime.

**Interviewee 1:**

It depends. And we, we have to walk the line because I try not to pass judgment on like the merits of give an idea because you know, who am I to say, whether this idea could work or not. It's more about identifying what I think are the biggest obstacles they'll face and the biggest risks and figuring out how to triage the biggest risks first, where the biggest risk, you know, from a technical standpoint, building something I've feel like is usually the easy part as someone who has built lots of startups. Okay. The actual development, the, of the code and the working product. That's the easy part to me. I think the founders are the ones who have the hard job. They're the ones who have to talk to the customer and, and get the feedback you know, figure out how to actually really do sales and, and convert enthusiasm into actual revenue. That's hard. I've done that. It's really hard. Software problems can be solved. The non-software problems. There's no guarantee they even can be solved. And so that's one of the things that makes those problems particularly hard is you're trying to solve thing that you don't even know can be solved. Yeah. And so you know, so our whole thing is trying to help them before we even get into the software development, understanding where they're coming from and where they're trying to get to what unfair advantages they have at their disposal that they can take advantage of and then how we can fit into that process and move them along. So, you know, we don't start with the default assumption that we're going to build this product that you're asking for right out of the gate, kind of, we like to take a step back and build our way up to that. Then once we actually get to the point where they're ready for a minimum viable product, that's when we jump into setting up the processes and, you know, scoping out what the minimum viable product is going to be, what it's not going to be, what we can put off to a later version, and then, you know, figuring out how to get that done in time

**Interviewee 2:**

As that, as an example, we have, like, we try to be as transparent. You know, our goal is to what I tell a lot of people is that, you know, our goal is to help our entrepreneurs be successful, not just to build code, you talk to other software companies and their goal is to get you in the door and get you to pay for software development. We're there, we're, we're typically de-scoping their project, reducing the cost, trying to get it into something that's manageable and, and something that really focuses on what their product is and what their idea is.

**Interviewee 1:**

And the reason for that is it kind of gets into your question of, you know, what sort of ideologies or processes do we use that help guide us? They're, they're all doing the same thing. They're all trying to maximize the, the problem ability of success. You know, agile is trying to maximize the, the probability of, of successful building and that what you're building is, is going to match the expectations and, you know, lean startup, combining that with the customer feedback loop to maximize the chance that the working thing you built is useful to someone and they act, actually use it. And so, you know, a lot of times there's a sort of stigma against software startups using a consultancy for their software expertise, because, you know, the, the idea that sort of, of echoed is how can you be a software company without an actual software expertise inside your company? Yeah. I think a lot of that comes from the fact that most consultancies are optimizing for billable hours, which is the exact opposite of what an early stage startup needs. A startup needs, maximal effect for minimal hours, both in time and money, whereas a consultancy maximum hours, regardless of effect. I think that that's only true of the consultancy if they have a short term viewpoint of sustainability for their own consultancy. Yeah. Because if a consultancy comes in for a founder and bleeds them, drive their money, all you're doing is you're minimizing ability to address the other parts of their startup, their ability to spend money on advertising or marketing or customer feedback, or, you know, further revisions to the product. Once they start getting customer feedback for iterating. And if you bleed them of their ability to do that, you're just contributing to them ultimately failing. Whereas if we take money out of our own pockets today, by telling you don't give us that money, you're not ready for that. Yet you should do this thing first to make sure this is the right thing to build before we jump in it, it seems like we're taking money out of our own pockets today because we're turning away, you know, cash in hand, but for a longer term viewpoint, if we're maximizing the chances of success for that startup, then that startup is ultimately more valuable to us from a purely, you know, GRE capitalistic standpoint. Like that's still better for us in the long run to make sure you're successful, because then we don't have to go out there and find another customer when you run out of money because your startup isn't successful.

**Interviewee 1:**

And so I think from when you think of it from a consultancy standpoint, it actually, isn't a competing interest when you look at it from a long term perspective from both sides, which is that you both win when you build a successful startup, whatever that entails.

**Interviewer:**

Yeah. It's, I think it's awesome. If you feel it like long term collaboration with all the startups, even though it has challenges, I guess. Oh

**Interviewee 1:**

Yeah, absolutely.

**Interviewer:**

Of all this in place, but I want to keep it at the software level. Yeah. Yeah. Meaning that do you also try to provide some V and V like verification and validation, but not just from the business aspect, but also from the product aspect, like you do testing or I don't know of the software if case

**Interviewee 2:**

Tell your story.

**Interviewee 1:**

Yeah. Yeah. So

**Interviewee 2:**

I come from a larger company, so I want to do a lot of, but yeah.

**Interviewee 1:**

Hear it. Yeah. it's very difficult because there's a balance of, of testing and, and you know, trying to steer clear of, of premature optimization. What, because one of the things that's really tough is when you build a product that works flawlessly that no one wants, you know, and so there's this, there's this real balance of, of making sure that everything is well tested and well validated from a technical side versus getting it in the hands of people. I think the only way, and so you end up the way that we usually end building startups is you know, once we've sort of de-risk and done the customer development and at least validated what our best guess is as to what the minimum viable product should be, we jump in and we build the minimum viable product and release it as soon as possible, just for the purpose of getting the feedback from the user and being able to try.

**Interviewee 2:**

About the process though, is that part of our process is we do minimal testing. What we found is that that takes a lot of money, takes a lot of time and for a heavily changing product, a lot of times it's wasted effort. Cause you know, as you're modifying and evolving the startup, you're going to have to rewrite all that. It adds a lot, lot of a lot of overhead to the project. We literally had someone come to us for me and her co-founder was from Ford or GM. Was it person there put in all the high end testing, you know, back end services, security. And then literally when she can came to they, she needed help. It literally added 20 to 30% more time to each task that we did because there's so much level levels of complexity there.

**Interviewee 1:**

Yeah. So one example of that, and it's actually good segue into what I was I was building toward, which is that it's really easy to, as a software developer from a technical standpoint, to want to build in the tests up front and to make sure that everything works really well. And oftentimes that's not in the best interest of the startup at a really early stage because they have a, they often have a greater chance of failing from building the wrong thing than they do from the thing they built, not working correctly. One example is we had a one startup that was bringing in energy data from the, the utility company. So like using the utility company's API to bring in and then analyzing the, the usage of the household to make some recommendations. And when we first started building it, they actually had a you know, one of the things we had them do before we started building was to make sure that they were had utility at least one utility company on board for giving them the data, because this was before a lot of utility companies had open APIs. And so they actually secured a partnership with a local utility company who said they would send them data. They sent them sample data that we could build the application around. And, and so we actually started with a, a test driven development process and we built tests around the sample data they gave us. And the, the approach that we took was they were giving us data in XML and the XML was a flat hierarchy. So it was literally like just a one level deep X L document with a lot of different tags, but then certain tags had properties on them that said they belonged to another tag. So it was actually nested data, but instead of giving us nested XML, they gave us a flat XML file with attributes that linked to their parent somewhere on the same hierarch level. And so the approach we took was for putting that into our structured database that we could then analyze was to have an initial step that read it all into memory and did the associations and B, and took the flat hierarchy and built it into a structured document.Yeah. And then we would just save the, the structure document all in one go. So rather than like writing each attribute to the database at once, we do all that in memory and then just save the structure document. And it was really fast and efficient. And we built out the entire product around it. And when we finally got close to launching that's when the utility company finally told us, okay, we're ready for you to connect to an actual data set or an actual customer's data. And the first time we connected to the customer's data, the whole application crashed. And it was because what they didn't tell us was that the example document they gave us was not how the API would actually return the data, the API, we thought the, the API would return those specific chunks that you queried for when in reality, they returned the same data. It was just all together. So there was no way for you to query for a section of it. You could only query for the whole thing. Yeah. And when you pulled, when the first time we pulled that whole thing into our process, because it was all in memory, the production data was about a thousand times larger than the sample data they gave us and our whole approach didn't work. And it was a really well tested approach. We tested every part of like getting it in the memory, putting it into the, you know, indefinitely, nested document and then saving it. Yeah. It didn't matter that all the pieces worked flawlessly because it was too big to fit into memory. Yeah. And so we ended up having to come up with a different approach that streamed the data into the database and built the relationships on the fly. Yeah. And so all the tests that we had had to be deleted, we spent all that time, making sure it was well tested and flawless. And then we had to completely change our approach when the application, you know, met production data. And that happens more times than not with startups where entire parts of the fund functionality, not only could the functionality be improved itself, but it turns out that, you know, even the things that you're absolutely certain of as a founder are at best just guesses until you have actual paying customers. And so there's just so many times when, like you realize it's not that the functionality is wrong, that it's the wrong functionality. You Don valid for the solution.

**Interviewee 2:**

And another thing more from a tactical perspective, you know, one thing that we've done is speaking about the money and the financial side, we have our founders do the testing, you know, cuz rather than havingus do it. They're capable because, and a lot of times they have more time than money so's. Is that finding that optimization.

**Interviewee 2:**

Exactly. So that, that really helps them. And like you were just saying, this is sometimes the first time they see their idea in reality and they say, oh, that's not right. We need to change this right now. And then we go in a different direction. And so rather than wasting valuable cycles of development, we end up giving it to them and letting them make those decisions and tell us, is it right? Or is it not right?

**Interviewee 1:**

So the, the main thing I tell people are like early stage startups that are asking questions about testing or, or, you know, the more cowboy approach of, of just building it and getting it out there is I think there's a absolutely an optimal time that you go from sort of seat of the pants development to having an automated test suite that that starts testing it. I usually say the whole point of an automated test suite is to validate that your application meets the assumptions of the customers. The thing is before you have customers, there's no customer assumptions because you don't have customers to make those assumptions. And so I usually say like tests don't make a lot of sense before you have customers. They do make sense sometime after you have customers. So I think there's going to be an optimal time where you have to like, sort of take a step back and say, okay, now we need to start validating the existing functionality of the application with an automated test suite.The problem is the optimal time. You're never gonna be able to tell the optimal time when you're like you're only ever gonna be able to tell what the optimal time was in hindsight. And so the chances that you decide to, to invest in the test suite at exactly the right time or vanishingly, like if there's an optimal moment, you're not gonna hit the moment. You're either gonna be before it or after it. Yeah. And so it's really not a question of how do you know when the optimal moment is to, to start, are building an automated test suite for the code. It's more about the fact that like you're either going to, to start building it too early and encounter premature optimization, or you're gonna do it too late where, you know, you may have like let a bug out in front of a customer or you may have spent too many cycles manually. Qaing something we've where you push it to production. For startups, I, if my choice is to either do it too early or too late, I try to air slightly on the side of too late. Only because too early is typically how you spend money before you have it. Like that's how you do pre you know, end up with premature optimization where you spend ending too much time and money on something before it makes sense. And that can kill a startup, like doing it too late is usually recoverable. And you're usually wasting money when you're in the point in time when you have more money to waste because, you know, you landed some investment or you actually have revenue from customers. And so it's usually like the less of egregious mistake to make is I feel like to do it a little too late versus too early.

**Interviewer:**

Yeah. So you measure also this key performance indicators, perhaps at some level when the startup is out there. And I guess, do you do this often.

**Interviewee 1:**

I would say not very often, it's part of the, the main issue with that is when you're really early on as a startup, you just, you don't have enough users, you don't have enough traffic, you don't have enough interaction for there to be any sort of like statistical meaningfulness to any AB tests or, or, or things that you do. And so I think when you're early on in a startup, it's more of an art than a science. Like you have to make educated guesses and go with your gut and you have to do things that make sense to you and that you can rationalize you. You sort of have to interpret the data with context because without context, there's not enough data to, to draw any conclusions from. So you have to do all the things that you're not supposed to do from a scientific point of view, which is why I say it's like, it's more art early on than science, because you have to do things unscientifically, cuz there's just not enough data to do.

**Interviewer:**

Scientific. And for this early-stage startups that you mentioned now, are they most of the time sort of resource or goal driven, like they look at resources that they have their, their disposal and they try to optimize those resources or do they stick to a goal they have from the beginning and they sort of care too much about the resources.

**Interviewee 1:**

I would say for early stage startups, it's probably more, I would say it's goal driven and resource constrained. Like very few of them have the luxury of, of not, you know, making all of their decisions within the context of the resources they have.

**Interviewer:**

They change goals once they realize they don't have enough enough resources or they stick to the goals?

**Interviewee 2:**

I would say they allow, them stick to the goals and try to figure out how to make it happen right over time. Okay. You know, they may, they may be financially constrained, you know, we've, we've talked to many people who we give an estimate to them and say, Hey, and they're like, wow, I don't have the finances for this right now. And we don't hear from them. But they go away and they start raising money from family, from investors. And then they come back and say, Hey, I'm ready to go. You know? So I, I feel a lot of time, most entrepreneurs are pretty focused on their goal and they try to make it real, make it happen.

**Interviewee 1:**

Okay. Yeah. And I think the trick there is, is understanding what the goal is like, that's the, that's the really tricky part it’s making sure that the goal you're focused on is the right goal. You know, making sure that they're in love with the problem, not the solution because you know, if you're in love with the problem, it allows you to iterate on the solution and even pivot if you need to versus, you know, being stuck on the solution and constantly hunting for a problem. Both I've seen both work. I to tend to prefer the, the, the problem needing a solution versus the solution needing a problem.

**Interviewer:**

Yeah. Yeah. Fully agree.

**Interviewer:**

Which one would you say was the solution looking for a problem?

**Interviewee 1:**

I don't know of any of, I, I can't think off the top of my head, any of our clients, but you see it a lot in start, like you see any startup. I think one of the areas that suffers from this a lot these days are like blockchain startups. That's kind of like the solution looking for a problem because I see a lot of like blockchain for this, where the, this of the start like really doesn't need blockchain. Like it's really just more complicated than it needs to be. And no one wants that. Yeah. So like that's one of those things where I think you see that a lot.

**Interviewer:**

And how about documentation? Do they sort of, are they interested the startups that you work with to document what they're doing or their artifacts?

**Interviewee 2:**

Dunno when it again, when it comes down to finances, documentation costs more money and typically things are early stage startups, things are changing so quickly. Documentation becomes obsolete quickly. Yeah. So typical, you know, we don't, you know, documentation from asort of designperspective.

**Interviewee 1:**

Maybe I would actually, I think there's a lot of documentation that we do that we don't necessarily think of as documentation when we're doing it. Like, think about, you know, when we have a, a meeting with a client and then we follow it up with an email that recaps what we talked about in the meeting. That's documentation.

**Interviewee 2:**

Yeah. Is that what you're talkingabout?

**Interviewer:**

Yeah. Mostly related to software.

**Interviewee 1:**

Actually. Right. And then in the software itself, like we're not writing a lot of like API documentation or anything like that, but how often when we release the software, do we send them an email that as like screenshots of the user interface and explains how the, you know, how the user interface is used and that sort of thing,

**Interviewee 2:**

Ad hoc documentation. Wedo. So, and we actually have a yeah, it depends on the startup for sure. But like we've had a lot of startups where documentation was key because of what they were trying to build and where they're trying to go. You know, we're also trying to make sure that they're not locked into us. Like if they find a good developer, they can bring on full time. Sure. It should be pretty seamless to take over the product. And so like as the project manager, a lot of the stuff you don't see actually is like all the apps that we build have read MES that explain how to install the application onto your local environment. That's true. Bootstrap, the database run the test. If, if they're, you know, later stage and actually have an automated test, we like how to run the test, all that kind of stuff.

**Interviewee 1:**

So we actually do have a fair amount and we even have in our the client dashboard that we built, we have a whole section for documents, like explaining different features, which was really important. We had one startup that we worked with for about two years. And every time we released a new feature, we just sent them an email that like explained how to use it. Like, you know, one sentence, you know, step 1, 2, 3, 4, 5. And using this new feature was screenshots on each step. And we built a custom application on the backend for us, for Startup 1, where if we copy, if we send an email with instructions and we copied our like documents at Startup 1 email address, it went into our project management software, pulled out, you know, all the pictures and the steps into a markdown file. And then we have a custom dashboard that clients use where they can see like the running total for, you know how much UN invoice time there is gonna be on the next invoice, what are their tickets that are open and you know, which ones are in progress and, and all that.

**Interviewee 1:**

And it had another section that kept track of all their documents. So like, we'd email them, here's how to use this. But then if they went into their dashboard, we collected all like how-to s into a section.

**Interviewer:**

So for usability purposes, you do mainly documentation not extend it for software

**Interviewee 1:**

Or yeah. Yeah.

**Interviewee 2:**

Like documentation. The code also is another form that we do. Yeah. So

**Interviewer:**

Yeah. Within the court,

**Interviewee 2:**

Yes. Okay. So I guess when I said documentation, we don't do formal documentation. Yeah. But it's more supporting the project ongoing

**Interviewee 1:**

Project. Yeah. It's, it's very kind of ad hoc, but trying to like keep track of it and collect it after the fact so that when they're like, you know, two years down the road and they bring on a who you know, user experience director or something like, you know, as they're growing, they're starting to specialize in higher, you know, more senior people. Then they can just kind of like, look at the documents to see how this part of the application works versus, you know, having to like figure it out themselves or interface with the us directly or something like that. But it's certainly a balance. It depends at the course, the stage. Yeah.

**Interviewer:**

And maybe sort of the last question when it comes to artifacts, things that we call in software engineering that are relevant and you can reuse them and you can how do you protect them for the startup or how do that you are working with protect the artifacts? Because as we know, like the source code could be an artifact and that's very relevant, very important. Documentation could be one of them testing would be another. So we are not teaching that a lot to be on list, how to protect your how to put IPS on your software or you tackle these things with your startups or is it

**Interviewee 1:**

Yeah. Yeah. So one thing that, how, yeah, so one thing that we do that's different than a lot of other consultancies is a lot of times when you work with a, a, a consult a development consultancy, the way their contract is structured is they retain ownership of the code and grant you an exclusive royalty, free perpetual license to use the code, which is horrible for startups because a startup has to have they have to be building their their assets, their IP, and for a software startup. A lot of that is the code. And so for startup to be able to, and not all startups, you know, need to, or should, but for a startup to be able to raise money and eventually be acquired, they have to be able to own the things that entice, you know, the, the investment or the acquisition. And so that's one of the things we do a bit differently is we just assign outright ownership of the code that we write to the startups so that they have that so that they can raise money and they can be acquired.

**Interviewee 2:**

We have them create accounts and bit bucket and they give us access. So they, they, they maintain control of all of it. You know, if, if we use Stripe it's funny, you're more strategic on more tactical. We're talking here, I'll tell you how, how we're doing it, you know, but when, when we do it, they create the accounts and give us access. So when they leave, they don't have to come back. Oh, can I have access to our Stripe account or this account? They have it. All right. Okay. And so they, when they, you know,

**Interviewer:**

How do they know you are not going to disclose this perhaps to some third party or whatever can?

**Interviewee 1:**

Yeah, yeah. Yeah. And, and, you know, it's there's also been, we've had this discussion with several founders too, where there's, even when you have a contract that spells out, like, you know, what are the, what is the IP of the company me what's shareable and what's not, and all that. It can be difficult because a lot of times you want to protect the knowledge, learn from building your application, which just, you know, as a, as a consultancy, really, even if you hired a full-time developer, like you can't prevent them from using, you can't make them forget something they learned. No. Right. And so, like, even if they say won't use the knowledge, like there's aspects as humans, we extrapolate from the knowledge that we learn anyway. So it's not like when we learn something, we only learn that one thing, we can extrapolate that to a hundred other lessons as well. And so one of the things that we tell them is, you know, if we learn how to better interface with like a third party API or something like that, it's not like we're not going to, like, if we get some other client who needs interface with that API, we're not gonna charge them to relearn something we already knew. Right. But, you know, we also explain that, like, you also get the benefit of all the other startups that we've worked with and all the things we've learned that do, and don't work. We're bringing those to you too. So it's more of a, a collaborative effort from, from the, the non-technical aspects of just like the knowledge you learn for doing certain things. It's more about you know, making sure that we don't use that knowledge for a competitor. And also making sure that the startup maintains their IP. You know, when you talk about IP, there's like the traditional, you know, patent strategy, which I would say in startups, it's tough. One of my mechanical engineering professors used to tell us a patent doesn't protect you. It only gives you the legal right to spend all of your money, protecting yourself. Yeah. And so a, a patent is relatively worth list to someone who doesn't have the money to defend and, and uphold the patent which is most startups. However, they're really valuable to anyone who does have the money to defend and uphold the patent and certain groups of people that have the money to defend a patent are investors and potential acquirers. Yeah. And so while a patent isn't going to help you execute better. And it's probably not also not going to help you protect yourself against competition, because you just don't have the resources to enforce it. What it is going to do is it's going to increase your potential valuation from an investor or an acquirer. If you come to that point of raising money or being

**Interviewer:**

Acquired, but you think it's important to protect the source code, the software for startups, because it's a lot of effort. I would say if even if they spend months still throwing it away to somebody else and maybe raises up competition based on, on the board actually on implementation, not just the, the knowledge, because knowledge, of course, it's a bit, and the challenge with software is that it's not physical. We can touch it. And and it's easier to distribute. Yeah. Yes. So there is a lot of challenge. This, we need to also address at university level. I'd say how to explain to students that what are these old companies are doing that don't want to tell other people they want to keep for themselves like source code or whatever things, artifacts.

**Interviewee 1:**

Absolutely. And we've, we've gone through several processes related to that too. You know, we had one startup where we've had actually a few that were spin outs of universities and what ends up happening is before they spin out their, you know, going through the tech transfer office and, and getting grant funding or funding from the tech transfer office to build things. And so we've had some startups where we've had to go through the process of once they spin out, figuring out which parts of the application are owned by the university and therefore, you know, being licensed to the startup sure. Versus which things then belong directly to the startup because they're being built outside of the tech transfer hours after they spun out. So there's a lot of considerations for how you architect the code to make that a little bit easier. We've gone through this question actually going through the acquisition of my last company that I co-founded figuring out how to make sure that so all, all, all software these days is, you know, all startups are being built on the shoulders of giants. So you're not rebuilding everything from scratch. You're using open source libraries, a lot of the time open source frameworks. And so a lot another thing people often don't think about is how the things they're using our license and that they're allowed to use it. For example, you know, using an open source library, that's MIT license versus GPL license, or a lot of people don't realize that the code snippets and stack overflow, you, you Google for, you know, some bug or some, some error message. You find a solution for it on stack overflow, you copy paste the code into your code, the code snippets in stack overflow, even though they're contributed by the community, they have a blanket license on stack overflow. That's more akin to a GPL license, meaning you're not technically allowed to modify that code without releasing your modified code back to the community as well. And so one of the things that, you know, I always do when I'm copy, like if I have to copy a snippet from stack overflow into the code is I'll copy it in and put a code comment above it that says adapted from this part of stack overflow. Make sure if you modify this block of code, you release it somehow, you know, or just don't modify this block of code. And so, you know, that's one thing that we've gone through from the acquisition process is making sure, like when we went through the due diligence of the acquisition on the technical side, we had to walk through our code with the attorneys and, you know, verify that we weren't using any code that we weren't allowed to use in our proprietary product. Sure. I mean, that was actually a big sticking point, cuz it turned out to be something that we were more knowledgeable about than the acquirer's attorneys. So were asking us to acknowledge that every piece of code used by our, our application and by any libraries or, or libraries of libraries is all properly licensed. And we actually asserted that that's practically impossible to verify the level they were saying. And so what we came back with was we'll verify that our code and our codes direct dependencies are properly licensed, but if you want to verify everything is properly licensed. Like that literally means going down to machine code into the operating system level to verify that I said, I would assert that if you knew what were asking your own company would not be able to verify that about your own code, because it's just, it would take me six months to figure it out if ever if EV right. And it would be very difficult. And so we actually had to push back and change that clause in the acquisition paperwork to be more reasonable and not just be a, you know, infinitely nestable thing that we had to verify. So that was interesting, but yeah, so there's lots of things I think to consider about the IP, not just from the patent side, but you know, owning the code and the licensing and all that.

**Interviewer:**

Have you guys ever heard about technical depth that people are getting out of the software? Do your startups get shortcuts instead of the doing the right thing or the right solution if they have, so they get a lot of technical debt, you

**Interviewee 1:**

Would say it depends technical debt isn't bad. In fact, if you have no technical, that's the thing like it's, it depends, right? Because if you have no technical debt, it meant you spent way too much time and money building it before you got out and started verifying. So it's actually, you know, if you have an application, if you have a, if you're a startup and you're building an application with no technical debt, I would argue you built it. Too rigorously. Yeah. Well not, not smartly. Yeah. You know, like that, wasn't a Smart use. Well,

**Interviewer:**

It goes back to the very beginning of our discussion that what are you doing? Are you trying to do the perfect software or are you trying to find the best solution?

**Interviewee 1:**

Exactly. So

**Interviewer:**

Yeah, it kind of have a balance.

**Interviewee 1:**

Yep. And so one of the other things I, I like to tell, especially when you can, you know, a lot of the founders we're working with on technical. And so we're trying to, to help them understand the technical side of things. And one of the things I, I tell people doing early stage startups, especially is one of the ways you can kind of think of developers and what they can do for you is you have sort of the like junior mid-level developers where they can, you can get a lot done with them the right way where you can get a lot done with them. If you explain to them not just what you need built, but how to build it. Yeah. And they can follow that sort of recipe. The problem is to, to be able to describe at that level of detail, how to build it, it means you need to have someone like a senior developer on your team to do that.

And if you're an early stage startup, you're probably just going to cut out middleman and, and have that person build the app. And so the, then you have senior level developers where you don't have to tell them how to build it. You can just tell them what to build and they they'll figure out how, but that gets to what I was saying earlier about with a lot of software consultancies, they'll have those senior level developers and the best case scenarios, they build exactly what you ask for. You know, they figured out how and built exactly native

**Interviewer:**

Developed

**Interviewee 1:**

System. They, I, there's a subset of senior level developers that I call the highly creative senior level developers with them. You don't even have to tell them what to build. You just tell them why you're building it and they help you figure out what to build. And so that kind of gets to your question about technical debt, because they're the, who have the, the experience and the creativity, not just the, the technical knowledge of how to build something efficiently or quickly or well tested because I've, I've worked with a lot of senior level developers who will spend, you know, two months building the perfectly scalable tested architected solution that no one really wants or that you know, was meant to be a piece of your product that could generate X level incremental revenue, but they built it as if it was your core business. And like, you're never gonna make that money back that you just spent on them building it. And so having the highly creative senior level developer who can put what they're doing in the context of why they're doing it and use that to drive those sort of judgment calls, I think is what's important. So it's, it's, you don't wanna deliver something to a startup, particularly with zero technical debt. If you're building something for like a bank or a health system may. Yeah. Yeah. But if you're building something for an early stage startup, most of the time, you don't want to do that, you also don't want to do something that built them into a corner. I eat too much technical debt. Yeah. So it's all about finding like the right amount of technical debt that gets them to the point of being able to, you know, de-risk the next assumption or validate the next assumption. And de-risk the next

**Interviewer:**

You say in English, the golden middle or the golden average.

**Interviewee 1:**

Yeah.

**Interviewer:**

Yeah. So that's the right thing. Yeah. But thank you very much. I have next.