Tips For Technical Startup Founders | Startup School

Video URL

Intro

Reference

How to Build and Perpetuate as a Technical Founder

Reference

succeed as a technical founder for the startup School talk quick intro I'm Diana who I'm currently a group partner at YC and previously I was a co-founder and CTO for Azure reality which was a startup building augmented reality SDK for game developers and we eventually had an exit and sold to Niantic where I was the director of engineering and heading up all of the AR platform there so I know a few things about building something from was just an idea to then a prototype to launching an MVP which is like a bit duct tapey to then scaling it and getting to product Market fit and scaling systems to millions of users so what are we going to cover in this talk is three stages first is what is the role of the technical founder and who are they number two how do you build in each of the different stages where all of you are in startup school ideating which is just an idea you're just getting started building an MVP once you got some validation and getting it to launch and then launch where you want to iterate towards product Market fit and then I'll have a small section on how the role of the technical founder evolved Pro product Market fit I won't cover it too much because a lot of you in startup School are mostly in this earlier stage and

I'm excited to give this talk because I compiled it from many conversations and chats with many YC technical Founders like from algolia segment optimal easily way up so I'm

What Does a Technical Founder Do?

Reference

founder sometimes I hear non-technical Founders say I need somebody to build my app so that isn't going to cut it a technical founder is a partner in this whole journey of a startup and it requires really intense level of commitment and you're in just a Dev what does a technical founder do they lead a lot of the building of the product of course and also talking with users and sometimes I get the question of who is the CEO or CTO for a technical founder and this is a nuanced answer it really depends on the type of product the industry you're in the complete scale composition of the team to figure out who the CEO of CTO is and I've seen technical Founders be the CEO the CTO or various other roles and what does the role of the technical founder look like in the early eight stages it looks a lot like being a lead developer like if you've been a lead developer a company you were in charge of putting the project together and building it and getting it out to the finish line or if you're contributing to an open source project and you're the main developer you make all the tech choices but there's some key differences from being a lead developer you got to do all the tech things like if you're doing software you're gonna have to do the front and the back end devops the website the ux even I.T to provision the Google accounts anything if you're building hardware and maybe you're just familiar familiar with electrical and working with eaglecad you'll have to get familiar with the mechanical too and you'll of course as part of doing all the tech things you'll have to talk with users to really get those insights to iterate and you're going to have a bias towards building a good enough versus the perfect architecture because if you worked at a big company you might have been rewarded for the perfect architecture but not for a startup you're going to have bias towards action and moving quickly and actually deciding with a lot of incomplete information you're gonna get comfortable with technical debt inefficient processes and a lot of ugly code and basically lots of chaos and all of these is to say is the technical founder is committed to the success of your company and that means doing whatever it takes to get it to work and it's not going to cut it if you're an employee at a company I sometimes hear oh this task or this thing is not in my pay grade no

How To Build

Reference

on how to build the first stage is the ideating stage where you just have an idea of what you want to build and the goal here is to build a prototype as soon as possible with the singular Focus to build something to show and demo to users and it doesn't even have to work fully in parallel your CEO co-founder will be finding a list of users in these next couple days to TF meetings to show the Prototype when it's ready so the principle here is to build very quickly in a matter of days and sometimes I hear it's like oh Diana a day prototype that seems impossible how do you do it and one way of doing it is building on top of a lot of prototyping software and you keep it super super simple so for example if you're a software company you will build a clickable prototype perhap using something like figma or Envision if you're a devtools company you may just have a script that you wrote in an afternoon and just launch it on the terminal if you're a hardware company or heart attack it is possible to build a prototype maybe it takes you a little bit longer but the key here is 3D renderings to really show you the promise of what the product is and the example I have here is a company called Remora that is helping trucks capture carbon with this attachment and that example of that rendering was enough to get the users excited about their product even though it's hard tech so give you a couple examples of prototypes in the early days this company optimizely went through YC on winter 10 and they put this prototype literally in a couple of days and the reason why is that they had applied with YC with a very different idea they started with a Twitter referral widget and that idea didn't work and they quickly found out why so they strapped together very quickly this prototype and it was because the founders uh Pete and Dan and Dan was actually heading analytics for the Obama campaign and he recalled that he was called to optimize one of the funding pages and thought huh this could be a startup so they put a very together very quickly together and it was the first visual editor by creating a a b test that was just a Javascript file that lived on S3 I literally just opened option command J if you're in Chrome and they literally run manually the A B test there and it would work of course nobody could use it except the founders but it was enough to show it to marketers who were the target users to optimize sites to get the user excited so this was built in just few days other example is my startup Azure reality since we're building more harder Tech we had to get computer vision algorithms running on phones and we got that done in a few weeks that was a lot easier to show a demo of what AR is as you saw on the video than just explaining and hand waving and made selling and explaining so much easier now what are some common mistakes on prototypes you don't want to overbuild at this stage I've seen people have this bias and they tell me hey Diana but users don't see it or it's not good enough this prototype doesn't show the whole Vision this is the mistake when founder things you need a full MVP and the stage and not really the other mistake is obviously not talking or listening to users soon enough that you're gonna get uncomfortable and show this kind of prototyping duct type thing that you just slap together and that's okay you're gonna get feedback the other one at the stage as an example for optimizely when founders get too attached to idea I went up the feedback from users is something obvious that is

Build an MVP: The Startup Process

Reference

ideas okay so now into the next section so imagine you have this prototype you talk to people and there's enough interest then you move on to the next stage of actually building an MVP that works to get it to launch and the goal is basically build it to launch and it should be done also very quickly ideally in a matter of can be done a few days two weeks or sometimes months but ideally more on the weeks range for most software companies again exceptions to hardware and deep tech companies so the goal here at this stage is to build something that you will get commitment from users to use your product and ideally what that commitment looks like is getting them to pay and the reason why you have a prototype is while you're building this your co-founder or CEO could be talking to users and showing the Prototype and even getting commitments to use it once is ready to launch so I'm gonna do a bit of a bit of a diversion here because sometimes Founders get excited it's like oh I show this prototype people are excited and there's so much to build is hiring a good idea first is thing is like okay I got this prototype got people excited I'm gonna hire people to help me to build it as a first-time founder he's like oh my God oh my God there's a fit people want it is it a good idea it really depends it's gonna actually slow you down in terms of launching quickly because if you're hiring from a pool of people and Engineers that you don't know it takes over a month or more to find someone good and it's hard to find people at this stage with very nebulous and chaotic so it's going to make you move slowly and the other more Insidious thing is going to make you not develop some of the insights about your product because your product will evolved if someone else in your team is building that and not the founders you're gonna miss that key learning about your tag that could have a gold nugget but it was not built by you I mean there's exceptions to this I think you can hire a bit later when you have things more built out but at this stage it's still difficult so I'll give you a example here uh Justin TV and twitch it was just the four Founders and three very good technical Founders at the beginning for the MVP it was just the founders building software as software engineers and the magic was Justin Emmett and Kyle Building different parts of the system you had Kyle who become an awesome Fearless engineer tackling the hard problems of video streaming and then Emma doing all the database work Justin with the web and that was enough to get it to launch I mean I'll give you an exception after they launched they did hire good Engineers but the key thing about this they were very good at not caring about the resume they try to really find The Misfits and engineers at Google overlooked and those turned out to be amazing so Amon and Golem were very comfortable and awesome engineers and they took on a lot of the video weapon just three months since joining

Principles for Building Your MVP

Reference

going back into the principles for for building towards your MVP principle one is the classic hologram essay on do things that don't scale basically find clever hacks to launch quickly in the spirit of doing things at those scale and the Drake posting edition of this avoid things like automatic self onboarding because that adds a lot of engineering building a scalable back-end automated scripts those sounds great at some point but not the stage and the hack perhaps could be manually onboarding you're literally editing the database and adding the users or the entries and the data on the other counterter thing is insane custom support it's just you the founders at the front line doing the work doing things that don't scale a classic sample is with stripe this is the site when they launch very simple they had the API for developers to send payments but on the back end the thing that did not scale it was literally the founders processing every manual request and filling Bank forms to process the payments at the beginning and that was good enough to get them to launch sooner now principle number two this is famous create 9010 solution that was coined by Paul bukite who was one of the group Partners here at YC and original inventor of Gmail the first version is not going to be the final remember and they will very likely a lot of the code be Rewritten and that's okay push off as many features to post launch and by launching quickly I created a 9010 solution I don't mean creating bugs I still want it good enough but you want to restrict the product to work on limited Dimensions which could be like situations type of data you handle functionality type of users you support could be the type of data the type number of devices or it could be Geo find a way to slice the problem to simplify it and this can be your secret superpowers that startup at the beginning because you can move a Lot quickly and large companies can't afford to do this or even if your startup gets big you have like lawyers and finance teams and sales team that make you kind of just move slow so give you a couple examples here doordash at the beginning they slapped it in one afternoon soon and they were actually called Palo Alto delivery and they took PDS for menus and literally put their phone number that phone number there is actually from one of the founders and there's the site is not Dynamic static it's literally just plain HTML and CSS and PDF that was our front end they didn't bother with building a back end the back end quote unquote was literally just Google forms and Google Docs where they coordinated all the orders and they didn't even build anything to track all the drivers or ETA they did that with using fancy on your iPhone find my friends to track where each of the deliveries were that was enough so this was put together literally in one afternoon and they were able to launch the very genius thing they did is that because they were Stanford student they constrained it to work only on Palo Alto and counterintuitively by focusing on Palo Alto and getting that right as they grew it got them to focus and get delivery and unit economics right in the suburbs right at the beginning so that they could scale that and get that right versus the competition which was focusing on Metro cities like GrubHub which make them now you saw how the story played out the unit economics and the Ops was much harder and didn't get it right so funny thing about focusing at the beginning and getting those right

Choose the Tech Stack That Makes Sense for Your Startup

Reference

at this stage how do you choose a tech stack so what one thing is to balance what makes sense for your product and your personal expertise to ship as quickly as you can keep it simple don't just choose a cool new programming language just to learn it for your startup choose what you're dangerous enough and comfortable to launch quickly which brings me to the next principle choose the tag for iteration speed I mean now and the other thing is also it's very easy to build MVPs very quickly by using third-party Frameworks on API tools and you don't need to do a lot of those work for example authentication you have things like auth zero payments you have stripe cross-platform support and rendering you have things like react native Cloud infrastructure you have AWS gcp landing pages you have webflow back-end back-end serverless you have lambdas or Firebase or hosted database in the past startups would run out of money before even launching because they had to build everything from scratch and shift from metal don't try to be the kind of like cool engineer just build things from scratch no just use all these Frameworks but I know ctOS tell me oh it's too expensive to use this third-party apis or it's too slow it doesn't skill to use XYZ so what I'm going to say to this I mean there's there's two sides of the story with using third party I mean to move quickly but it doesn't mean this this is a great meme that Sean Wang who's the head of developer experience that everybody posted the funny thing about it is you have at the beginning quartile kind of the noob that just learned PHP or just JavaScript and just kind of use it to build the toy car serious engineers make fun of the new because oh PHP language doesn't scale or JavaScript and all these things it's like oh our PHP is not a good language blah blah and then the middle or average or mid-wit Engineers like okay I'm gonna put my big engineer pants and do what Google would do and build something optimal and scalable and use something for the back end like Kafka Linker Ros AMA Prometheus kubernetes Envoy big red or hundreds of microservices okay that's the average technical founder the average startup dies so that's not a good outcome another funny thing you got the Jedi Master and when you squint their Solutions look the same like the new one they chose also PHP and JavaScript but they choose it for different reasons not because they just learned it but they wreck recognizes this is because they can move a lot quicker and what I'm going to emphasize here is that if you build a company and it works and you get users good enough the tech choices don't matter as much you can solve your way out of it like Facebook famously was built on PHP because Mark was very familiar with that and of course PHP doesn't quite scale or is very performant but if you're Facebook and you get to that scale of the number of users they got you can solve your way out and that's when they built a custom transpiler called hip hop to make PHP compound C plus plus so that it would optimize see so that was the Jedi move and even for JavaScript there's a V8 engine which makes it pretty performant so I think it's fine way up was a 2015 company at YC that helps company hire diverse companies and is a job board for college students so JJ the CTO although he didn't formally study computer science or engineering at UPenn he that taught himself how to program on freelance for a couple years before he started way up and JJ chose again as the Jedi Master chose technology for iteration speed he chose Django and python although a lot of other peers were telling him to go and use Ruby and rails and I think in 2015 Ruby and rails were 10 times more popular by Google Trends and that was fine that that didn't kill the company at all I mean that was the right choice for them because he could move and get this move quickly and get this out of the door very quickly I kept it simple in the back end postgres python Heroku and that worked out well for them now I'm going to summarize here the only Tech choices that matter are the ones tied to your customer promises for example at Azure we in fact rewrote and threw away a lot of the code multiple times as we scale in different stages of our Tech but the promise that we maintain to our customers was at the API level in unity and game engines and that's the thing that we cannot throw

What Happens In The Launch Stage?

Reference

part three so you have the MVP you built it and launched it now you launched it so what happens on this stage your goal here in the launch stage is to iterate to get towards product Market fit so principle number one is to quickly iterate with hard and soft data use hard data as a tech founder to make sure you have set up a dashboard with analytics that tracks your main kpi and again here choose technology for your analytics stack for Speed keep some keep it super simple something like Google analytics amplitude mix panel and don't go overboard with something super complex like lock stash Prometheus these are great for large companies but not at your stage you don't have that load again use Soft Data if I keep talking to users after you launch and marry these two to know why users stay or churn and ask to figure out what new problems your users have to iterate and build we pay another YC company when they launch they were at b2c payments product kind of a little bit like venmo-ish but the thing is that it never really took off they iterated so in terms of analytics they saw some of the features that we're launching like messaging nobody cared nobody used and they found out in terms of a lot of the payments their biggest user was GoFundMe back then they also talked to users they talk to GoFundMe who didn't care for any of this b2c UI stuff they just care to get the payments and then they discover a better opportunity to be an API and basically pivoted it into it and they got the first version and again applying the principles that did a scale they didn't even have technical docs and they worked with GoFundMe to get this version and this API version was the one that actually took off and got them to product Market fit principle number two in this launch stage is to continuously launch perfect example of this is a segment who started as a very different product they were classroom analytics similar stories they struggled with this first idea it didn't really work out until they launched a stripped out version of just their back end which was actually segment and see the impressive number of launches they did their very first launch was back in December 2012. that was their very first post and you saw the engagement in Hacker News very high that was a bit of a hint of a product Market fit and they got excited and they pivoted into this and kept launching every week they had a total of five launches in a span of a month or so and they kept adding features and iterating they added support for more things when they launched it only supported Google analytics mixpanel and intercom and by listening to the users they added node PHP support and WordPress and it kept on going and it took them to be then a unicorn that eventually had an exit to Twilight

When You Launch: The Right Way to Build Tech

Reference

the last principle here what I want to say for when you're launch there's this funny state where you have Tech builds you want to balance building versus fixing you want to make thoughtful choices between fixing bugs or adding new features or addressing technical debt and one I want to say Tech debt is totally fine you gotta get comfortable a little bit with the heat of your Tech burning totally okay you're gonna fear the right things and that is towards getting you product Market fit sometimes that tiny bug and rendering maybe is not critical for you at this point to fix like in fact a lot of early products are very broken you're probably very familiar with Pokemon go when it launched in 2016 nobody could log into the game and guess what that did not kill the company at all in fact to this day Pokemon I think last year made over a billion dollars in Revenue that did not kill them and I'll give a little background what was happening on the tech it was very uh very straightforward they had a load balancer that was on Google cloud and they had a back-end and they had a TCP termination and HTTP requests that were done with their nginx to route to the different servers that were the AFE the application front end to manage all the requests and the issue with there it was that as users were connected they didn't get terminated until they got to the nginx and then as a result client also had retries and that what happened when you had such a huge load that in fact I think Pokemon go by the first month after launching they had the same number of uh active as as Twitter which took them 10 years to get there and they got there in one month of course things would break it was basically a lot of users trying to log in was kind of creating a bit of a dito's attack now December is a bit on when you launch some of the common mistakes after launching and I myself has made CTO Doge sad it is tempting to to build and say what would Google do that's almost certainly a trap would try to build like a big company or hiring to try to move quickly sometimes I think this is more of a nuanced question can be a mistake or the other thing is focusing too much on fixing refactoring and not building features towards iterating to product Market fit not discovering insights from users sometimes I see ctOS like okay we launched I get to conquer down and just get into building totally no again your role as a technical founder very different you got to be involved in the journey and really understand the insights of why users Stay or Leave Your products you have to keep talking to them and the other mistake I see is like oh we're just building features for their product but you also need to build Tech to grow in fact some of the best growth hacks where Engineers pair it up with sales

How the role evolved from ideating to hiring

Reference

and growth folks who are non-technical so now the last section on how the role evolves so assuming you got product Market fit what happens this is this point where you can actually then put on your big engineering pants and figure out pieces of the tech that need to be built to scale you need to and the attack will break which is actually a good thing breaking because of too much demand and that's totally okay that's my example from Pokemon go you'll find the pieces that need to be reworked refactor this is when you do it not before now not before product Market fit and you'll decide also what the engineering culture will look like and this is a stage where you actually do more of the hiring and here you're probably going to evolve from leading a small team of Engineers to hiring your first hires who are going to be people that you know and at this point Your Role really changes because you'll start having communication overhead and this is when you realize your role morphs like between two to five you still get time to code about 70 when you get to five to ten you only have less than 50 percent and Beyond 10 you probably won't really have time to code and have to decide how to structure things and whether you're going to remain as a architect type or role or you want to be more of a people role and be more of a BP rich now to summarize uh hear the talk first stage ideating Bill the goal is to build a prototype as soon as possible and the principle is built very quickly in a matter of days stage two you're in the process of building an MVP which I think a lot of you are in this or the previous one the goal is to build as quickly to launch in a matter of few weeks and the principles are do things that don't scale create a 90 10 solution choose the tech for iteration speed and the last one is once you launch all of the previous ideas on 9010 solution do things that don't scale still apply and add these onto it and the goal is to get an iteration towards product Market fit so you're going to also quickly iterate with hard and soft data with analytics and user interviews you're going to continuously launch and you're going to find the fine balance between building and fixing and where techdat is totally fine feel the heat for that Tech that is totally fine and if there's only one take away from this whole talk is that startups move quickly so thank you everyone

Prepared by: Abdulmajeed Al-Muarik

Note:

This automated transcript serves as a testament to the invaluable content created by Y-cominbtor. Their exceptional expertise and captivating delivery have made this video an absolute gem for knowledge seekers like myself. As a personal project, I embarked on the mission of transcribing this remarkable content. To help individuals effortlessly follow along and absorb the valuable insights shared within.