What is a product manager?

- [Narrator] The ambiguity of the product management role is near to its essence. - Welcome back, I hope you're excited about starting this course. Let's go ahead and dive right in. In this lecture, we're going to talk about what product management actually is, and by the end of this lecture, you'll have a really solid understanding of the role, and hopefully be even more excited about the rest of the course. Does that excite you? If it doesn't, you can't be a product manager. You should probably leave. I'm kidding, but this Venn diagram pretty much sums up my entire existence. I also love graphs. So, product management is actually a pretty difficult role to define, which is why you don't really find any concrete, concise, good explanations just by googling it. Try googling it right now and I challenge you to find a really good description. In fact, the role of a product manager and their exact responsibilities change across different industries in different companies. If at one company, you're a product manager, and you have one set of responsibilities, you might have a completely different set of responsibilities at another company depending on the company, the industry, and the size. What we'll focus on here are the core tenets that make up product management. The first, and most important thing to know about product management is that you're actually not a manager of anybody. No one reports to you, you're nobody's boss, and you definitely can't fire anyone. This is actually by design because a product manager needs to continuously interact with a number of different people, get maximum collaboration from their engineers, and also the designers they work with. Think about it for a second. If you wanted honest feedback from engineers and designers that you work with, but you're also their boss, do you think they're going to tell you if they have any disagreements with what you have to say? Probably not. So you want them to tell you every single time that they disagree with you. A product manager is someone that sits in between multiple areas of a company and acts as a communications hub, organizer, and enabler for everyone else. So, here's kind of a funny story. We once had a new engineer join our company, and he asked, "What do you do?" He hadn't worked with a product manager before. And before I

could answer, the other engineer said, "That's the person that stops sales and legal from emailing you!" (canned laughter) As a product manager, you're kind of a blocker slash enabler for engineers and designers. The communication and enabling aspect of product management is a very important one. Engineers are very good at engineering, and designers are very good at designing. So, it's best if these people spend their time doing what they're best at. While engineers focus on solving hard technical challenges, product managers are talking to various stakeholders, users, receiving feedback, and looking at metrics to decide what things can be built next, and which of those things is most important, and the best use of time. Have you ever used a product, either software or something physical, where many things about that product are really good, but then there's one small piece that's just really terrible? If there's something wrong with any product, you can blame the product manager. Basically, a product manager is someone who's responsible for that product being successful or not. To recap, the product manager is many things, a communications hub, a prioritizer, a researcher, a presenter, but most importantly, they're responsible for the ultimate success of the product. So, your next question is probably, "All right, so what's a product?" Well, I'm glad you asked because that's what we're going to talk in the next lecture.

What is a product?

- Hey, welcome back. So, what is a product? Bear with me here because I know this seems like a silly question, but it's actually a topic that you need to really pay attention to as a product manager. Obviously everyone knows what a product is. It could be anything. Could be a keyboard that I have here in front of me. It could be a laptop, or the device you're using right now. My shirt is a product. The web browser that you have on your computer is a product. All of these things are products by the classical definition. So we know from the title, product manager, that a product manager is someone that manages a given product. However, it's not always the case that a product manager is in charge of an entire device or piece of software. In many startups, and especially large companies, product managers are in charge of

sections of what we normal consumers would typically think of as a product. I'll give you an example here. When you use Facebook, you're actually using a multitude of different features. As companies like this, each feature is so vital and complex that it's usually assigned to a group of people called a product team, or a feature team, to manage it. So at Facebook, the photos are one product. It's a feature. It's called product in product management lingo. The newsfeed is a product. The user profile is a product. Messaging and commenting, those are both products too. Every single one of these product is managed by a product manager, and worked on by them and a team of designers and engineers, which again, are called product teams or feature teams. Let's go a little bit deeper with that Facebook example. The newsfeed at Facebook has multiple product managers working on it, believe it or not, because it's such a big piece of technology. One product manager leads a team that works on the ranking algorithm. One product manager leads a team that works on the advertisements. And there are a bunch of other product managers working on different things that have to do with that newsfeed. But there are also cases where product managers are not split up by the features in larger applications, or products like Facebook, or by pieces of software, but rather split up by platform. For instance, at a company, there may be a product manager in charge of everything on the Android app, and another one in charge of everything on IOS, and another one in charge of the website. This is just a different model of working. There are also product mangers that work on things like advertising and subscription services depending on the business model of the company. We'll talk more about the different types of product managers in a little bit. So now we should be clear that as far as the role of a product manager goes, a product can be anything from an entire real life product as a consumer sees it, or in some cases, split up into smaller sections for larger companies, or more complex products. All right, in the next lecture we're going to talk about the different types of product managers. I will see ya there.

Three different types of product manager roles

- What is up, everybody? In this lecture, we're going to talk about the different kinds of product managers. We're also going to learn some buzzwords, so yay for that. So there's more than one type of product manager in the world of software development and technology. Three that we're going to talk about right now are internal product managers, consumer product managers, and business to business product managers, which are also sometimes called SAAS product managers, or Software as a Service. So are you confused yet? It's fine, don't be. The primary difference in these roles is their stakeholders, and that is the buzzword I was talking about. Stakeholders is a fancy term for people who you are building for or who have input into what you are building. So can you think of some examples or stakeholders? Well, users are stakeholders, and so are executives that rely on what you're building to be good. Lawyers at your company that need to be sure what you're building won't get the company in legal trouble, those are also stakeholders. Marketing is also a stakeholder because they need to know what you're building and they want to have some input because they have to market good stuff. Anyways, enough with the stakeholders; let's talk about the internal product manager. These product managers are sometimes part of a team called internal tools, or something like that. They and their teams usually build tools for other people in their company organization. They build these tools for use internally as opposed to building for some general user or customer out in the public. A really good example of an internal product manager is one that works with her team to build a piece of software that, say, the company support team uses to reset passwords or change account information for users. The stakeholder is someone internally. It's the person that's using that piece of software. So how 'about the business to business product manager, or SAAS or Software as a Service? This is a type of product manager that builds products at a company whose clients are other companies. Some examples of this are Oracle or Salesforce. Those products are designed to solve problems for other companies. At companies like this, the stakeholders, or the people that the product manager is working with their team to build for are these other companies. This means that the product manager interacts a lot with the salespeople at their own company, and needs to make sure what they build meets the business requirements of the businesses that they're selling to. If you're a B to B

product manager, you're going to be talking a lot to sales teams. So on to the third type, business to consumer product manager. What does a consumer product manager do? Consumer product manager tends to be more common. It just means you're a product manager where the product is for an average consumer. I myself am a consumer product manager at SoundCloud, because I work on parts of the app that millions of people out in the public use every day. Some other examples where there's a lot of consumer product managers are Facebook and Twitter and Instagram, this sort of thing. Consumer product management role takes a wide range of skills to be successful in, as well as a lot of vision and creativity. In the consumer role, product managers are trying to maximize usage or some other metric. No one is there to tell them that item X is the best thing to do next. That contrast to the business to business product manager, for example, because the salespeople are saying, "Hey, we're going to make $5 million if we build this feature." On the consumer side, no one can really predict exactly what is the thing that needs to be built for sure. So the consumer PM spends a whole lot of time talking to those users, those users, the millions of 'em out there, coming up with multiple different prototypes, user testing, AB testing, and just analyzing tons of data. The bigger the user base, the more powerful the testing becomes. All right, so that's it for the three types of product managers. We have an internal tools product manager, a business to business product manager, and a business to consumer product manager. Now there are other types of PMs out there, but those are really the big three you got to be familiar with. I will see ya in the next lecture.

How to think about the type of PM you want to be

- Hey everyone, so we know now that there's a few different types of product management roles, and each one of these sometimes differs in the types of work that they do. And so it makes sense that different personality types might gravitate towards, say, these different product management roles. Let's talk a little bit about that right now. First, the internal product manager. Due to the fact that you're building tools and products for people

within your own company and your stakeholder's all internal, this type of role is great for people that might be starting out in the role of product management. This role also allows you to learn a lot about technology because usually you'll be making tools that integrate with other internal systems or third party tools like marketing software, CRMs, that sort of thing. Any internal product management role, you'll be doing a lot of project management and you'll have a lot less risk of losing the company big money if you make a mistake or a feature is buggy. This is because you're not building for millions, you're building only for the people that you work with, people in marketing and sales and these types of departments. Now how about the business to business or software as a service project manager? Just like the internal product manager, this is also a good introduction role to the role of product management. Part of this is because you usually have a much, much smaller number of users to build for, not as few as the internal tools role, but much less than the consumer product manager role. This is because business to business companies are usually selling pretty expensive software to only several hundreds or thousands of companies and not millions or tens of millions of users around the world. This type of product role lets you be a little bit more flexible and creative in how you build things, but not necessarily as flexible on what you're building, because often times priorities are weighted heavily by the sales team and how much money a certain feature will bring in. You'll also probably be building on tighter deadlines because sales deals often have deadlines associated with them. Both this role and the internal product manager role have a higher likelihood of you being responsible for a product or a feature on one or very few platforms, which is unlike the next role. So our third one, the B to C, or business to consumer product management role. This is a role that is good for people who don't mind uncertainty and a lot of pressure. It is one of the most challenging product roles in the product management world because you're often building for millions of users and usually have the product on multiple platforms, like web, IOS, and Android. You can also lose the company a lot of money if something breaks, so if you and your team release a feature that is full of bugs and it causes tens of thousands of users to lose time on the app or not be able to use it until you update it in the next release cycle, that is some big problems. On top of

these pressures, you have to do extensive user testing, you have to continually look over feedback from users and it's never quite certain what you should build next. You have to iterate and test in a very agile way to get the features that you need to build. Like I said, the business to consumer product role is not an easy one, but then again, it is one where you will learn a lot. And because you're building on multiple platforms like I mentioned, you're going to learn about all sorts of stuff from web development all the way to mobile development. All right everyone, I'll see you in the next lecture.

Product vs. Project management

- Hey everyone, welcome back. So you might be wondering to yourself, what is the difference between product and project management? Believe it or not, there are a lot of people out there that think these things are one and the same, in fact, they are not, the roles that is the job titles of product manager and project manager are very, very different. However, Project management as a skill is definitely something you will still need as a product manager. So let's dive right in and talk about the differences between these two. We already know that product managers are responsible for the success of a product, but are they accountable? And what does success mean? Well, success is defined by KPIs or metrics. A product manager is responsible for being successful by reaching certain metrics, and don't worry too much about that right now, we're going to have a whole section on metrics later. So anyways PM's have a metrics goal that means success and they need to reach that goal and the method by which they reach that goal is not defined, it is totally up to them and their team to decide how they want to achieve that goal. I'll give you an example here, let's say I run an eCommerce company and I hire you as a product manager, I tell you that I want to increase checkout conversion percent by 10% this quarter, you can go off and do user research, look at all the data and any of a number of other things that you might want to do to decide how you're going to approach solving that problem or reaching that goal, okay. So you might decide to redesign the checkout process because users don't understand it or maybe decide to put in faster servers because the site is a little bit slow and you find out that it

being so slow drops past the conversion percentage because so many people leave before they check out, no matter which of these approaches you pick you succeed because you get that 10%. However, you, as you probably already know, one of these approaches is going to be better than the other, that all depends on the upfront research and user testing that you do with you and your designer and the rest of your team, Okay? So how about a project manager? Well, a project manager is different because they are simply responsible for accomplishing a project, not a goal, and a project usually has a timeline and a budget as a constraint, project managers are extremely important for rigid things, like let's say, building a skyscraper, if I'm the CEO of a construction company and I hired a project manager to build a skyscraper I'll want to hire a very good project manager, to get the job done on time on budget, and I want to know that that person knows how to best react to things like weather delays or supply shortages or that sort of thing. What I would not want in this instance is someone that's acting like a product manager and saying that I want to build a building that's a hundred storeys tall and holds 10,000 people and the product manager can kind of do it. However, they feel like they're going to accomplish that goal because again, product managers and their teams are doing a lot of experimenting to see what will work best, and you definitely don't want to be experimenting when you're building a skyscraper, you want to know for sure that the way you plan to do it is going to work. Now, like I said before, as a product manager, you definitely need to know the skill of project management because on the front end, you are collecting requirements and ideas and defining potential future initiatives, and then you're actually putting those things into development, you're putting them into exit. You go into execution mode with your team, at that point, you do need to know how to project manage things, at least in a software sense. Don't worry though later in the course, we talk about project management and we have several lectures that go through the various components of project management as a product manager, working in a software world. Now, once again I want to say is that project managers aren't totally useless in software, and in fact, there's a lot of things that they're useful for outside of buildings, skyscrapers, and bridges, and that sort of thing. Can you think of any types of software engineering projects where project managers are completely

necessary? How about when two tech companies combine together and they want to merge their technology platforms, that's already kind of known how this is going to work, how they're going to merge together. Otherwise the companies probably wouldn't have merged in the first place, but there are so many moving pieces here that someone needs to be checking in constantly and making sure that things are happening on time. Here's another example where project management is a little more prominent and maybe useful in product management. Take for instance, an agency like a software consulting agency or an ad agency or something clients will come up to them and say, we want you to make a campaign, or we want you to make a particular piece of software and kind of everything is defined on what needs to be built. So then it becomes, again, that execution mode where the project manager really just needs to make sure everything is executed on budget on time, and there's not a lot of experimenting to try to do things differently or reach a particular goal, okay? So that wasn't so hard, was it, there's a really big difference between product management and project management. And we know that there is some project management inside the role of product management, but as far as careers go and having those titles as a profession, they're two completely separate things. Brief recap, product managers are responsible for accomplishing a goal by any means. They feel will be best after they consult and collaborate with their team and do user research and all sorts of stuff like that, and project managers are responsible for accomplishing a set of things, which is usually do these requirements on this project, either in a physical world, like a bridge or a building or on big pieces of software and then do it within this timeframe and on this budget, okay. So I will see you guys in the next lecture.

A day in the life

- Hey everyone, welcome back. I wanted to give you a really quick taste of what the day to day activities are for product manager. So we're going to go hour by hour through my typical day at the office. So I get to the office at about 9:00 a.m. And the first thing to do is check emails and technology news. Keeping up with the industry is really important as a product

manager. About an hour later at 10:00 a.m., I go look at our team's metrics dashboard. And I really look for any weird behavior against our release logs. So if something's going weird in the metrics in a direction we don't think is optimal, then what we do is we just look at what happened with the releases lately and try to track down the reason that that metric is being affected. At this point, I'll also go and check out customer feedback from emails to support our community operations team. I'll look at our app store reviews to see what people are saying about a recent release we've made and I'll even go hang out in Beta forum for a bit to see what sorts of issues people are having, or if they have any feedback for us. About an hour later at 11:00 a.m., we have our team standup meeting. So, our team gets together and we look at our sprint board and our backlog. And then each of us says what we're working on at the time, what we recently completed, and if there's any blockers, or if we need help with anything. I also try to make it a point to tell our team the engineers and designers that work with me, what metrics look like lately you know, that morning if they're looking good or bad, if there's any interesting things people are saying in terms of the product feedback, this is especially if we've had a recent release from our team, and everyone's excited to see you know, how the users are liking it. So the standup goes until about 11:15 a.m., because it's only a 10, 15-minute meeting. After that I've talked with the team designer about our user problems and the latest feedback. We may even go to the whiteboard to sketch out some solutions that we might be thinking about, or just work on things we've already been brainstorming over the last week or a few days. So at about noon, 12 o'clock. If we're planning any upcoming features, I'll take some time to write specs, tickets and user stories, and insert requirements or wireframes, from the designer into those tickets. At 1:00 p.m., depending on the day, they usually take a little bit of time to test the latest mobile release for our feature. This is known as the alpha release. This comes before beta, and we'll talk more about this a little bit later in the course. So I test this on Android and iOS on these two test devices I have. And then our designer helps with this as well. And both he and I report our bugs and issues that we found during testing into a new ticket inside of JIRA, our product management tool. So from 2:00 p.m., and onwards the stuff I do changes drastically from day to day. So here's a few things I may do depending on the day and what's

going on in the rest of the company. If it's a Monday, then I meet with the team to do sprint planning or sprint grooming or a retrospective on the last sprint. On most days, it's likely that I'll attend meetings with other product managers or stakeholders to work on cross team planning, or get a feature cleared from other stakeholders. Example of this would be to meet with a sales team and tell them that something is changing. Some other random frequent happenings that are sprinkled throughout the week or things like meeting with our data science team to discuss the recent tests that we ran and look at exploratory data from user behavior to help my designer and myself come up with the right solutions for the next iteration. I also might attend some user tests of our new features or present metrics, milestones to executives, investors or other product managers. Sometimes we work with a team to present a recent accomplishments to that company, we do this on Fridays. So the company everyone gets to demo kind of what they've done. Our team will also hold open houses to ensure that everyone in the company has the ability to talk to our team and align around any questions or feedback that they have for us. I also meet a lot with new employees on product vision and direction to orient them. This is really important to make sure that all the new employees coming in understand what each component of the product does and which teams are responsible for what. So as you can see, there's really not a typical day as a product manager and it's never dull. The days are always dynamic, but some things like checking industry news and your metrics and meeting with your own team never change. I'll see you all in the next video.

Why product management is awesome

- Hey guys, welcome back I'm going to talk to you a little bit about why product management is such an awesome thing to learn and why you should be so excited about taking the course. Product management is easily one of the cores in most firm positions you can have in the entire company or really in the other company for that matter. Why? Product managers are a communication hub and jacks of all trades. It's their job to know about a lot of things and sometimes a huge amount about very certain areas depending on

what you're working on. Product managers also get to directly affect every single part of the business and they interact with every key person on every team from marketing, to sales, to legal and engineering, and even design. All the way up to executive and often times they're in touch with all the board members. To top it all, product managers have incredible career potential and even get paid the same as engineers in the tech industry. The topic of product management is a very big one. It's not new as it can drive the other subjects. This is a long course simply because the topic is a big one, and I want to make sure that you understand as much as possible. After you finish this course, you have a very solid understanding of product management as a whole the different types of product management, the specifics of product management the day to day activities of a product manager, and the internal metrics different technologies, how to communicate with the various people around the company, including engineers, designers and executives and we even help you out with your job search, your resume' and the interview if you're looking to get your first job as a product manager. By the end of the course whether you're an entrepreneur and aspiring product manager or a current product manager that is simply looking to get more skills you have the tools you need to push your career to the next level. In the next lecture we're going to dive right into defining product management and get on our way.

Hooray for free stuff

(upbeat music) - Hey guys, welcome back. This is not really a real lecture. We just want to talk about all the free stuff that is coming to you from this course, we really worked hard to put together things that would be really valuable to the students that want to become product managers, just want to learn this discipline. We're excited about two main ones, but check out the resource section, whole list of stuff, softwares, free trials, extended trials, resource guides, cheat sheets. We have a lot of stuff in there. Check it out. But the two ones that I want to talk about that you guys can start using immediately are-- it's the Slack chat. - Yeah. So the Slack chat is basically we created a Slack room where, or a Slack group, where you can plug in and you can get into a group of people that have taken this course, or that are actively

taking this course, including myself and Evan. So this is a real life, on the fly sort of chat where you can come in, share resources, things you've read, videos you've watched, ask questions, have other people answer them. And you know, Evan and myself will also be in there answering questions and passing along resources. - What I really like about the Slack chat is you guys can talk to each other. You can direct message each other, but also we've separated it into channels. So we have channels we're talking about, just PM in general. Then we have sections where we're talking about looking for a job and then we have another channel for interviewing with a job and then getting a job. So you guys can all hang out depending on what stage you're at in your PM process. Alright, so check out the resources, guys. Lots of cool stuff in there. Post any comments if you have any confusion or anything like that. Otherwise, see you in the next lecture and we're getting started soon. - All right, exciting. We'll see you guys in the next area.

The four major phases of the product lifecycle

- Hey, what's up. So before we jump into product development it's important to understand the market stages that products and companies go through on a macro level zoomed out level. First, this is what we call the product life cycle. Once a company has started, or once they have developed a new product there are four stages that product or that company go through. The four are introduction, growth, maturity, and then decline. Products and companies progress through these stages of development and the way that you know which stage they're in is how much revenue they're making over time. Also some other things like how fast the revenue is growing and what their monthly recurring revenue is. So let's talk about each one of these four phases really quickly. The introduction phase first of all. This is when a company first introduces a product to the market. Usually the product will have little to no competition in the market. At this phase of the life cycle the business typically loses money on the item and the only people that are buying it like immediately are the early adopter types. The second phase is the growth phase. And at this point the product has been accepted by the marketplace and sales start to rise. Usually again starting from the tail end of

those early adopter users. The organization may choose to create improvements to the item to stay competitive and at this point there are usually still few challengers out there in the marketplace trying to compete with them. So then we have the third phase maturity. In the maturity stage of the product life cycle sales will reach their pinnacle. The other competitors enter the market with alternative solutions at this point because it's so attractive and they start making a lot of competition in the marketplace it gets pretty crazy. The company that introduced this new product may start to find it pretty difficult to compete with everyone else in the industry. Lastly, we have the decline phase. And at this phase of the product life cycle, as merchandise and products and the company just reach its saturation point that sales begin to diminish. Remember we have all of that competition out there. Most products are phased out from the marketplace right about this time because of the drop in sales and all the competitive pressure. The marketplace sees this item and says, "Okay, this is no longer new and great "it's a little bit old, not very relevant anymore." Now it's important to know that depending on the product or the industry, this stuff might shift a little bit. It's also really important to mention that there's no definitive guide to telling where a company or product is within these phases. It's really just a way of thinking a framework so we can understand how companies and products begin and end. So we've named these four phases you've got introduction, growth, maturity, and decline. Can you think of some real life companies in each one of these phases. We're going to take a look at some of these real life examples in the next lecture.

Product lifecycle phases: Real-world examples

- Hey everyone. So I wanted to give you a couple quick real life examples of products that are in the product life cycle and a couple of the different stages. So remember we have introduction, growth, maturity, and decline. So let's go ahead and take a look at my web browser here and we'll look at a couple of this products. So this would be a product that is in the introduction phase. And that is because it is really, really early in the market it's something I found the other day called the Dream by Rhythm it says website

rhythm.co. It's basically some kind of a sleep wearable that helps you get better sleep but the real telltale here is that it's not quite launched to the absolute public but it is available to be bought by early adopters so if you click apply now they've got this first version coming out here very soon. So this is something that is in the introductory phase, and there's really no doubt about it. There's a lot of ways to find companies like this you can go on places like TechCrunch or Product Hunt and check out companies or products that are just now starting. So now let's check out a product that is in the growth phase. Right, so my example for a growth product is Snapchat. And most people know what Snapchat is but if you don't it's a little bit of like a picture based social network that's been growing pretty quickly it's only a couple of years old but it's still in growth and this is their website which doesn't show you much but you can see the app there. But real telltale here is if you go to Google trends. So let's type in Snapchat and to Google trends and we can see over time the interest is just skyrocketing and it's really showing no signs of slowing down and they're beginning to make money so they're really, really in this growth phase and there's a lot of people that are starting to look to compete with them at this point. All right so let's take a look at a product that is in the maturity phase. How about Twitter? This is something we've talked about a little bit in terms of the company but Twitter is having some struggles with their growth. A lot of people think it's in the mature phase because their product is becoming a little bit more difficult to use for the mainstream audience and if we just Google some stats on them we can get articles like this saying that their growth over time is really stagnating this graph right here shows that we're getting a little bit of a plateau here 16, 16, 18, 18, and 19 is not really growing like it did back in the day from eight to twelve and ten to thirteen. So Twitter is definitely a company that is in the maturity phase unless they can do something really amazing to their product, it's probably going to stay that way and maybe even go into decline. So let's take a look at example for something that's in the decline phase. How about Yahoo? Yahoo is a really old company actually and they've been in decline for a while but it's getting really bad lately. If you've looked at some of the articles about them lately, you can see ones like this Hedge Funds are dumping their stock with like just a simple Google search or go to Google finance and you get stuff like this where we can back off the last five years and check out their revenues

that are stock price and it's definitely on the way down there's even talks of them selling a piece of their business at this point. So just using Google and looking up products or companies is a really great way to kind of feel out where they are in the product life cycle. Like I said before, there's really no definitive guide to where a company is in particular right in the cycle but you need to understand really how companies go through these cycles and they try to stop from the declining. All right makes sense. So we've got those four major phases of the cycle and you can just do some internet research and get a really good idea of where companies are at that point. I'll see you in the next lecture.

Product development process

- Hey there, so now, we understand how larger products and companies start, grow and then eventually mature and decline. Let's talk about products themselves on a smaller scale. Every product that is being developed has a series of steps that they go through. We call this the Product Development Process. Sometimes you'll hear it be called the Product Development Process Lifecycle, or Product Development Lifecycle, but it's the process that products go through when they're being developed. So there are seven main phases to this Product Development Process. They are Conceive, Plan, Develop, Iterate, Launch, then Steady State and then finally we either Maintain or Kill the product. We're going to talk about the first four in this lecture and we'll talk about the rest in the next lecture. So the first phase, conceive. This is the phase where we collect user problems. We brainstorm solutions, we're saying, all right we want to make a product, but we want to figure out what we should make. We identify the focusers we want to work on. The biggest source of ideas for this phase is usually inside of a company from the employees of that company, themselves. Phase number two, Plan. In this phase, we do market research. So we've already got our ideas and we do the research on those ideas. We look at the business case. Is this something that's going to make us money? We do customer interviews and see if people find this to be a problem. We also roadmap out what we think we could make, how long it might take and at what points we

should have particular features if we're actually going to do this. So we move on to the next phase. The third phase is develop. This is where we get our hands dirty. So we want to make timelines and we want to write out features that we're going to have in this product. We're going to make our user stories and our specs. And we're going to talk to development about estimations on very specific details of how long will this particular thing need to take to be developed. By the time we begin developing the requirements should be set. And they really shouldn't change until we get some sort of validation on the early parts of the completed product, or MVP. Which we'll talk about a little bit later. Then we move on to the fourth phase, Iterate. So this is when we finish the MVP, or very early prototype or the early minimum viable product of this new product we're developing. We get early feedback from users. We test the assumptions that we made with the original idea for the product. We don't wait until we are all the way done because we want to get the testing in as early as possible from the users. So we use tools like, having an alpha and a beta. You've probably seen these sorts of things out there where you can join beta programs for software. This is because these things are not complete and they want to make sure they're heading in the right direction before they do a full launch. Again, those first four phases, Conceive, Plan, Develop, Iterate. And we're going to talk about the next three major phases of the product development process in the next lecture. I will see ya there.

Getting deeper into the product development process

- Hi, so in the last lecture we discussed the first four phases of the product development process. The company conceives of the new product. They plan it out. They develop it and then they iterate on it before you even launch it. And you do this using the early test feedback or the alphas and the betas and there's a few other things we can do to get that feedback. But after that's done we want to do three more things. We want to launch, then we want to hold it in a steady state and then after that we decide if we want to maintain it or we kill the product. So let's talk about launch. - [Announcer] Lift off, we have a lift off. - We want to work with the marketing team and the legal

team, the PR team, the sales team. And we want to position this product for public launch. Then we actually send it out there to the public. We see what kind of reaction we get. Then the sixth major phase is called steady state or just basically hold it in either a steady state or continue iterating on it. During this time we're collecting metrics on how people are using it. How often are people buying it. We analyze the product and we optimize our metrics. We try to get the maximum return for our investment on it. During this time we also have marketing continuing to market it and sales people continuing to sell it. We assess our continued efforts here to basically see how likely are we to continue moving forward with this. Then we move on to the seventh and final phase of the product development process, Maintain or kill. At this point we decide, using all the data we've already gathered and especially from the last phase. How frequently are people purchasing this? Are we staying on top of the market? Are we competitive with this product anymore? And we look at, more importantly, how much money are we spending to maintain it? To see, is it even paying for itself? How's our return on investment? If it's not doing so well then maybe we decide to kill it and we decide to move on to something new. One thing that's important to mention here is that it may not have anything to do with the revenue of that product. Or maybe even the usage it could just not fit our company vision anymore. Our executives and the people that are leading the company might decide, just based on market circumstance, that the company is now going in a different direction. So we need to go ahead and cut our losses on that product. Or go ahead and put it to bed. If we decide to kill the product then we do what's called sunsetting. You've probably heard this before. This is a slow transition to the end of life of the product. We message our user base. We tell them what's going on and we have an end-of-life plan for the users and the people that may rely on it. So if we have a company or product that is storing data for people for instance, then we may allow them to back up their data before we shut down the product. Everyone's probably seen a product that they love to use be killed and it's always very sad. But unfortunately sometimes there's not much you can do about it. But I have had a lot of good experiences with products that are dying and being killed out but they still allow you to bring out your data and something like that. This is really good practice. So those are the seven main phases: conceive; plan; develop; iterate; launch; steady

state and then finally we maintain it or kill it. One thing to mention here is that this might sound a little bit academic to you it is because it is actually pretty academic. This is a process that every product goes through. But you're not going to be actually using these terms in the office everyday. It's just very important as a product manager to understand that every type of product or company goes through these types of phases and there's things to consider about each. So without any further ado let's go into the next section. I will see you there.

What is Lean Product Development?

(upbeat music) - So what is Lean? Well Lean is a product development philosophy that revolves around cutting out all of the unnecessary work or effort, until you're absolutely sure you need to do it for your users or customers. Let's say I have an idea for a food delivery business where someone can just text on their phone, their order to a cell phone number, and we bring him the food, like that. What would you need to make this business a reality? Seriously, put yourself in the position of someone that's going to be starting this business and think about what you would need to get it to be a real thing. Normally, we'd incorporate we'd hire some drivers, we would build a website, we would put out some advertisements, and we would buy a cell phone number for people to text for the food. So to better use resources here, I could have done some work up front myself by doing things like having people text my own cell phone number or putting out flyers myself, or driving the food to people's houses myself, once I got a text message. This would require a lot more work from me upfront, but it guarantees that I'm not spending money on stuff until I know I actually have to. So at the point that I get enough business where I can no longer do these things myself, or I can no longer drive to the people's houses myself, or take the orders myself, then I start hiring the drivers. This is a Lean approach. So Lean is really just about building something in an intelligent way, where I'm not using the resources until I know I actually have to. And this way we save a ton of money. Lean is not specifically for software though. There is a process called agile that is specifically for software. But I do recommend reading more about

Lean, and if you'd like, there's a whole book on the topic called "Lean Startup" by Eric Ries, and it looks a lot like that. So in the next lecture, we're going to talk about agile.

What is Agile?

- So we know that lean is a way of building a product or business without wasting resources, without spending the money until we have to. But what is agile? Aren't they the same thing? Well, a little bit. In this lecture, we're going to talk about agile, and why it's so important for a product manager to understand. So what is agile? Well, it's pretty easy. Agile is just a way of applying that lean mindset to software development. And agile as you know, is an adjective. You can say something is agile, but when people, especially in the software or tech industry say the word agile, they're usually meeting agile software development, which is just a sort of a project management framework for software development. Agile is an iterative method to developing software, where we group things into small batches and do them in order to not waste resources. So let's imagine that you are writing down the features that you think an app should have. So you're making an app, write 10 features down, and you think these are absolutely necessary for the users to want your app. If we're doing this in an agile way, we would probably only research and develop the first two-to-five features that we think are most important. So we would take those features, we say, "These are the most important ones," we design them, we research them, we develop them, we release them, and we see what the user feedback is on those three-to-five features, before we go on and develop the rest of the features. In the next lecture, we're going to talk about two very specific frameworks inside of the agile development process called kanban and scrum. All right, see you in next lecture.

What is Scrum and how does it work?

- Welcome back, everyone. In this lecture, we're going to talk about scrum and kanban, which are two agile development philosophies or frameworks. So, we know now that agile is a philosophy for developing software in a lean and iterative way with plenty of collaboration between the team members. But, what are the guidelines for putting it in place? How does it actually look? There are two very popular agile software development methodologies or structures. One is called scrum, which you've probably heard of before and the other is called kanban. Some people pronounce it can-ban, but whatever. You can think of both of these methods as merely guides to developing software in an agile way. So, let's start with scrum. It's probably the most common of the two. Let's walk through how to implement it or what it looks like in four easy steps. Now, there are a lot of things about scrum, but these are the four main ones. Step one, the sprint planning meeting. So, you start off with a product backlog, which is a big, prioritized list of all the things you've either decided to work on or may work on. Your team holds a sprint planning meeting, which is where you take the most important feature or several from the top of your product backlog and you move it to your sprint backlog. Then you talk about all the work you're going to need to do in order to implement it. Together, your team writes out what work needs to be done in order to make that feature a reality. You put the work into a project management software into things called tickets. Step two, this is where you start developing. The core tenet of scrum is that the work your team does is boxed into a timeframe called a sprint, usually two weeks. Some companies do three week sprints and some do four, but two is the most common. Throughout the two weeks, your team works on the things by taking 'em off the top of the sprint backlog and moving them to in progress and then eventually to done. These are the tickets that they're moving from column to column. At the end of the two weeks, you should have completed everything in the sprint backlog. If not, those things go to the next sprint. So, step three is standup meetings and these are very common and very helpful. Standup meetings are these small, daily meetings, usually hold in the mornings. Why are they call standup meetings? Well, the theory here is that if you remain standing as a team and you're not sitting down, then the meeting will be very brief and just concise in general. And we find that to be pretty true. So, the goal of the standup meeting is for every team member to

talk about what they worked on last, what they're working on now, and then what they're going to work on next. Most importantly, they should bring up any questions or requests for help during the standup meeting. These daily meetings, usually every morning again, keep everything proving really smoothly. So, step four and another big component of implementing scrum is retrospective meetings. A retrospective is kind of the opposite of a sprint planning meeting. The retrospective is a meeting you have at the end of each sprint with your team. You get everyone in a room, yourself, the developers, the designers, et cetera, and you talk about three main things about the last sprint. What went well, what did not go well, and what questions people have. It's usually your job as the product manager to lead the team through this. These meetings ensure that everyone that has questions or concerns about the team or the structure or your process is getting these things heard and it keeps everything moving along very smoothly. All right, so let's recap on scrum and the big four major components. The first one is that we have a sprint planning meeting where we take everything from the product backlog, we put it in the sprint backlog. Secondly, we have the sprint itself, which is a time box period of time like two weeks, three weeks, or four, but the most common is two. Then we have the daily standup meetings during the sprint. These are 10 or 15 minute meetings where we keep standing up so that people don't take too long and we just talk about what you're doing yesterday, what you're doing today, and what you're doing tomorrow and clarify any issues or any collaboration needed. Finally, we have the retrospective meeting where we talk about what went well during the sprint, what did not go well, and any outstanding questions that people have. All right, everyone, we'll see you in the next lecture.

What is Kanban and how does it work?

- What is up everybody? So what's Kanban? And how does it compare to Scrum? By the way, is it Kenban or Kanban? I'm almost positive it's Kanban but if I'm wrong, tweet at me or something. So like scrum, Kanban is just a framework for implementing agile software development, except it's not as strict as Scrum in terms of meetings and times. So you may have actually

seen, what's referred to as a Kanban board before. It's just a bunch of columns with cards on it that you can move from one column to the other one to reflect a state of that item. Like TO DO in progress, and then Done. However, just because something has a Kanban style board does not mean that that is the Kanban process. Instead, the key concepts with Kanban are that it does not use sprints. So you don't timebox the work of your team into these two or four week periods. Also since there is no sprints, there's no sprint backlog, which we'll learn about it, but only the product backlog itself. So what happens is the team just works on their tickets. They do the work, they move it to done, and then they take the next task off the top of the product backlog. And that backlog just goes on forever. It's just endless. Kanban is also different from Scrum in that it does not prescribe any particular meeting types like scrum does with standup meetings and sprint planning meetings and retrospective meetings. And there's one more big difference. Kanban operates off of the theory that only a certain amount of items can be in progress or in any given state at one time. How many items can be in each particular state is up for you and your team to decide. Some software teams use Kanban, but it tends to work best with teams that are not very concerned with things like estimation and are continuously moving tasks through the same number of steps pretty quickly until they're complete. For instance, customer service teams often use a Kanban style of working, right? So we are familiar with Scrum and Kanban. And then the big question is, which one is best? Sorry to disappoint you, but there's no definitive answer here, as it depends on what your team prefers to use. The best process is the one that you like using. As you can tell, Kanban is a more relaxed way of doing agile development, but it has a big disadvantage since it doesn't use sprints, it's going to be more difficult to project the amount of time it's going to take to complete work. Don't worry too much about that right now. 'Cause we're going to talk about things like estimations and velocity at a later lecture. So let's recap. Kanban is just another way to implement an agile type of software development workflow. It's a little less prescriptive than Scrum and it does not use sprints. All right, everybody, I will see it in the next lecture.

What is Waterfall development?

- In the Agile app example, we took those 10 features we thought we wanted, and we only selected a few that we thought were the most important, and we took those and we researched them, designed them, developed them, and then released them, and validated whether or not those are the things that users were interested in. Waterfall looks a little bit more something like this. We take all 10 of those features that we think we want, and we develop them all at the same time, so we research, design, develop and release every single one of them all at once. Doing things in a Waterfall way to sometimes pretty risky, because your software product might have a ton of features, and when you have a ton of features, it's going to take a long time to develop them all at once. During that time, your competitors may do something, or you may find something else about the user needs or market that makes it much harder to adapt. You also might find that after you've developed it, your users wanted something else that you've spent little time on, and you spent the most time on stuff that they're not actually going to be using. So after all of this trash talking on Waterfall development, you might be thinking when is it ever useful? Well, there's actually a few specific use cases where Waterfall development is pretty useful, and we're going to talk about those in the next lecture.

Real-world examples of Waterfall and Agile

- In the last lecture we learned that Agile software development is a methodology for applying that lean mindset to software development inside a company. So what does it look like when we actually apply it inside a development environment? What does it look like day to day? Let's say we work at a music streaming company and we've identified that the most important thing to our users is to have a ability to build a playlist. So using the Agile development methodology, we can get everyone in a room, the engineers, the designers, and us, the product managers. And we can decide on the very core features of building a playlist and what that entails. So if we talked to the designers and engineers, we can be pretty certain that the most

basic requirement for building playlist functionality is going to be adding a bunch of tracks or songs to a list called a playlist. With this knowledge, the engineers can go ahead and start developing the database that we know we'll need to build the playlist functionality. In the meantime, the designer and us, the product managers can get together and start talking about how will the playlist act and how will they look and how will they feel inside of the app? So us, the product managers and the designers can go off and wireframe some potential designs, the designers can make 'em look really nice. We can test them with users and we can make changes based on user feedback. We can do this process a couple of times while the engineers are working on the database. By the time we've got the perfect design and look and feel ready, the engineers can put it inside of the app because they're done with the database. So now with the engineers done with the backend and the database, they start implementing the actual playlist functionality in the app. Then, the product manager and the designer can go on and start talking to users about the next features they should build and then start doing the entire process over. So all the engineers are actually implementing this feature into the app, the product manager, us and the designers can go talk to other users and look at metrics to decide what feature we should build next. Then we go and do wireframes again and design, and we get some more feedback from users and we can just complete the cycle over again. When companies use the Agile development methodology, usually everyone is in very close communication and collaboration. Engineers, designers, and product managers are usually in the same room or office and they're always talking. So how about waterfall? When is it useful, and what are some examples of it being a useful process? Well, think about Microsoft Windows or Mac OSX, which are operating systems for your computers. These are operating systems that have literally thousands of features. It would be really, really hard to pick a couple of features that stand on their own. All of these features have dependencies on others and you couldn't really just release a few to the public. Another example of waterfall being a little bit more useful than Agile is in mission critical software applications like breaking or engine control systems in cars. Do you think that using Agile development and developing only a few few features at a time inside of a car for the braking systems is a good idea? Definitely

not. Well, what about things side of software? Waterfalls used in other places where it's pretty useful. Let's imagine we're building a skyscraper. Do you think we'd want to build five floors at a time and then put a bunch of people in it and see if it still stands? Probably not. We're going to build all 100 floors together and we're going to do it from a spec that we know from the beginning has everything we need. When the waterfall software development process is used, the team does not collaborate quite as closely as the team, that's doing an Agile software development process. You can have the research and product requirements team in one city, the designers in another city and time zone, and the developers in an entirely other city in timezone. This is because everything is defined upfront and it gets passed from one team to the other and nothing changes. So I hope everyone has a pretty decent understanding of the differences between Agile and waterfall and what sorts of real world examples these are used in for your purposes as a product manager, it will be focused mostly on Agile, but don't think that waterfall completely useless because it's not.

Introduction to ideas and user needs

- What's up, everybody? Everything in life has to start somewhere. No, I'm not going to lecture you on philosophy or something like that. But, when you're product manager products also start somewhere. And that means that features and new functionality... It all comes from some place. Everything that you see or use in your life. Any product. At one point it was either an idea in someone's head or it was a need. Like a requirement. Someone said, "Hey I need this team to build this for me". And then they built it. So there's this theory out there that product managers and executives they all get in a room and they're the idea people that just sit there and come up with these fantastic ideas. And then the developers and the designers you know, make it look pretty and build it. And then the executives get rich. That's actually so far from the truth. I don't even want to go there. You're job as a product manager is not to be the idea person. We're going to talk about that in a little bit. But, it's really just about getting this huge influx of ideas and requirements from all over the place. Requirements are... Other people coming up to you and

saying, "Hey, I need you to build this". Maybe a sales team or marketing team. Hey, we need this functionality. That sort of thing. So you have all of this stuff and your job is really just to say, "Of all of these things which ones are good ideas and which ones are bad". Why are they good ideas and when's the best time to do them given all of the other stuff that's going on at the time. In the company, with technology and in the industry. Anyways, that what this section is all about. So, get ready to take a dive in to the land of ideas and user needs and false user needs. And users thinking they need this thing when they actually need this thing. Anyways, I'm getting ahead of myself. But, this is all coming up. So, I will see you in there.

Where ideas come from as a PM

- What's up everybody, in this lecture we're going to talk about where ideas come from as a product manager. So as a product manager, your entire job revolves around deciding what you need to do next. This requires a list of possible features to build and also a method of deciding which of those is the most important. Now, later on in the course, we're going to talk about prioritization features and epics, which we'll learn about, but for now, we're just going to need to focus on getting that list of features in the first place. So where do ideas come from? Where's the top of the phone for the product manager in terms of what you might build in the future? Well, I want to start by saying that any company you work for as a product manager, is going to have a long list of requests and feature ideas that need to be developed. If this is not the case, then you should feel really lucky. Even if you do have a really big list of stuff that needs to be built. There's always going to be instances where new ideas come in and take precedents over existing ones based on things like user feedback volume, what the competition is doing, et cetera. Alright, so let's go ahead and assume you're a product manager at a new company and there's already a few things your company says you need to build with your team. But you also want to make sure there's plenty of other ideas in the backlog to think about and develop after that. Where do you get these ideas? Despite popular belief, the product management job is not about being the idea guy or idea girl. Ideas and

requests come from everywhere and you need to collect and organize them as a product manager. Feature ideas and requests usually come from four main places that we can easily remember using an acronym that is E-M-U-C, EMUC. So let's go through each one of those. We'll start with E, Employees. This is just ideas coming in from your co-workers, management and you, yourself. The second one is M and it stands for Metrics. This is just problems and inefficiencies you find when you're looking into how users use your product. For example, I may be looking at some data on how users navigate through an app. And I might notice that a lot of people just spend one second on a particular area and immediately go somewhere else. Well, that might give me the idea that this is an unclear section of the app and we need to do some sort of redesign or at least go figure out what's going on, and then try to fix it. Alright, so now let's move on to U, which is Users. This is user feedback from forums or emails to your company or yourself and social media. It's just everyone out there that uses your product. So the last one is C, Clients. And this is a special case that applies primarily to product managers that are business to business, because it's not always the case that a user is the same as a client. A client meaning the person that paid for the product. So depending on what type of product manager you are, the importance and volume of requests may change. Do you remember the three types of product management roles, consumer product manager, internal tools product manager and business to business product manager? Let's do three examples really quickly of the PM types and where the ideas might come from. So if you're an internal product manager, most of the ideas are probably going to be coming from your primary stakeholders, which are your co-workers, employees that you work with. If you're a business consumer product manager, most requests are probably going to come from the users, but you should also look for problems and metrics and listen to new ideas from the low co-workers, especially management. Now, if you're a business to business product manager, it means you're working on a product that's being sold to other companies. In this case, requests and ideas for features are going to come primarily from employees and clients. Like I said, a user is not always the same as a customer or a client, if you're a business to business product manager but we will talk a little bit more about that later. Alright, let's sum it up really

quick. You're going to have ideas and feedback coming in from all sorts of places, depending on the type of product manager you are. Some of the volume of feedback is going to be different from those different groups. But you can remember these groups by the acronym EMUC, Employees, Metrics, Users, and Clients or Customers. Alright everyone, I will see you in the next lecture.

Getting to the real user needs

- Welcome back everyone. So in the last lecture, we talked about all of the places where you can receive user feedback and ideas. Once we have this list of ideas or feature requests, do we just go build them? Let's say that a user has given you some feedback and they want you to build a native mobile app for your website. So you get that feedback and you just go out and build it, right? No, being a Product Manager is about a lot more than just learning a few skills and applying those skills. One of the most important things about being product manager is having a product manager mindset, and a big component of that is being able to understand the core pain points, the real problems behind what people ask you to do. So in this lecture, that's what we're going to learn, how to get that essential skill. Let's start off with an example. Let's say that I have a friend and I don't really have any friends. So this is an unrealistic example, but let's say that I do, and they are standing outside in the rain and they're getting all wet and they're upset about being wet. And they say, "Cole, can you give me a towel?" And I give them a towel and they dry off and they get dry for a few minutes or maybe a few seconds, and it's still raining. So they keep getting wet, did I really solve the problem. Well, of course not. They asked for a towel, so that's what I gave them. That was a request to me. But is that the actual problem? No, the actual problem here is that they're in the rain. So the actual solution would be to get them out of the rain. So I know that example sounds pretty ridiculous, but this sort of thing happens all the time in the world of product management. People are always asking you to build things or add features and those things at first glance seem reasonable, but actually they're merely symptoms of a real problem the user is having behind the

scenes. Being a product manager is all about finding a solution to a problem instead of trying to fit a problem to a solution that you or someone else has thought of in the first place. Given any request or a feature idea, the first thing I want you to do is think to yourself, is this solving an actual problem? The second thing I want you to do is ask yourself if we do this exactly as they ask without looking at any deeper, is it going to have any unintended side effects? I'll give you an example of an unintended side effect. Everyone's familiar with Twitter, right? They've got several hundred million users. The way Twitter works, is you can type some characters and send it out as a tweet, and that appears on your profile and it also appears in a newsfeed for everyone that follows you. On Twitter, you can also retweet, which is clicking a button on their tweet and it puts it on your profile. And all of that also feeds into the newsfeed. So if you are following a ton of people, then your newsfeed can understandably get pretty busy. One of the most common things people ask for on Twitter is the ability to just filter out retweets. So the idea is that if the product managers and developers at Twitter just made a button to where you wouldn't see retweets on your timeline anymore, it would make things a lot less messy. It sounds pretty reasonable, right? Now, Twitter as a business model is a company that really just tries to show everyone out there what is really popular right now. So what are people around the world talking about? Think about retweets for a moment. When I retweet something it's like saying, I really like this, I endorse it. I like it so much that I'm willing to put it on my own profile. So if we built this feature and it got pretty popular, then we could have say 20 to 40% of people not seeing retweets at all. What ends up happening here is less and less people retweet because there's not as large of an audience and then Twitter has less of an idea of what people really strongly like or really strongly endorse. This makes it even more difficult for them to separate out signals on what's globally super popular. This affects normal users as well. So with less people retweeting, it's more difficult for them to tell what is super popular or what people really, really like. So remember the request here was to allow us to filter out retweets on the Twitter timeline. And we've just covered, that might not be the best idea because it has some unintended consequences. I'll give you an example from my career of a feature requests that did not actually address the user problem. A while ago, I was a product manager at an eCommerce

website and we were working on building a new content management system for the marketing team because they had outgrown the old one. This is basically a system by which they can go and edit the photos and the text and that sort of thing on the website while the website is live. So I'm meeting with the marketing team and I'm writing down requirements and they're saying, we need it to do these things. One of the things they mentioned was they said they needed the ability to put up a small message on a specific product web page to tell the user something. And they wanted to be able to put that up and take it down pretty quickly. At that point, I could have said, "Okay, I'll take that into account. "And I'll make sure that whatever we do, "we'll give you that feature." But instead, what I did was ask them, what's behind this request. What do they actually want it to do? So I asked why this was actually a requirement and for what reason do they need this? So it turns out that the merchandising department told the marketing department to tell me that this was a requirement. And the reason was, it turns out, that they would hold sales on particular product pages. And on this shorts page, in particular, the sizing and pictures were a little bit misleading. So when a lot of traffic went there, it ends up that a lot of people left because they were confused. So what they wanted was a little, basically a message where users would know to call our customer service line to better understand it. So after hearing all of this, it was pretty clear to me that the real problem wasn't that they needed an ability to put up a message really quickly for customers, the real problem was that the page was extremely misleading and causing problems in the first place. So after hearing this, I went back to the designers and developers and told them about this. And we went to the drawing board and talked about how we might make this a lot clearer for customers. Once we had solved this on the webpage, it was not really a problem anymore for the merchandising department and the customers didn't need to call into customer service anymore. So problem solved, but the real problem was solved, not the initial idea or requirement given to me by another team. I want to show you a really easy way to get to the core issue of a request and see what the actual problem is. And this is a technique that a lot of salespeople use and have for a really long time. So it's really easy. You just ask why, but then you ask it two more times, at least. So if someone asks you to do something, you say, why they'll tell you, you say why again, and they'll tell you

a reason that's a little bit closer to the truth. And then you say why again, and by the end of it, they'll probably have illustrated to you what their real pain point is. It seems really obvious, but in fact, it's not. And there are so many people that just skip over this simple step. Ask why. I once took a sales class back after college. It was really, really good. And this guy that caught the course was in his early 50s. He had done some of the best sales jobs in the world. And his number one piece of advice was to do this. And this technique works with product management just as well. So in this lecture, we learn that the product manager's job is not only to listen to the request the person is giving us, but to find the real pain behind the request, what is the real reason they're asking for this? We also learned that the easiest way to get to that is to continually ask why.

Users vs. Customers

- Welcome back, everyone. So do you remember where I said that sometimes the people that pay for your product are not actually the users of your product? We're going to talk a little bit more about that right now. So there's a difference between users and customers. And if you work as a B2B or business-to-business product manager, this might actually be pretty common. So one of my first product management jobs was at a company that sold social media monitoring software to companies that were interested in identifying trends on social media. So the company was called Mass Relevance, and this product we sold, the ability to log in to a platform and check out all these social media trends was something that cost like 2 to $500,000. So it was not cheap. So the people that bought this were major companies, and the people that signed the checks for this were directors of marketing or social media. And when they bought it, they were not the ones using it but they would give the logins to their underlings, like the social media managers or the marketing managers, and you can see how these are two totally different people. So when we were getting feedback, we would get feedback from two different groups. We'd get macro-level feedback on what our product should be able to do in general, this was from the higher-ups, like the directors of marketing and that sort of thing, and then we had really

granular feedback from the users themselves. These were the people that were logged in every day. And they'd have feedback like oh, where's the button for this? I would like to do this inside the platform. Two separate user groups and two separate sets of feedback. All right, to sum it up, the users and the clients are sometimes two completely different groups with different sets of feedback but it's important to keep in mind. Got it? All right, moving on.

Market research: Sizing the market

- Hey, everyone, welcome back. This section is all about competitive market analysis. A big part of product management is making sure that the market you're attempting to address is large enough to make it worth going into. This applies if you're a product manager at a company and you're looking at building a feature or product that will put you in another market or if you're starting a new business on your own. There are two common approaches to thinking about market sizing. One of them is top down and the other one is bottom up. The top down analysis is calculated by finding the total market and then estimating what your share of the market is. Let's say that we're selling an iPhone app for let's say, ice cream connoisseurs. Top-down estimate would be something like, okay, we know there are 100,000 ice cream connoisseurs with iPhones in the United States, so if we can just get 10% of them to download our app at $5 a piece, we're going to make $50,000. The problem with this approach is that you need to make assumptions on how much market penetration you will have inside that market. Even though it sounds reasonable, it's not the most respected way of estimating the potential market. A bottom up analysis is done by thinking about the current sales of similar products and then estimating how much of those sales you can capture. This is a much more accurate approach because it looks at patterns that exists in sales today rather than the total market. This is a better way to do it as a product manager but it takes a little more effort and takes a little bit longer. Again, let's pretend we have an iPhone app for ice cream connoisseurs. First, we must ask ourselves where people currently buy apps related to ice cream or food. Obviously, this would be the Apple App

Store in this case. Next, we think about the volume of sales for existing similar apps. You may think that there's nothing similar to your app, but in reality, there's probably things at least slightly similar. If there are no other ice cream connoisseur apps, we can look at other apps that have to do with other types of desserts or foods. We do a little bit of online research or we look at websites like App Annie and we look at the sales volumes and popularity for the existing apps out there. So let's say that the similar apps are downloaded about 1,000 times per month globally. At that point we can make a conservative assumption and say that if we get 5% of the sales that they do per month, it comes out to be about 50 downloads per month. So do you see the difference between these two approaches? The top-down approach thinks about the total market size of potential number of users, which is always quite optimistic. The bottom up market looks at the current trends and data to make sure our assumptions stay a little closer to reality. It is much better to be conservative when it comes to starting a product or a business or adding a new feature, than it is to be overly optimistic. Lastly, I want to give you some tips on tools and techniques you can use to look at some market data. One of the first things you can do is just Google, industry reports for X, or industry or market size for X. Just using Google. This is a really good way to find already existing reports that industry research groups have put out. Secondly, if you are looking at websites of competitors, or other similar apps, then you can use a website like compete.com and look at the amount of traffic the different competitive websites get. You can compare this to other ones or other industries and get a really nice competitive overview. Another tool you could use is the Google AdWords keyword tool. This is a tool that Google provides for people to do research on advertisements they want to place with Google. It shows you the volumes and related terms of searches on Google and it is extremely valuable in determining the amount of interest out there for your app or your product. You can also search things like Twitter and Reddit to see what people are saying about things related to your product. This also has the bonus of showing you what frustrations they're having, if any, because people usually will add additional comments or respond and say, "Well, this is what I like or don't like about a certain product."

Introduction to finding competitors

- Hey guys, welcome back to the course. So last lecture we learned about sizing up your market, that's important because you really need to know about the opportunity that you are sizing up, and the relative importance of that opportunity versus your other opportunities. Project management is all about making decisions between opportunities. So that's how big it is, but you want to know something that's even more important than the size of your market? The competitors in your market, and that's what we're going to be talking about in the rest of the section. Now there are a lot of different market circumstances that could happen with regards to competitors. You could be going into a big market with your new product or new feature, and that big market could have lots of competitors. You could also be going into a small market and that could have very few competitors, or it could be the other way around. We need to know how to collect information on those competitors and make an objective judgment about what that means for our new product or feature. Being able to make this type of analysis and decision is crucial to being a product manager and crucial to being an entrepreneur. Now if you're going into a market or growing into a market that has a slew of different competitors that all do things very similar to what you're proposing, some of them do the exact same thing, and some of them do what you're trying to do but better, that's something that you need to know long before you actually engage your development team and your company resources in tackling that opportunity. Similarly, if you have a market that appears to be wide open, no one seems to be doing what you are proposing to do, you also need to be able to make a critical decision. Now remember, that's a weird scenario, there's two sides to that coin actually. Is it that case that no one doing what you're proposing because you're such a genius, you're the first one to come up with the idea, you Steve Jobs you, or is it the case that no one else is doing it because there's no customer demand and they all figured out that it was a bad idea on their own? Now as a product manager, you're going to have to be in charge of feature triage. That basically just means that you're going to need to know which potential opportunity, feature, or project is most likely to, one, get you more users, two, make your users happier, and

three, enhance your brand, and as an entrepreneur who's managing their own product, it's the exact same, you need to know which opportunity is the most likely to get you your returns that you want so that you don't waste a lot of time and resources, and you can't make that call unless you know what your competitors look like, what they're doing, and what they currently offer, so we're going to cover that throughout this entire section. First off, we're going to start with finding the competitors because we can't make a critical judgment unless we know what we're making a judgment on. So see you in the next lecture.

Find competitors as a product manager

- Hey guys, welcome back. Let's get our market intelligence on and start finding some competitors. We cannot carve our beautiful Greek statue of a woman with no arms, always no arms, unless we have a large stone. Terrible metaphor? Yeah, probably. But I think you know what I mean. We first need to channel all of our energy into building a list of potential competitors, and once we have that list, then we can start chipping and carving, and making the critical decisions that we need to make as a product manager. Okay, let's get started. And step one, you probably can't guess it. Actually, step one for this is capture. Now, what do I mean by capture? Well, we need to set up some way of recording the competitors we find. Do that before you even start looking for your first competitor. Whatever you use, Evernote, Google Docs, a piece of paper. You can even talk into Siri and have it record something for you. You just need something set up. For those of you who want to follow along and do everything I do, open up a Google Sheet. That's what I'm going to use. Or an Excel if you want to. And I'm just going to fill column A with the competitors we find. Why do I put them there? You'll find out later. The first thing I'm going to do to this list is I'm going to divide it into two categories, known and unknown. So first one is known. And then I'm going to leave five spaces. I will explain. Unknown. Now, for the top category you can put as many as you want, to be honest, and if you need more than five spots, by all means you can shift everything else down, but do you know what I mean for known? Well, if you're a product manager, presumably you

work within a company, and the company is probably prior established. In that scenario, you're probably going to know who your biggest competitors are. These are the people that you probably grumble about in your sleep, and theoretically talk trash about to your other employees. These are the people that, when you're driving your car, you end up swerving because you get so angry when you hear their commercial on the radio. You should know these, so put these down. This gives you a leg up. Now, people out there who are doing this as an entrepreneur, you're not going to know this, so you're probably not going to have a known category. Now, if you are in a company and you don't know who your known competitors are, number one, don't let your boss know, and number two, try to figure this out as quickly as possible and don't let anyone know that you don't know this. I'll give you a quick tip for finding this in roughly four seconds. Second one and second two go to google.com. Second three and second four, all I want you to do is type the name of your company, and then the word "versus." So if I were using the company GoPro, I would say "gopro versus." There you go. Sony has essentially answered the question for me, or at least given me a headstart. Right in the search suggestions Google tells me that GoPro competes with Sony and Canon, and if I were to search, I could find even more. Looks like Xiaomi, and Contour, and et cetera, et cetera. Great, you got it? Now put your known right there, however many there are, and in the next section, that's where we're going to go out and start digging for the ones we might not be aware of. These are the ones that are kind of hiding in the bushes. And as we collect these, we're just going to be typing these down, and like your therapist, passing no judgment on them whatsoever. We are just collecting them, known and unkown. So how exactly do we search for unkown competitors? Well, the way you do that is you focus on problems. Not your problems, crybaby. I'm talking about the problems of your customers. Now, you probably have figured this out already up until the point in the course, because we've said this over and over and over again, all ideas, products, and features all stem from solving a problem for someone. Even the most frivolous things out there still solve a problem if you sit and think about them long enough. So what you need to do is just stop. Take the product or the feature that you are evaluating and answer the question "What problem does this solve?" And then when you figure that

out, answer the question "For whom?" Now, a pro tip, don't put "my users" in the second section, because that is how you get tunnel vision and you just end up making mistakes. We need to get a lot more specific than that. So let's do it with an example, and we'll use GoPro since they're one of my favorite punching bags. What problem does a GoPro HERO black 4 solve? Well, the problem it solves is probably that people want to film action sequences, or film shots that involve moving objects, cars, paragliders, parachuting, whatnot, and the traditional applications are just not suited for it. You're not going to be carrying a large camcorder or a DSLR in these situations. You need something with a different form factor. So to put simply, it is hard to film action sequences with traditional cameras. And for whom? Well, mostly thrill seekers. Specifically the ones that want to film it, so I'd probably say thrill seekers and/or videographers. Now, once you've identified the problem that it solves and for who, you're basically halfway there. Okay, now that we have a distilled version of what our project or feature does, we know who it's for and what problem it solves, let's use three different techniques with this information to find competitors. Now, there are obviously tons and tons of ways that you can find competitors, but these are three easy ways that will get you a very large amount of coverage. Be very surprised if you had to use anything else. Fortunately for you, it's easy. Just go to google.com. Google's the best search engine. They're the best at finding things online. That's why we use them. Now, strategy number one. I want you to channel the type of user you're building your feature or your product for. This is a user that has the problem you've identified. Now, that user probably complains about the problem. How would they complain about the problem? What specific wordings, or phrases, or context would they use when complaining about it online? Try to channel your annoying sister for just a second and think about what have they been saying before that you've seen, you've recorded, you've observed, and is there any other way that they could articulate this or verbalize the problem that you can think of? Take those and write them down. If I was a thrill seeker that really needed a camera to stick to my head, I'd probably be complaining along the lines of: "DSLR too big for action shots," or something along the lines of: "keep breaking my camera for..." insert the activity, parachuting, or BMX biking, or skateboarding, or all the different things that GoPro people use these for. Try to write down as many

as you can. And once you have the variations down, take those variations and start googling them. What you'll find are three different things. One, companies that are using that verbiage. Companies often do that. They take their customers, or their potential customers, see what they're complaining about, and they'll use the exact same language in their marketing message. So when you search for that exact wording, you might find companies that are targeting them. The second thing you'll find are individual people complaining about it online. Now, that itself is not particularly helpful, but when they're complaining about it, chances are there is a conversation around it or resulting from it, and then you can see how other people reacted to it, and in most cases you'll find how other people solved the problem. You can find potential competitors that way. "Can't find camera small enough for my skateboarding videos." "Oh, man. That sucks. Have you tried the Sony Cyber-shot?" I'm not really into the cameras. And number three is that you'll find what Google seems to be related to that problem. Google's very good at connecting the dots between what people search for and what they typically are looking for or what websites they're going to, so you might serendipitously find competitors because Google made the connections for you. Now, as a pro tip I suggest using this tag in Google. It goes "site colon," and then insert the name of a website you want to search. Now, what this does is it tells Google to only search a specific website. Now, why would we use this in this technique? Well, we're trying to channel customers saying something, so we probably want to find individual people actually complaining about it, and we want to find places where they're probably discussing it. So if you say "site equals," then what you can do is you can insert some well known sites and communities where people often talk about this stuff. So I'll just give you some of the hot ones, like reddit.com is a fantastic way of finding consumer conversations. Quora.com is another one. Believe it or not, Yahoo Answers is a decent resource for finding this stuff. Look through the results. Prune what you can find out of it, and add it to your list. Now, strategy number two. Strategy number two is to literally describe exactly what your product or feature does. Be like a robot and just literally describe it. "Tiny camera fits to your head. Shoots video. Tiny camera designed for action. Compact camera you can attach to your head that films." Things like that. Now, what you'll get, again, are three different things. One, you'll find

companies that use those literal descriptions of their features. Now, they might not use the marketing verbiage or the complaint verbiage that we had in the last strategy, but they might use some of the feature descriptions, very literal, "it does this," and that's a great way of picking up competitors you otherwise wouldn't find. Now, the second thing you're going to find is at the top of your search list there's going to be ads, and those ads are essentially companies who have self-selected themselves to pitch to you because they think that that function you're searching for is related to their product. Those people are paying to be there, therefore probably a good chance they could compete with you. And then number three, what Google, again, thinks about it. Very, very helpful. Now, pro tip for this strategy. One thing I would suggest doing is play around with quotation marks. Now, when you put quotation marks around what you're searching, that does a hard search. So when you do a hard search, Google only looks for exact matches of whatever's in the quotations. The reason why you might want to do that is because if I said "camera that mounts on your head," I don't want that to match up with something that says, "mount a camera on your head," like a suggestion, or "mount a horse and use a camera," things like that. Those are a little bit too off the mark. And then the last one is try to search for how you in your head would pitch this product or feature. If I was pitching the GoPro, maybe I might call it "world's smallest 4K camera." That's a one-sentence pitch. Try to think of some one-sentence pitches that describe what it is that you're working on, and try entering that. Try as much as you can to be specific when you use this strategy. Don't just say, "This app saves you time." A lot of things out there save you time. This isn't going to help you if you're that vague. If your app saves you time by saving your voice notes to Evernote, well then search for it like a pitch on that. Like, "Save time by recording your voice. Save time on the go by using your voice. Write to-do list with your voice. Save time on the go by just speaking," things like that. Okay, got it? Those are three different techniques. Figure out the problem you solve, who you solve it for, and then use those three different strategies to drum up a list of competitors that you think might be worth analyzing. Throw those in the list, and I'll see you in the next lecture.

Direct, indirect, and potential competitors and their impact

- Hey guys, welcome back to the course. So now that we have a list of some dumb competitors, we now need to talk a little bit more about how we can chop this list up and better strategize. Now, there are a lot of different shades of competitors out there. There's the wimpy, not trying very hard competitor. You might have a, let's say, scatterbrained and doing everything competitor, and you might have a big, scary competitor that is looking straight at you. We need to sort these beasts out so we don't have a cluttered mess of threats. Don't be Chicken Little. The sky is not falling. In order to do that effectively and be effective as a product manager, we need to introduce the four different types of competitors. Now, if you have an MBA, good for you. Skip this lecture. But if you do not, I think you should stay in because you'll learn something. Now, we have four different types of competitors. There's direct competitors, indirect, potential, and substitute. In order to explain the four, let's use a terrible metaphor. Now, let's just imagine that you are running a restaurant that serves Italian food. And specifically, you like to sell the tortellini. Tortellini is the pasta that has, like, cheese and meat inside of a thing of pasta. I'm not really an expert in this. Now, direct competitors are competitors that are not only going after the same target customer group as you are, they're also trying to solve the same problem you are. Now, direct competitors usually have very similar solutions, and users are often going to have to make a direct and conscious decision between using your product and using the competitor's product. In the restaurant metaphor, these are the guys that are literally across the street from you serving Italian food in a fancy dining atmosphere as you at the same price range. Oh, and also they specialize in tortellini. Now, indirect competitors are competitors that solve the same problem, but generally in a different way and for a different target customer group. They could be serving a smaller subset within your target customer group, or they might include your target customer group, but they have a much larger market that just so happens to also address your group. They could also be targeting entirely different verticals or sectors. In the restaurant metaphor, this is the restaurant that's in a different neighborhood that serves Greek food that people often confuse for Italian food. Now, the third type of competitor, potential competitors, are

competitors that offer something to the same target customer group or a similar target customer group, but they don't address the same problem that you do, or they don't have anything remotely in the same product line as you. I also like to refer to these as peripheral competitors 'cause you like to keep them in your periphery because they're worth keeping tabs on. These are the metaphorical dolphins in your tuna net. You were looking for tuna, but you got some dolphin in there. In the restaurant metaphor, this would be like an Italian grocery store. They don't serve hot meals on location, but they could foreseeably add chairs and a restaurant function. Now, the last one are substitute competitors. Now, what exactly is a substitute? It's a product that solves the same core problem, but it's not angled or delivered in even remotely the same way. It also doesn't generally target the same people. Now, with an Italian restaurant, you're probably targeting people who want to go out to eat, want Italian food, and are also hungry. Now, a substitute might be DiGiorno's pizza. It solves the core problem, which is you're hungry, you want Italian food, technically, but it's not going out to eat. It's not even remotely anything like it. But you could substitute it for going out to eat. Now, how this relates to what we're doing in this section, well, it's actually rather obvious. Go back to your list. Now, remember in the last lecture, I was using GoPro, so I made my own list of competing products for GoPro. Before we had it in known and unknown. Now what I'm going to suggest is you actually get rid of those and instead list off each one based off of the four different types that I've suggested to you. Are they a direct competitor, an indirect, a potential, or a substitute competitor? Once you have that, then what you can do is you can prioritize in your head and have a much cleaner idea of what the competitive landscape looks like. Put at the top direct competitors. We are the most concerned with those. Put below those indirect competitors, because we are also interested in what they're doing, not as much as the direct. Then we look at potential competitors, and the last one is substitute competitors. Now, the reason why we want to rank these in kind of a tier system, tier one, direct, tier two, indirect, tier three, potential, tier four, substitute, is that tier ones are the ones that we're concerned the most about, but also we need to make sure that our product offerings are as competitive as possible to those. Do we care whether or not our product or feature is competitive to the other three? Yes, but not as much as the

direct. The substitutes, we need to at least make sure that we are a viable alternative to. Potentials, we need to make sure that our product offering is something they can't themselves do easily. And indirect, we again just need to make sure that we're not losing too many customers to them. And there you go. That's what that would look like under the four different categories. It's a lot easier to understand. Now you could direct your attention to the ones that actually matter. All you have to do is dig into them and see exactly what they do. These up here in the direct category do almost the exact same thing as a GoPro Hero 4. In the indirect category, we have things like the Cybershot and the Powershot, which were cameras that could be used as action cameras. In fact, they're used for moving objects, and they do solve that problem. But they're not for the same people. They're targeted for people who want to take videos of their kid's tennis match or track and field match, not strapping it to their head and jumping out of a plane. We also see things like the Vievu, which I thought was a GoPro direct competitor. Turns out it's not. It's a body camera for police officers. So, same problem, need to shoot things in action settings that are small and portable, but not the same people. And then there are a couple other ones. Under potential, I put the Eyeshot 360, which I realized actually is a wi-fi enabled camera, and they target it primarily for surveillance means, like putting it on your dash and recording or putting it in your apartment and recording. Not the same problem, not really the same people, but they have the same expertise, the same general area. They could be a potential competitor. And the last one is a substitute, which is Opteka, which I didn't realize was not a competitor. It's actually a grip handle that you can put other cameras on and then use it like an action camera. So that would be a substitute. They could use it, but it doesn't solve the problem at all the same way. Okay, guys, we're moving on to more analysis of how to judge your competitors.

The five criteria for understanding competitors

- Hey, guys, welcome back to the course. All right, so this far in the section, we figured out how to find competitors, how to compare them to

ourselves, how to compare potential features and potential products we're thinking of to our competition. But now we need to talk a little more broadly about as a product manager and as an entrepreneur, what in general do you have to keep tabs on with your competitors? Some I'ma introduce five separate criteria that are crucial for you as a product manager or entrepreneur to understand about your competitors. Now, a large part of what makes you successful as a product manager is your ability to keep tabs on your competitors and make sure that you are ahead of the curve and know what exactly they're going to do. One way of thinking about it (laughs) is that it's kind of like a horse race. And especially if you're working on a team that has an established product, you're going to be constantly jockeying with your competitors for users, features, press, funding, amongst a number of different things. And in order to be successful in this jockeying game, you can't just keep tabs on them. You have to have a deeper sense of what they can do. So let's talk about five different dimensions that are absolutely crucial to understand about your competitors. And the first one we're going to talk about is called the product core. Now, that's a little confusing. But what I mean by product core is product team. I'm talking about whoever on your competitor's side is in charge of making their product. If it's a physical product, it's going to be the physical engineers that are building it. And if it's a, let's say, web application, it's going to be the developers. And the question we want to answer here is how good is our competitor's product team? Mark Zuckerberg, the founder of Facebook, made a very (laughs) notorious public statement when he said that any day he would take one great engineer over a hundred average engineers. Now, that's obviously kind of counter intuitive, right? You would assume more people can do more, so what exactly does he mean here? Well, what he means is that especially in web development or development of applications, having a great engineer actually can do a lot more for you than a lot of people that can get low level work done. Top-notch engineers in general can outperform larger groups of good engineers. Instagram is a great example. Instagram, when it was bought by Facebook for $1 billion dollars, you would think was a huge company. No, actually, they only had 13 employees. I believe only eight of them were product engineers. So know how good their product core is. Make strides and make attempts to figure out that question. Now, why does this matter to

you? Well, if your competitor has, let's say, a better product team, that means in general they can out-execute you. They can out-engineer you, and they can outproduce you. So if you're in a situation where you have a team that's not as good as your competitor's, you need to take that into consideration and probably not wade into features or wade into product categories where one of the key differentiators (laughs) or the key components of it is the engineering quality. Of course, if on the flip side, if you have a better product core, it's the exact opposite. You can wade into areas that rely heavily on the quality of engineering. And if you do that, that's something that your competitors will have a hard time catching up to and imitating. The second criteria we need to use when considering our competitors is to know the size of their user base. Now, why does it matter how big your competitor's user base is or how big their customer base is, in the case of more traditional businesses? It's kind of counter intuitive. Right? And immediately, it's hard to figure out what that would matter, right? 'Cause if you had a large user base, that doesn't necessarily mean you can build better products faster. Well, actually, this is a pretty important one. Companies or competitors that have large user bases have certain advantages that companies with small user bases don't have. Oftentimes, when a company has a large user base, any time they launch a new feature or launch into a new market, they have the potential to essentially dominate the market just by virtue of how many people they bring into it. They also have an easier time getting press. In general, press and press coverage is going to be more inclined to cover companies that are larger and more well-known. Another advantage is that a large user base will allow you to strike deals with other companies more easily. Much easier to negotiate when you say, I'm bringing x amount of users to the table. I'm the largest company in town. And the last thing is that size often is an advantage intrinsically. Think about social products, right? A social network is only as valuable as however many people are in it, so the larger one is inherently going to have a better product. Now, here's a classic example, and it's Apple Music. I don't know if you've heard this, but Apple's launching into the music space, specifically the subscription music space. And they're going to compete with Spotify and SoundCloud. Sorry, Cole. Now, can Apple make a better music player? We don't really know. And are they at a disadvantage because they're coming late to the market? Actually, none of

that matters because their user size is so large. Consider this, Spotify and SoundCloud have something around 30 million combined paying users. Now, think about Apple. How many users do they have? It's actually one billion. So even if Apple comes into that space, converts 1% of their market into their Apple music product, they are serious competitor. All right, guys, take a sec to think about those, and I'll catch you in the next lecture, where we cover the next three, see you then.

The last three criteria for understanding competitors

- Hey guys, welcome back to the course. So let's round out the rest of the five criteria that we need to focus on as product managers. Now, the third criteria is design. Now be aware of your competitors ability or inability to make products that are aesthetically pleasing or beautiful. Now it sounds right shallow, but in general, people are more inclined to use products that are well thought out and well designed. How many things do you use simply because it's easy to use and maybe it's pretty, and I'm sure you guys in your head have figured out the next example. Again, it's going to be Apple. Apple's a great example because they are so focused on design and usability that they are a threat to virtually any market they enter. Now they are especially threatening to markets that use design and usability as a key differentiator. Is Apple going to enter in the toaster market, probably not, but you just saw what they did to the watch market. Another real life example is a company Path. Path was making a social network to compete with Facebook. And at a certain point, believe they had something like 30 million users. They stuck around for quite a long time, solely based on their ability to make a more beautiful product. And their product was very nice. If I were in a similar space, I would be weary about treading into a space controlled by a firm that has top notch design. Now number four is brand and brand is kind of a funny thing. We often think as something that's really abstract, but when it comes to managing products and your competitive advantage, it actually can be huge. The perceptions that people have of you and the products you sell can determine what you can do in the future. A brand can be a blessing and a curse. It can allow to be more effective in certain areas and restricted in

others. In general, strong brands demand a much higher level of customer loyalty. When they launch into a new market or a new initiative, they're more likely to bring a large group of people to that. They also have the advantage of being able to charge higher prices. Remember our comparison of GoPro? We probably thought, GoPro why charging $500 and other people are charging $300 for the same thing. Well, because GoPro can do it. They have the strongest brand in the space. If you look at the market share, they are doing very well. So be aware of the relative positioning between your brand and the brand of your competitors, especially as it specifically pertains to new products or new features. If your competitor has a stronger brand, it can be hard for you to catch up with any new initiative or product that they built. They're inevitably going to get more, press more funding. And since they're a stronger brand, they're going to get a better benefit of the doubt. People in general are more flexible when they're trying out a brand they already have an affinity for. Now on the flip side, what about in the case where your competitor has a weaker brand, maybe it's diminished, or maybe it's actually conflicting. It's really hard for a brand that's pigeonholed in one area to branch out into areas that people would perceive as unrelated, Sony can make a TV, a camera, a toaster, and a car, if they wanted to. Now Lexus cannot make a toaster. They cannot make a TV. They can't make a camera. They can really just make cars. So be aware how highly regarded is the brand of your competitor, and what's the gap between the perception of your brand and the perception of theirs? Does that relative difference allow them certain advantages or certain avenues that you wouldn't be able to take advantage of? And again, be aware of any weaknesses to exploit. Are there certain things your brand can do that there's couldn't? And the last thing we're going to discuss is speed. When your competitor sits down and starts working on a new product or feature, how quickly can they build it, turn it around and ship it, so to speak? Try to get a sense of this because it's very relevant in a lot of different situations. Now, when is it most relevant? Well, this is most relevant in situations where your company or your competitor's company is significantly larger. In general, it's a natural consequence that as a company gets larger and larger and larger, they'll have more communication structures, more hierarchies and more bureaucracies that will typically slow them down regards to what they're trying

to execute on. So be aware, are you getting so big and so slow that a small team that's more focused and more agile can beat you to any new market, feature, or user base you're going after. If that's the case, then you probably need to shy away from situations where the product success is directly tied to the product launch date. One of the reasons why MySpace is not the number one social network anymore is because Facebook started off as a smaller group. They were able to build features faster and they basically just ran circles around MySpace, while they tried to catch up and it took them forever and it didn't work. All right, so those are the five product core, user base, size, speed, design, and brand. Get a decent understanding of your competitors in each of these five different criteria. That's what's going to allow you to make the right call. In the next lecture we're going to talk about where do we go to get this information and what techniques can we use to constantly monitor it in case it changes. All right, see you in the next lecture.

Monitor competitors

- Hey guys, welcome back to the course. So, in the previous lectures in this section, we covered how to find competitors, gather information on them, figure out if they're direct or indirect, and then, compare them. Now, those are all crucial skills for gathering the information you need to make judgments on your competitors. However, competitors are always going to be there, if not forever, at least for a while. And like, you competitors will change. Competition is an ever changing landscape. And as a good product manager, which is what you want to be, right? You want to keep tabs on your competition so that as soon as they make a change, you can redirect or adjust. Now, remember, the five dimensions that we judge our competitors on. We have the product team, how good are they or how bad are they? We have design, is it pretty? We have brand, how important is their brand? Is there a positive asset for them? Is it so strong it prevents you from doing things? How big is their user size? Is it big or is it just girthy? And the last one is speed. How much of it are they on? Now, those are the five criteria that we want to keep in our minds constantly about our competitors, and it's always going to change. They're going to make changes and take initiatives that will

increase certain ones of the five and sometimes they'll make mistakes, and they'll lose some ground on some of those five, and you need to be there to know when it happens. So in this lecture, we're going to cover three different events that you should constantly be looking for and we're going to cover three different ways that you can keep tabs on, if those events happen. Or another obviously, a ton of tiny little things that can happen with your competitors and it's worthwhile to know. Maybe they move a button over here, maybe they make a big change to their web app, whatever it is, it's all worth knowing. But there are three things as a product manager that I think stick out more than the others because they have the ability and the potential to change all of those five criteria we previously mentioned. So, the first one is funding. Now, this is not really something I've talked a ton about before, but a lot of companies receive money. If you're in the startup space, a lot of companies receive venture capital. The amount of money that's in their bank account can make an enormous difference for a number of our five variables. More money means they can hire more people. More money means they can spend more money on ads, means that they can hire more contractors, they can pay for more press. So, that means that they can dramatically change their speed. More money also could mean that they now have more to work with for recruiting, for their product team, and so, you should keep your eye out in case their product team improves. I does a small changes to brand that doesn't necessarily change their design, but it could allow them to bring on people who can add more design and usability. And it also could or could not have an effect with user size. Oftentimes, especially with startups, when they raise money, they do spend a lot of it on advertisement and growing their user base. All right, but how do you know when they get funding? Where are you supposed to put your ear to the ground for this? Well, actually, it's pretty easy. Generally, when companies get funded, there is a website that keeps track of all of that for us, and it's incredibly useful, it's called Crunchbase. So, let's check it out. This is what Crunchbase looks like, just crunchbase.com. All they do is keep track of how much money people have raised. Oh my God! WeWork raised $430 million, that's insane! Sorry, and back to what we were talking about. All you have to do is search up here for the organization that you're trying to keep tabs on. Let's say, I work for Spotify. Sorry, again, Cool! If I wanted to keep track of

SoundCloud. SoundCloud, honestly, is the least of their worries, I'd be more worried about Apple Music. But let's say that we wanted to keep tabs on them. Right here, we can see information about them, all this stuff is really nice. But the most important thing is it gives us a timeline of their funding right here. So, not only can we get historical information, we can also see when this gets updated and that'll tell us that they just received a new round of funding. So, how do I get notified when this happens? Well, it's pretty easy. You just hit Follow, and then it'll ask you to create an account, do that, and then you'll get emails directly to your inbox every time something major happens with your competitors, specifically, their funding. There you go! Another thing that Crunchbase is really, really helpful for, happens to be our number two thing that we want to look out for, and that is acquisitions. Acquisitions, if you're not familiar, is when a company buys another company. Imagine a really big fish and a really small fish and then the big fish eats the small fish. Except in this scenario, that small fish actually attaches to the big fish and makes it larger, kind of like the Anglerfish. I know that's random, but look it up, it's the creepiest thing in the world. When a company buys another company, they're doing it for a number of reasons, and it usually comes down to the five reasons that we care about. Oftentimes, companies will buy another company because they have a good product team, and they want to integrate that into their own team. Sometimes, they want to buy a company because they have a lot of users and they would like to integrate those users into their own user base, or at least, cross-sell them information or gather information on their users. Either way, bad for you! They could also buy them because of design. There are examples of companies that have bought other companies to absorb their design team. Remember, when we talked about Path. Path got bought for pretty much that reason. Now, there are really two things that doesn't necessarily affect our brand, unless acquisitions are prestigious, I mean, they're kind of are or Speed. Speed is tough. Oftentimes, companies will buy another company because that company seems to get stuff done really quickly. But, you kind of run into a problem where a larger organizations as a whole, as they get larger and larger, gets slower and slower. So it's... I don't know, it's kind of a wash. How do we find acquisitions? Is we just go up here and they're listed right at the top. Click it

and it'll take us to a page that gives us all their acquisitions. It gives us the date, we can actually click through and read about the company if we want to. And since we're already following it, we will get an update. It will only give us the terms of it, that's pretty cool. And the last event that we really care about are feature or new product launches. Why? Well, because those are major maneuvers that your competitors are making and you absolutely have to know if they're doing and especially, if they make a product or a new feature that is in some way competing directly with you or copying you. The best way of keeping tabs on that and a number of other things, it's a website called mention.com. Now, it's not free but it is incredibly useful and it has a free trial. More importantly, what mention.com does is it tracks all of your competitors, social media accounts, it looks at chatter between customers, and the company. It looks for major updates on the companies blogs. Any large seismic shifts, they will email you, and you can even set your preferences to tell it how often to email you. Only email you when big things happen or email me any time anything moved at all. Let's check out Mention really quickly. Sign up, it's free, and then you can create your first alert. So, all you want to do is tell it what you're trying to train their listening system to. So, we're looking for a competitor. Tell them the name of your competitor, we'll say SoundCloud, there we go! And move on to the next step. Here we can set our priorities, what we care about the most. If you're more concerned about what they say in their social channels, you can make sure to listen to that more, add more of them. But we're just looking for feature changes, product launches. So, when those happen, it's going to be a seismic shift, they're going to have a blog post about it, they're going to have a Twitter post, and probably a Facebook post as well, you'll get one of them. So just move on. You can pick what, all you want to pull in. This is incredibly useful for the news setting. Why? Because, a lot of times, you might not see the feature launch but news will pick up on it or news will pick up on it way before it launches. That's really helpful. Also, for tracking things like perception which affect their brand. Move on to check out your results, were almost done. And there we go! Our account is already set up. Look, we have 76 new mentions for the alert, SoundCloud. All right, so it went out and gathered quite a lot of information about SoundCloud. It's going back in time. If I wanted to look at the mentions, I would just go here and then I can

just scroll down this feed, and I can check it out. Here, I can search and I can filter through it. There are a lot of really cool tools for picking out the signal and the noise, so to speak. You can set things like, keywords you're looking for, or even exclude keywords so certain things don't ever show up in your feed. You can also, in your settings, change how often they email you and what type of interactions they consider important and will notify you about. All right, cool, Mention is a very powerful tool. Check out what you can use it for, you can find a million and one things you can do with it. This is fantastic for figuring out when they get money, when they get acquisitions, and when they launch new things. And the last thing I want to cover is Google Alerts. Google Alerts is not as powerful as Mention but it's a broader blanket. All you have to do is type in the company name that you want to stalk, and what it's going to do, is it's going to frequently email you and notify you if you want to do desktop notifications of anytime new search results show up for your competitor. That is the broadest sweep you can use, credibly useful though, to see how their SEO changes, to see how anything is popping up, conversations things way out there you might not have seen. All this information, really helpful. All right guys, keep track of your competitors 'cause if you don't, they might get out of your sights and that'll come back to bite you later. All right, see you in the next lecture.

What is a feature table?

- Hey guys, welcome back to the course. Alright, so at this stage, we have a big list of direct indirect substitute and potential competitors. Now let's use a technique that can help us compare ourselves against all of our competitors. It's called the feature table. Now you've seen a feature table before, looks like this? Basically what it is, it's a comparison chart that we can use to compare our product versus the product of our competitors on a number of dimensions. On the x-axis, even though it's not really an axis at the top, we're going to put our competitors and on the y-axis again, the left side, we're going to put the dimensions that we want to compare ourselves on. Now this exercise is meant to give you a bigger picture of how competitive you are in general in your market. Why is that important? Well, you as a

product manager or an entrepreneur need to know where your other competitor's products are because you need to know what is cutting edge and what is competitive before you come up with product ideas and before you launch your products. And the more obvious reason that if you launch products that are worse than what's already out there, then in general, you're probably going to fail. There are some exceptions, but most times, you're going to fail. Now, do you need to make feature tables on the job as a product manager? Well, not necessarily, but knowing how to make these will be immensely useful to you and it could become a valuable asset for you and your team later down the road because these are the best ways of conveying quickly and easily for everyone understand, are you competitive or not? You can't mentally keep track of every single thing your competitor is doing, adding and removing. GoPro has something like 50 plus competitors. That's insane. Alright, so let's make some feature table. Get some practice. That's what we're going to do in the next lecture. See you then.

Put together a feature table

- Hey guys, welcome back to the course. So let's make a feature table. Now you remember what those are, they're a comparison chart. And on the x-axis, the left side, we put the dimensions that we want to compare ourselves to our competitors on and on the top, the y, so to speak, we put the competitors that we want to compare ourselves to. Now, you can do it like I'm about to do, I whip out a piece of paper and I draw it out. It can get messy very quickly, but that's the way I prefer to do it. You guys can do it whatever way you want. I suggest if you're not going to use a written version, use Google sheets or Excel, why? Because it'll look clean, it's easier to view and also the cells can stretch as you type out more. So let's set it up. Alright, so first off, if you're putting your graph on a piece of paper, draw out your graph using Google Sheets or Excel, you don't actually have to do anything. Just make sure that you leave enough space at the top for, let's say, at least four to five competitors. You can put as many as you want to, but I'd say at a minimum, for now let's put four to five. And then on the other side, leave enough room

so that you can put a good variety of dimensions to compare yourself on. Now at the top, we fill in our competitors. Now which competitors do we pick? Because we had direct, indirect substitute and potential. I suggest starting with direct competitors, why? Because those are the ones that can be the most similar to you. But the differences between you and your direct competitors are the most important. I personally didn't like to take the indirect and the substitutes and the potentials, draw those in a separate graph, which I'll analyze after my direct graph. Now in general, direct competitors are the ones that you really want to stay up to date with. Watch their products, see how they change, see what they add, what they remove, and try to get a general idea of how they're going to react to you when you do something. Now the name of the game for your other types of competitors is to make sure that your solution is obviously better than your substitutes, your indirect competitors and your potential competitors for your target customer group. Use that table as a quick reference to make sure that it's obvious you are better. Now if you have a situation where it's not obvious, you are better than a substitute, you have a serious problem and you need to go back and start from scratch. In the restaurant metaphor that's like your Italian restaurant serving pizza that actually tastes just as good or worse than DiGiorno pizza. That's a fundamental issue you have to address. Now on the left, we're going to put features and factors. You can really put whatever you want over here. I suggest you start with these two features and factors, why? Because they're not very subjective. They're pretty easy to nail down and be specific about. Now what you choose to put in this part of the table is entirely up to you. You need to think really hard about what you think your target customer group cares about with regards to your product. When they go, and they're looking at your product, let's say it's a physical product, what do you think is going to be the one, two, three, four and number five reason that they choose yours over the others. Now, keep in mind, every product has a different vision, and a different positioning. So the features and factors that you think are relevant to your target customer might be different than what your competitors think are the most important. Maybe you think ease of use and price is really the key formula to winning your market while your competitor might think instead that it's design, aesthetics, and integration. The key here is not to put what your competitors think is

important. It's to put what you think is important to your customer group 'cause at the end of the day, your product vision is how you plan to win the market. It's your bet. It's not realistic to think that you're going to beat every single competitor in every single category, especially if you deprioritize certain aspects over others. So brainstorm your best and honestly do some research as well, to get a good chunk of what you think is important. Now, for any given product, let's say I might put price as one that's a very common factor. I might put reliability that could easily be a common factor and in physical products, or, you know, even software's that are dependent on things like uptime. And then as a feature, I could say, it needs to do feature x and it needs to have the ability to do Y. Note this is just a template you need to fill these in yourself. It'll help you kind of get through this. Let's imagine again, in that restaurant metaphor. What would I mean when I say features or ability to do something. I know it's goofy to think of a restaurant in terms of features, but it is actually a way of thinking about it. So I could say a feature that I think is important, and that my products going to have is that we offer a price fixed three course meal. And I think that's really important when people pick a high end Italian restaurant. Now for the ability to do something, maybe I think that being able to make reservations online is important. It's important to their perception of the restaurant, and that's going to get you more customers and a better experience. Now that you have it set up, go out, research your competitors, and the best way of doing that is to actually go out and use their products. Give each product an honest appraisal, put in the box what you need to know but keep it short and uniform. Answer a yes or no question with yes or no and then it's up to you, you can put a comma and put one or two more words that might be a little bit more specific. But the key trying to make them uniform, so when you glance at it, you don't have to read every single cell. Alright, so that is the template. Next lecture let's try to make an actual one. See you then.

Practice building a feature table

- [Instructor] Hey guys, welcome back. So let's make a real feature table. And let's do it for my favorite company, GoPro. Why they're my favorite, coz

they're very easy to beat up on. True story, they're the most shorted company in the world if you don't know what that means just Google, GoPro shorted company. Okay, let's see how this works in action. Now, on the right, we're going to put our actual product as usual, we're going to put GoPro and then we're going to fill out the other competitors. So let me introduce the competitors. Now, remember, when we did this as an example, and we wrote down some of the direct and indirect and substitute competitors, well, we're just going to make a feature table for the direct competitors. And I've just picked a handful that I think would be worth comparing. So let me introduce the competitors. Okay, number one is TomTom Bandit. This is the one that yes is made by that GPS company that was made obsolete the day Google Maps came out. It also looks to me like a plastic hot dog, but we're going to put it up there because it actually sells pretty well. Now the next one is Veho, now that's the one that looks exactly like GoPro. They copied every single detail down to like the tiny little corners and even the colors, I'm amazed it didn't get sued. And the last one is going to be the Sony Franklin Delano Roosevelt. They all have very strange names. Okay, so stiff competition. Now we need to think about what are the most important factors for the people that need an extreme use action camera. Now if I were a GoPro, I'd probably dip into some of our data on customers. I would research a very specific customer profile I was going for but for our purposes, we'll give it our best guess. I'm going to guess one of the aspects is going to be price. Why? Because I'm assuming a lot of our customers are broke surfers who don't want to spend too much if they don't have to. The other one I think might be important is aesthetics. Why? Well, because with an action camera, you're someone who does extreme things. Perhaps you're a vain person. I'm making judgments maybe not true. I think it's probably important that there's a cool factor to it. You really don't want it looking like the Microsoft Zune. The next one I'm going to put is weight, kind of spitballing on this one, but I'm going to guess since the idea is that they're small and portable, you might wear it on your head or your chest or something like that. That you don't want it to infringe on your ability to do whatever it is that you're trying to do. So I'm going to guess you don't want something that has the weight of a bowling ball. I'm going to put 4K as well. That is the new kind of cutting edge way of filming. We are not shooting

in 4K right now. If we were shooting in 4K, you would be able to zoom in and get into my pores. But for people that are doing this, I would imagine they want that. Why because higher resolution makes your pictures look better in general. So I would imagine if you're going to do something extreme or cool, you want it to look as cool as possible. The next one I'm going to put is Max FPS, which means frames per second. In camera speak, what I'm getting at is can it shoot in slow motion, right? So right now we are shooting in 24 frames per second. If I shot this in 60 frames per second, I will be moving really slowly and it might look like some of those sequences you see on commercials. So I want to see how slow these can go. I think that's going to be interesting because again, people use these for action shots, they want to slow down and watch the coolest parts. Last thing I'm going to put is accessories. Now accessories, I think are important, right because a small action camera, it's small, it's kind of awkward. You need to have an easy way of putting it on your wrist, on your head, on your chest. And using a lot of tools to get what you need done. You're not going to go grab a fork and duct tape it to it, that's just not going to cut it. Now there are obviously some things I ignored, and these are just decisions I made. I ignored things like battery life, ignored things like the ability to edit on the actual camera which is apparently a feature amongst the action cameras. I also decided on camera controls probably weren't that important. I don't really know how important it is to be able to scrub through your filming on the little box. It just seems like all you need to do is press record and go. Now I'm making these judgments because GoPro users probably want the cutting edge. They want to film really cool stuff in short periods of time. That's why I don't think battery life is going to be tops for them. I could totally be wrong and I don't think editing on the thing is going to make sense because people who use GoPros probably want to make a montage or a video that looks as good as possible. That's not something you're going to get editing on a tiny little box. Okay, I was lucky to get all this information just from individual websites and a top 10 competitors comparison website. Relatively easy. For you guys, it's going to be easy as well. Just go to the actual competitors website, they'll tell you everything you need to know. Okay? In reality, what I would probably do is go out, read consumer reviews, talk to customers see what they experienced, and most importantly, I would play with the actual

product. There's no substitute for you actually using their product and making a judgment. All right, let's fill it out for the GoPro, the GoPro cost close to $500, 499. Aesthetics, they're all kind of ugly. I think there's some improvement, but for the most part, I think it's probably a seven out of 10. It's better than a lot of them. Weight, 89 grams that means nothing to me right now. Does it shoot in 4K? Yes, it actually does shoot in 4K and I'm going to put a secondary piece of information, which is that it can shoot up to 30 frames per second when it's in that high resolution format. Max FPS, I just put yes because it can go above the normal fps and it can go up to 120 if you're shooting in 1080p. For those of you don't know what that means you, don't worry about it. Accessories, I put 100 plus because when you search on Amazon there are an enormous amount of accessories for GoPros. Okay, cool. So now let's just start filling them out. Tom Tom bandit, this is what I gathered from their website. They cost $349 pretty expensive, but cheaper than the GoPro. Aesthetics, I would give them a four out of 10. Again, I said they look like a plastic hotdog or submarine with the big eyeball on the front. I honestly don't know what they're going for. Do you just run around with it like this just pointed at people like a laser pointer? Weight, 190 grams, I'm not sure what that means. But in relative comparison to the GoPro that's a big honking item. 4K, yes it does do it but it only shoots in 12 frames per second. That's questionable. The max FPS is 60 frames per second versus our 120. And does it have accessories? Yes, but not very many, only about 10, I could find. Moving on to the Veho. The Veho is cheap, it's $200. The aesthetics are very similar to the GoPro. It's just a little bit worse, so I give it a six out of 10. The weight way is 84 grams. Can it shoot in 4k? No it cannot. The max FPS is again like our favorite plastic hot dog, it's about 60 frames per second. Does it have accessories? More than the hot dog, but still not a ton. I think I found around 15 ish. And the last one the Sony Franklin Delano Roosevelt. That one is probably the closest competitor. This one is $399. Aesthetics, I give it a six out of 10, doesn't look as good as the GoPro but it's not that bad. Weight, it's 114 grams. It can shoot in 4K and it can shoot up to 30 frames per second compared to the GoPro, does exact same thing. The max FPS is 120 and the number of accessories I could find was over the 25 range. Okay, there you have it. Now what we could do is internalize this and understand are we beating our direct competitors? Are we

the obvious choice or are we only appealing to one segment? Are there areas we should focus our energy on improving? Are there areas that we should just entirely ignore? Answer that question. What do you think? What should GoPro do, based off of what we just filled out?

What do we ultimately care about as a product manager?

- Hey, everyone, this section was all about competition in market analysis. What you learned from Evan's lectures is stuff that's really vital to understand if you're developing a new product or company from scratch. However, if you are a product manager at a company, or looking to get a job as one, this stuff will be extremely important to understand from a theoretical level. Like we've mentioned before, if you're working as a product manager at a company and they don't know who their competitors are, then that's a red flag. The people you work with, and of course, the executives, will know exactly who the competitors currently are, or if there are none yet, then they will know who's in a position to start competing with you soon. The biggest tip I can give you as a produce manager, when it comes to competition and market analysis is to stay on top of industry news and pay very close attention to the features that competitors build. It's also great to look across industries at similar patterns, even if the company does not directly compete with you. Here's an example of that. If I'm building social features for an app that lets people read books on their smartphone, I don't only want to look at other smartphone book apps, I should also look at other apps that are social in general, like Facebook, Twitter, and Instagram to see what they have in common, how users are interacting with them, and what I might learn from the lessons they've already learned, since they've already been doing social. Overall, if you're always up-to-date on what competitors are doing and have informed opinions about where they will go next, you will gain respect from coworkers and go really far in the role of product management. So that's it for this one. I'll see you all in the next lecture.

What is customer development?

- Hey guys, welcome to the next section, super proud of you for making it this far. Now in this section we're going to cover a very prominent concept and startups in large businesses today called customer development. Customer development is one of the number one tools a product manager will use to get any sense of whether or not you're building the right thing, what the market will accept, and what the market will absolutely just reject. So it's something you have to know. Now, a lot has changed since this book, The Lean Startup came out, companies startups. Even small businesses are looking at their markets and looking at their customers as well as their business ideas in a radically new way. If you're unfamiliar with The Lean Startup, we'll cover that later in the course so don't worry. But the old way of thinking was that you would sit around come up with the best idea for what you think your market will accept, then what you would do is you would build that and push it out to the market. The attitude in general was that some products and some new features succeed and some fail, such is life. You go from concept, to production, to promotion, and that's the entire lifecycle. Now, if there's one thing we've learned from The Lean Startup, it's that you can't come up with product ideas in a vacuum. There's a lot more that goes into what makes a product successful and what makes a product unsuccessful. And that information nugget that you wish you had before you started building your product and before you launched it, is actually held by the customer. So customer development in a nutshell is the practice of establishing a continuous and iterative communication line with your customers, so that you can come up with ideas, come up with hypotheses, try them out with your customers, get feedback and use that and feedback to inform your product decisions going forward. By establishing a line of communication with your customers, you can on a constant basis test and validate your product ideas with regards to who's going to buy it and the market in general. Now, I know a lot of you guys out there have heard of customer development and you probably are thinking that's part of it, right? Well, let's talk about customer development really quickly from an academic sense. Now, customer development is actually a framework and was developed by a man named Steve Blank. He is a professor at Stanford and a thought leader in startups and how small companies can

disrupt large industries. The concept as he teaches it, was expounded in his book called The Four Steps To The Epiphany. That's any hint there are four steps to his customer development framework. Step one is discovery, step two is validation, step three is creation, and step four is building. Now, he teaches the customer development mindset as something that permeates every single section of your product lifecycle, from coming up with the idea all the way to growing your business. And this is absolutely true and product managers do use customer development in every single section of their business. But for our purposes in this course, we're going to focus on the parts of the customer development framework that are actionable, and are the most useful, and that is the customer interview. As you manage a growing product or even a budding brand new product, you're going to want to increase your product IQ by understanding the real reasons why your customers use or do not use your product. And by doing customer interviews, we can hear it in their own words, which is incredibly valuable. So for a product manager or a founding entrepreneur, customer development is going to look like this. Now, the first step in the process is that you're going to come up with an idea, this is the core nugget that you're going to build your enterprise or your product on. And that means that you're going to instantly go into the process of validation, try to figure out whether or not this is a good idea to build. At this stage, you're going to use customer interviews to figure out whether or not your product is needed, whether or not the problem you think you're solving is a real problem, and whether or not customers actually have that problem and are interested in your solution for it. You're going to constantly reintroduce this feedback that you've received into your validation process, eventually landing at the MVP stage, and then on to the next stage, which is the actual development of your first version one product. Now during the development phase, you're going to be using customer interviews and customer development as a way of figuring out what features are the correct ones to build and how should you prioritize what goes into your version one, so that you can be efficient and save resources. After your version one, you go into a process whereby you're going to start iterating on your product. You're going to start improving it, adding features, changing things that don't work out. And at this stage, you're going to use customer development constantly to figure out things like, are customers enjoying the

product? Are they using it correctly? Who exactly is using the product and getting the most out of it? Is it the group that we thought it was? You'll use this as a tool for figuring out whether or not you're targeting the right people, figure out if there's anything that you are missing and to give you clues as to what new features or new avenues might be open to you. Customer development is primarily a tool for two things. One, risk mitigation, and two opportunity recognition. And that is what you are going to use customer development for as a product manager. So in this section, we're going to focus on how do we do that iterative process? How do we do it correctly? How do we know if we're getting the right feedback? Who exactly do we talk to? And what insights can we get from all interviews? Alright guys, hope you're excited. This is an incredibly useful skill you will use regardless of whether or not you become a product manager. See you in the next lecture.

The four types of interviews

- Hey guys, welcome back to the course. So, in this lecture, we're going to start talking about the actual customer interviews. In this lecture, we're going to cover four different types of interviews, talk about why we will be engaging with customers, what we're trying to get out of them, and how in general these interviews are conducted. Now, as a product manager, there are numerous reasons why you will want to engage with your customers in an interview setting. Entrepreneurs, on the other hand, generally will talk to their customers at the beginning, at the outset, just to figure out whether or not they're onto something and they'll then use that information to figure out how to build an MVP. But there are other reasons why you might engage with the customer, and it kind of just depends on what you need as an organization, as a product owner, and where you are in your products progress. Now, the first type of interview is the exploratory interview. Now, an exploratory interview is the most free-form interview there is. The idea here is you're really just waiting out to see, talking to your customers, trying to establish whether or not they have certain pain points, whether or not they're open to certain solutions, and in general, what do they feel like? What do they look like and how do they operate? The idea here is that, you are exploring,

you're fishing, you're trying to find things your customers have a problem with or excited about. You're trying to find things that they do differently than other people. You're looking for any little clue or hint, that can give you some type of insight about the product you already have or the one that you're building. This is often something you will do when you're trying to come up with ideas, ways of changing your product, ways of expanding it. You will have a backlog but, you do have to come up with new ideas. When you talk to an actual customer, talk to them about their day, the context in which they use your product or could use your product, they'll give you a million different things that you could build a solution for. And the best part about doing this in person is you can visually see how badly they want this solution and whether or not they would pay for it. Here's an example exploratory questions that you might ask a customer. Let's say we're using an app that has to do with getting directions. I might ask, what's the worst part of your commute? It's an open-ended question, fishing for some type of pain, and to understand the hierarchy of problems they have within the context of a certain situation. Another question I might ask is, when you get lost driving, what do you do? I'm fishing for substitutes, for solving their problem, and again, they'll color-in kind of my picture of what normal consumers do when they're lost. Now the second type of interview is the validation interview. And this is by far the most common one. The purpose of a validation interview is that, you have a theory and you want to test it out. Your theory might be that, users will go bananas for feature x. Or you might be pre-product and your theory is that, my solution solve their problem. That is always a theory, until you launch, until you get validation and until you reach product market fit. Now let's talk a little bit more about validation interviews because they are the biggest deal of them all. You will do them as a product manager and you will do them as an entrepreneur, unavoidable. Generally validation interviews, are run in a very specific way. Because you have a theory, and you want it to be proven or dis-proven, you kind of have to run it, in a scientific way. Now, do you think that in a validation interview, you just walk right in there and say, "This is my product, or this is the feature I'm adding, what do you think? Do you like it?" If I were standing right in front of your face, like I am now, but coming through the screen, and I asked you, "Do you like my website? I spent six months working on it." Do you think you're

more likely to say yes or say no? You're probably more likely to say yes, because you're probably a decent person, and you want to please me and that's generally how human nature works. Validation interviews are hypersensitive to bias. So much so, that they're run in a way that, you don't even introduce your idea, your theory, your product, your new feature, till the very, very end. You spend the majority of the interview, talking around the problem your product solves. When you do introduce it, you don't hype the idea, you don't defend it, you don't exaggerate what it is, you simply present it and then collect honest feedback, and try to chart how they react, and how they think after it. The problem is that, if you bias their answers in any way, if you infringe on their ability to think freely, even if it's very subtle, you have essentially ruined the interview, any data you got from that is junk, it's poison. So learn to zip it and let them do the talking. Let them in their own words, describe the problem they have and what they think of your proposed solution. You spend the majority of these interviews not pushing your customer, but corralling them in the right direction, trying your best to get them to talk about the problem you are trying to solve, in their own words, and to see if they themselves find it on their own. Let's give some examples of a validation type of question. I might ask you, "Do you ever lose things in your house?" Followed up by, "Do you ever lose your keys?" What I'm doing there is I'm asking them about things they lose in their house, seeing if they say anything about their keys, and if they don't, I try to narrow the focus down to keys. And then I asked them if they lose their keys often, to see if that is actually something they do. Now that's a good pre-product question, but what about a post-product question? Well, a post-product validation question could be like, when you're using our app, and you want to collaborate with your co-workers, what do you end up doing? This might be on the heel of a feature, that allows them to collaborate within your app, and you're trying to chart how they solve that problem, whether or not they have that problem, or it's even something they care about. The third type of interview is a satisfaction oriented interview. And satisfaction interviews, they're exactly what they sound like. You're trying to dig out what parts of your products are working for your customers, and what parts of your product are not. You're also trying to gauge in general, what they think about your product. As a product manager, you need to know whether or not, opinions of your product

are going south, if they're jumping ship and going to your competitors. But more importantly, than just knowing that, you need to understand why they are. So get at the root of their dis-satisfaction, or, figure out what exactly you aren't doing right. Some example questions might be, what should we stop doing? A great open-ended question, trying to get them to tell us, what exactly, simply is not working. If something's abrasive, if something is just going against the grain for their experience, they'll tell you. Another one I like to ask is, what's one thing I could do to make this better for you? If they have something that is lacking, they will bring it up. It's a great way of digging out, the number one thing on the top of their mind, and whatever is most important to them. Now the last type of interview is an efficiency interview. In an efficiency interview, you're trying to figure out how exactly Your product in-grains into the life of your customer, and as a function of that, how can you improve it to better serve its purpose. You're trying to get an idea for, who uses your product? What do they do on it? When do they use it? Under what circumstances? And from that contextual learning, you can figure out what parts are serving them well, what parts are unnecessary or inefficient? And what parts are actually degrading from what they're trying to do. You fish for things like, are they using X, Y and Z feature and what are they using X, Y and Z feature for? It's very common, you'll find people not using your features or your product, exactly like you thought. When you do find that, that is generally a sign that something is broken in your process. Either something's too hard to understand, or the process is too inefficient, or you're just targeting the wrong people. Some example questions you might ask in an efficiency type interview is, how easy is it for you to use feature x? You're just trying to figure out in their own words, is this something they enjoy using? Is it easy for them to use? Is it easy to understand? all of that. Another question might be, something along the lines of, if you wanted to do blank, how would you do it in our product? It's always interesting to ask customers to show you how they would do something, and you can see, at what point in time they got confused, mis-directed, did something wrong, or figured out something you never thought of. Okay, so those are the four different types of interviews, and I know you're asking the question, well, aren't there interviews that do both, or a combination of them? And yes, absolutely. So, don't necessarily think of them as just types of interviews, they're more the

purposes that you would use an interview for or whatever your focus is, on an interview. You can definitely do validation interview, and an efficiency interview in the same setting. All right, in the next lecture, we're going to talk about best practices when you are talking to your customers. How can you make good questions? How can you tell when you're asking the wrong questions? And what in general do you want to shy away from? All right, see you in the next lecture.

Key differences in customer development

- Hey guys, welcome back to the course. Hey so I am going to make a quick pit stop to talk to you a little bit about how customer development is going to differ in your career as a product manager, and this will help you understand how we teach this section. So the one thing you need to understand about customer development is that it evolves throughout the life cycle of your product. You, as the product manager, your role is going to evolve as well as your product goes from baby to teenager to adult. Now what you need to be aware of though in customer development is that there is a very large chasm or divide between when you are validating a very new product and reaching out to customers, or when you have time to improve an existing product with an existing customer base. Those are two roles that every product manager will actually have to tackle. However, we treat them a little bit differently. Now when you don't have a product and you are trying to figure out whether or not you are going to build a product, you are primarily going to be running the customer development and running the customer development interviews with potential customers, and when you have a product that means running it with existing customers. Now it doesn't sound like there's a huge difference but it is actually an enormous difference. Potential customers do not know you. They probably don't know of what you are trying to do. The information that you are going to try to get out of them is different. The way that you have to reach out to them is different, and in the second case it's also very very different. Your customer that you are going to be talking to you already know who they are. Reaching out to them is generally easier, and when you get them on the line or in person it is an entirely different spiel. Now

keep this in mind as we go through this section, customer development will vary in terms of purpose, but very rarely in terms of form. We still will use the same guiding principles that we learned in this section for either pre-product or post-product interviews. The primary difference is the type of information that we are trying to get out of them. So as we go through this section, I am going to be dividing everything as pre-product potential customers and post-product existing customer, and in a couple cases I am going to explain one. Next lecture explain the next one. I suggest you learn both techniques, but if you only want to learn they type of situation you are in right now, say you are an entrepreneur and you want to move product, then you can focus on pre-product potential customers. and if you are someone who knows that you want to manage a product that already existing you can focus on the other, but again I'd suggest both. Alright see you in the next lecture.

Who you should talk to

- Hey guys, welcome back to the course. So let's in this lecture start the process of customer development. All right, so in this lecture, we're going to cover the first step of the customer development process. There are four steps to this and I will fill you in as we go. So step one in the process is figuring out who exactly we want to talk to in the first place. Now, you obviously don't want to talk to just literally anyone, you want to make sure it's either a potential customer or a customer you already have, that is relevant to the new features or new products you're trying to launch. Now, as I explained in the last lecture, pretty much everything we're going to do is divided down the middle between pre-product and post-product. So first I'm going to explain the process by which we can figure out who we are targeting and who to interview for pre-product and then at the end of the lecture, we'll cover post-product. Okay, for the pre-product scenario, you have not yet created it. You are probably in the validation section of development. You're still trying to figure out what should go in it. Do people actually want it? Is there a market? Now in 90% of circumstances, you probably in your head, have already thought of a target group of people that you think this new product is going to appeal to. If that's the case, then you're a step ahead. Take the idea

that you have and write it down. We're going to try to get to at least three potential groups of people to talk to. Now, if you're sitting there right now and you have absolutely no idea who a potential customer for your new product is going to be, I would be worried. You're in a situation where we call solution looking for a problem. And I hope the next exercise we go through will be beneficial. Now I want you to write down at least a minimum of three different types of customers or customer groups that you think would find your product appealing. Why do I ask for three? Well, because any product out there is going to appeal to not just one type of customer, it's going to have appeal to other types as well. It might not be as strong, but it still has an appeal. Now write down the customer groups you think, answer that question. Once you have three, then we're going to sign, which of these three is worth targeting first. And we're going to do that by playing a game called 3 Questions Stud. I just made it up. 3 Questions Stud is based off of the game, 5 Card Stud. And if anyone plays poker and has played 5 Card Stud, you know, it's the dumbest game in poker. But for our purposes of figuring out who to target, this metaphor works perfectly. The other reason I call this 3 Card Stud is that we're going to judge our potential customer groups on three different criteria and much like a game of stud. You get three criteria, three ratings, and that is it. If you're pulling 3 Card Stud in real life like the poker game, you would get three cards. Your opponent would get three cards. You would look at them. Whichever one adds up to more or as a pair or whatever, wins. That's it. Now the first step in this game is to ask yourself a question and then answer it to yourself. Your product solves a problem for someone. And that is the case with 100% of new products. If you think about it long enough, you will think of a problem that it solves. Google Maps solves the problem of, I don't know how to get where I'm going. I get lost easily and even frivolous things like mobile app games, those solve the problem of being bored during short periods of time, where you can't engage in other things. So once you figure out what the problem is, ask yourself the question, who has this problem? Now, if you dwell on that, you should be able to come up with at least a couple answers to that question. Now there might be customer groups you've never even considered and obviously there'll be the ones that you probably had in the back of your mind. All right, so grab a pen and a piece of paper or an Excel sheet or a Word document, Evernote

doc, whatever it is that you're using to capture your ideas and your thoughts. Draw a table and make it three columns and three rows. Now on the top, we're going to write our customer groups, customer group one, customer group two and customer group three. Fill in the actual names of the ones you previously came up with. Now on the left side, we're going to write down the three criteria and this is pretty much the name of the game. So the number one we're going to write is size. The second one we're going to write is called pain to payment. And the third one we write is accessibility. Together, these criteria are probably the fastest and the most relevant criteria to use to determine who is actually worth going after right now. Now size might not be obvious. When I say size, I mean market size. Think of the customer group you are talking about. How big is your customer group? Is it something that has thousands of people or millions of people? Let's say you were building a system that helped property developers and real estate agents. And those were the two that you're going after. People who run property development, companies like CEOs and real estate agents. How many CEOs in the world are there that run real estate or property development companies and conversely, how many real estate agents are there? So the first case, there's not a lot of them. So that customer group would rate low on this criteria. The second one, real estate agents, there's a bajillion of them. So this would rate high. Now, second criteria pain to payment. This is a ratio. Now I want you to think about each customer group and think about how much pain do they have associated with the problem you are trying to solve. Someone who constantly gets lost in the city, probably has a decent level of pain and someone who is bored, waiting for their doctor, that's not a huge level of pain. Now, the second part of this criteria is payment. How likely are they to pay you for this service? And if they are willing to pay you, are they willing to pay you a lot of money or a little bit of money? And based off of those two, you can kind of come up with a good ratio or a bad ratio. High pain, high payment is very good. Low pain, low payment is quite poor. In the Google Maps example, we have high pain, but we have very low payment because there's tons and tons of free options out there. So it actually would rate somewhere in the middle. And with the mobile app game, again, we have low pain and payment is also quite low. You rarely pay more than say $5 for a mobile app game. And the last one is

accessibility. Accessibility just means how easily can you get in contact with these people? Are they easy to find? Do they have their contact information posted somewhere? Or are they used to receiving queries from the outside world and responding to them? To go back to that first example, CEOs of property development companies, very inaccessible, their information is not going to be public, and if you email them, they're probably not going to respond. They are trained to do that. They get too much email and that's not their job. Whereas real estate agents publicly post their information. They do that on purpose and they are very used to responding quickly to people who inquire from the outside world, because that is part of their job. And quickly try to think about each criteria and each customer group. Give them a rating from one to 10, or if you want to do it based on the card metaphor, one through 10 or Jack, Queen, King, Ace. (laughs) And then at the bottom, I want you to add up the value of your hand or the value of your one through 10 total. One of those will have a higher number than the other two. And that's how we know this customer group has the most potential. And were probably going to be the most successful reaching out to them. You're right, you got it. Just answer the question of what problem does your product solve? Who has that problem? And once you have three groups, try to evaluate them on three criteria; size, pain to payment and accessibility. All right now, for the post-product people, if you are post-product, that means you already have an established customer base. You probably have people who are paying and engaging with your product. And you probably also have a lot of perspective leads and prospective customers that have already established themselves as being interested in your product. In this case, who do you interview? Well, it's actually relatively simple. Think about the product that you are proposing to launch or the feature you're to add, or the problem you were trying to smooth out in your existing product line. Try to think of it in terms of your user base, try to apply basic demographics. Are the older people that you're going to be targeting? Are they younger people? Are they married people? Are they people that have slow internet? Try to think of basic demographic data, use that information and then filter out your customer group and reach out to them. That one is infinitely easier because you have a big pile of customers and you just need to decide which ones are probably going to give you better information. Whatever problem you're trying to fix or

whatever initiative you're trying to run, probably pertains to some defined group within that customer base. And it's your job to pick them out and contact them. All right, guys in the next lecture, we're going to do a example of playing 3 Card Stud with a real idea. If you guys don't care, you think you got the idea, skip over that into step two, where we start talking about finding potential interviewees. All right, see you then.

Find interviewees externally

- Hey, guys, welcome back to the course. All right, we're moving on to step two of the customer development process, finding your potential interviewees. Now, you can obviously gather feedback about your product, or your proposed product, without actually talking to a potential, or an existing customer. In this lecture, we're going to talk about pre-product scenarios, when we're looking for new customers, or people that are not already aware of us to interview. Now, there are a lot of different places online or offline that you can go to connect with people that might be interested in your product, but for our sake, I've reduced it down to four different places, or types of places to look that consistently can find you people that are interested. Now, at this stage of the game, you know which target customer group you would like to talk to, because we played 3 Question Stud, and you identified one of the three as probably more worthwhile to go after, and one that you'll have more success reaching out to. Now, the first place, and, I think, probably the best place to start reaching out to potential customers that you can talk to is LinkedIn. I'm sure most of you guys are familiar with LinkedIn. It's an online social network that is primarily for networking. It holds a lot of information about the people that sign up, and a lot of that information is public. We generally get to see where people go to school, what their job is, what they've done in the past, where they live. We also know generally interests they're in, and groups they participate in. We also have endorsements that show us what skills they have, so that is a treasure chest of different ways that you can narrow down on exactly the target customer you have have in mind. Now, for those of you out there that have a LinkedIn account, and you have lots of connections, this

works extremely well. For those of you that have an account, but don't have a lot of connections, it still works very well, and for those of you that don't have an account, I suggest creating one, and connecting with some of the people you know. It really will only take you 10, 15 minutes, and by this time tomorrow, you can actually use it to find people. Now, the best way to use LinkedIn is actually just to use their search feature. Now, what the search feature is going to do is it's going to search based off of your connections, and then your connections' connections. That's why I say, if you only have a medium amount of connections, only 50 people, 20 people, it still works, because as long as you're connected to people that have a lot of them, then the search has quite a large group of people to look through. So, if you define your customer group by, let's say, their job type, this is perfect. You define it based off of their education, where they went to school, what location they live in, this also works very well. So, let's use the example of the real estate agents. If we were building a product that went after real estate agents, or appealed to them, all they have to do is in the search bar, just type in real estate agent, and right at the top, it says I can pick between people, jobs, and groups. Those are all very helpful, but I'm probably just going to pick people, because these are actually going to show me people that have it in their title, meaning right now they are a real estate agent, and even down here, it's already started to give me second-degree connections. Those are people that are connected to people that you are connected to, that fit my criteria, but I can click People, and it'll give me an extensive list of people. All right, so right off the bat, I have tons and tons of people, 10 different pages of people that I could reach out to. If I then wanted to add some criteria about where they live, I can use filtering criteria over here. So, maybe I only want to interview people that are near where I live in San Francisco, within 50 miles. All right, now I have a very good list of people that I could start to connect with. I could add in more criteria if I wanted to, but for our sake, let's not. Now, if I wanted to contact these people, the way that you would use LinkedIn is through one of two means. One is you can send them what's called an InMail, and InMail is a message that goes directly to their message system. Now, this is a paid option. I don't think you pay a lot of money for it. You probably have to do a subscription package, and it gives you a certain amount of InMails. If this is something you frequently use, then it's definitely

worth doing. So, the second option is to just connect right them, because when you connect with them, it'll give you the option of writing a message. So, click through to their profile, and then you can hit Connect, and in the message down here, I can write something along the lines of, "This is what I'm doing, and I would like to talk to you." We cover that in the next lecture. How do we get them to actually talk to us? If they don't see this, they still will probably connect with you, at which point you can then send them a direct message, and that doesn't cost any money. All right, so the number two place that we're going to cover are forums. Now, what do I mean by forums? Well, forums are generally just online communities that are all built around some kind of a theme. Pretty much any type of affiliation, or job, or interest you can think of has a forum, or an online community dedicated to it. And the reason forums work so well is because it's a bunch of people talking about whatever subject or topic you picked, and it's very easy to search them, or find people that have a problem that you are trying to solve, have experience in the area you're trying to go into. It's pretty much easy pickings, 'cause people will talk about what they're interested in, and they'll self-identify themselves. The first place I like to start is a website called Reddit. If you ever heard of Reddit, it's an online forum, and it's absolutely massive. It's an online community, and the way that it works is it's actually tons and tons of forums inside of one forum. Reddit calls these subforums subreddits, and every subreddit in this website is going to be based off of some type of theme, or topic, or subject, so if you can identify some of the things that your target customer group have in common, are interested in, or anything that they would affiliate with, you can find a subreddit for that. The best way of doing that is to search through their search box, and it'll give you a list of subreddits, or what you can do is actually search through Google, and if you use the code site:reddit.com, and then followed by searching for the topic you want, you can do it that way as well. Sometimes Google's actually better than Reddit. So, let's see how this works for the real estate idea. All I'd have to do is say real estate agents right there, and see if it shows me, and then right off the bat, we have three subreddits right here, and it gives you some gauge of how many people are in it. 41 people, 20 people, eight people. I can click Next, and I can look at other ones. These get more and more specific, but for the most part, I can dive into any one of these, start

looking around, start interacting, and then start reaching out to people. Another great option is a website called Quora. Now, Quora, spelled Q-U-O-R-A.com, is essentially Yahoo! Answers, but for adults. It's a website where people will ask questions, and then people who have expertise in that subject will answer them, and it's a website where people kind of trip over each other to give the best answer, but because so many people have asked so many different questions, what you can do is look for specific topics, or groups of questions, and then look at the people that are constantly talking in them, answering questions, interacting, and replying. So, I could search for real estate agents, and then right there, I can pick the actual topic. So, it's going to show me all the questions related to real estate agents, so all that I'd have to do is just jump into any one of these, and see if the people that are asking the question, or answering the question, are people that fit my target demographic. If they are, I can then message them, and try to make a connection. All right, the third place that I suggest using to find potential customers to interview is Twitter. Twitter is fantastic as a resource for finding people that might be interested in your service. The reason why it's so effective at this is similar to the reason why forums are, in that people are constantly talking on it, and people are constantly self-identifying themselves with things they're interested in, things they affiliate themselves with, and what they do for a living. There is also just a gigantic complaining-fest, which is great, because you can find people that potentially have the problem you're trying to solve, and reach out to them. Now, the way that you'd use Twitter is you can just go to search.twitter.com, and it'll allow you to just search through it. What you're actually searching through when you use this feature is what people are saying, and not specific details about the person. So, if I said real estate agent, it might not actually be as effective. Why, because real estate agents on Twitter probably aren't saying, "Real estate agent," over, and over, and over again. It might still work, but it's more effective to think about things they might be saying, and then search for that. So, a real estate agent might be saying something along the lines of, "Looking to buy a home," and then we can scroll down, and it looks like right off the bat, here's someone that says, "Looking for a home?" "Are you looking for a home?" Lots of people here are self-identifying, and the first one we captured right here is a man named Allan Todd. He works as a real estate

agent, see how that worked? Now that we know who they are, we can tweet at them, or we can try to follow them in the hopes that they follow us back, at which point we can send them a direct message. All right, and the last go-to place for finding potential customers is, it's actually your competitors, okay? 90% of the time when you're launching a new product, there are probably going to be competitors already there, and if they're not doing the exact same thing, they're probably doing at least part of what you propose to do, or they're just servicing the same target customer group. If you can identify your potential competitors right out of the gate, well, then it's relatively easy to just go to their social media pages, see who is engaging, and you can then figure out that those people probably are customers, or are interested customers for your competitor, and you can contact them. Let's say we were doing that real estate business. Never said what it was, but let's say one of our competitors was Opendoor. It's a relatively well-known startup. We can find their page on Twitter, and then just look for who is interacting with it. We can see who is liking their page, and there's someone who likes it, there's someone who likes it, and these are people that interact with their page. We can then go through these people's pages, and contact them in the same way. Now, this works the same for Facebook, this works the same for Twitter. If your competitor has a blog, you can also go on there, and find people who are commenting on their blog. That is also a goldmine for finding potential people to talk to. All right, so as of today, those are the four main categories that product managers are probably going to use to reach out to potential customers that they can interview. If you need more, or are interested, check out the resources. Otherwise, we're going to move on to step three, which is how exactly do you get these people to talk to you, all right, see you then.

Find interviewees internally

- Hey guys, welcome back to the course. So last lecture we talked about, where can you find and drill down on potential customers that you can interview, if you're in a scenario where you are pre-product or pre-launch. So the other side of the coin to this is, what do you do when you are post-

product? Let's say you have some ideas for things to add to a product you already have. You have a feature list, you have a roadmap, you have a backlog. The customer development framework tells us that we need to get out of the building and start talking to our customers but where do we find the customers? Now a lot of you at home are probably thinking, well, isn't this a little more obvious? And you're right, it is more obvious. Presumably, if you're post-product, you have a list of customers, a list of free users, and potentially a list of people that have demonstrated interest in your product. Even if you're a relatively new and you only launched not that long ago, you'll still have what are called early evangelists, which are people that use it as if it were a version for polished product, and they just love it. Regardless of where you are in this product development process, there are customers, you can interview. Now, the most obvious thing is that why don't you just go and look at the list of customers and just go that route? Now you can do that, you can just tap someone on the shoulder and say, "Hey, I need the list "of all the emails of all of our users "and let me contact him." Well, there's two downsides to doing that, one is that it comes off very spammy, and it actually can break your terms and agreements. So you need to make sure that that's within the bounds of spam. The other thing is that you're going to have a harder time getting people to talk to you just because this is a cold email. This is as a cold as it would get, you're just emailing them 'cause you saw their email on a list, it'd be hard even to customize it for them. So you can, by all means just send out an email blast to your user base and you will get interviews that way but in this lecture, we're going to focus on four ways to get warm leads that will save you some time with regards to finding them and getting them to talk to you, and is probably going to get you better customers to talk to. In general, some of the best customers to talk to are ones that have already pointed themselves out as people that are enthusiastic or interested or have a problem with something in your product. People that have opinions and people that are invested in your product enough to have one of those two emotions are going to give you a much bigger bounty of information then the first way of going about this is by using your company's live chat feature. Now a lot of companies, startups, even large businesses use live chat systems. Live chat systems are exactly what they sound like, you can chat with someone in the company, live by

clicking on a box, usually in the bottom right of your screen. Whether it's Olark, or tawk.to or one of the more complicated ones like Intercom.io, if you have them, and then you have a wealth of information from which you can choose ideal customers to reach out to. So if you have a live chat system, great, continue on. Now there's two ways of going about this. One is the passive and one is the active. In the passive approach, what you want to do is essentially go and ask to see the chat logs from your customers and what they're saying and what questions they're asking. It's really interesting stuff on its own but what you can do is you can look at what people are saying and try to find individual people that meet your target demo. Then when you message them you can pull information from that chat log, make it sound like a continuation of the conversation, people are much more likely to respond to that and you get plus points for customer service. Now, the other approach is the more active version. A lot of live chat systems, the vast majority of them have the functionality that allows them to force themselves on your screen so you might have seen this before, if you ever run a website, and you have something pop up as you go to another sub page, and it usually says something like, do you need help finding something, you say yes or no? You say, yes, it automatically connects you. If you have that functionality, you can definitely find people that appear to be doing things that you think would fit into your target demo or you can look for specific users when they get online or you can just do it at random. When you force a conversation, then you can pop up, you can talk to them, and go that route. Alternatively, if you don't have that capability, that's fine. A much simpler. version of this active approach is either to set in on the customer service live chats, if you're in a small organization, this is very easy. Or ask someone who is in customer service to at the very end of their chats, once they've solved the problem, just simply ask them if they would be willing to talk to someone in product management role for 10 minutes. It's very similar to when you call the bank and they say, after your phone call, would you like to take a five minute survey, yes or no? A lot of people say no, but a lot of people say yes, so don't ignore the live chat, this is some potent stuff. These people have self identified themselves as being engaged with your product. The second treasure trove, where you can go to get people to interview is your blog. Now, most companies have blogs. If you don't have a blog nowadays, it's kind of

the exception. It's kind of standard practice, it seems like so if you have a blog, a lot of blogs allow commenting. So it's rather simple. Look at the people who are commenting on your blog. If you don't have a blog, look for people who are commenting on your outside external press. Those people have something to say about your product, they're probably users and also you have this advantage. They want to be heard, okay and that's the reason why they're posting on your blog to begin with, they have something to say, no one post a comment, it's entirely passive that says nothing. They always want to be heard and if you message those people email them and say, would you like to talk for 10 minutes about the product? They will feel special and it will convert. Now a third area to consider are power users. So we previously established that you probably have in a list of all of your customers and if your organization even slightly knows what they're doing, they're probably collecting information on each user, so all you have to do is look specifically for power users. Now you can segment power users by a number of different things. You can look for them based off of frequency, how often do they use your application? And that's generally a very good metric. People who are only casual about your product probably don't log in very often and they probably don't use the features very often and these are very easy things to track. Be very surprised if your organization was not tracking them. You can also look for things like high engagement and every organization has a different definition of engagement. It could be buying things often, it could be sending messages often, it could just be people that use it frequently and message your own company often. So don't neglect power users. They are in general, more informed about your product, they spend more time using it and here's the catch, they're more invested in your product so if you say to them, let's have a conversation about improving it, there's a very good chance they'll be interested in that. And the last one is Twitter. I love using Twitter for talking to customers, because it's fantastic for that. It's just an online machine for conversations. Twitter is a goldmine for reaching out to these people and it's really simple. You guys probably have a Twitter account, just look at who is tweeting at you, who is replying, who is liking and who is sharing your posts. Chances are they already follow you, which means you can direct message them, those messages get answered almost all of the time. So they've essentially self identified

themselves as saying, "Hey, we like your product enough to spend time "talking about it or engaging with you." They have their foot in the door, they're probably going to talk to you. Okay, guys, so those were four different ways that you within an organization and with a pre-existing product can pick out customers who are most likely to respond to you and give you better information than if you pick someone at random. So use those four tricks yourself, live chat, commenters, Twitter and look for power users. In the next lecture, we're moving on to step three, which is getting them to talk to us and it's a very weird dance. Alright, see you then.

How to get them to talk

- Hey, guys. Welcome back to the course. So now that we've identified who you want to interview, we're going to talk about how do you get them to talk. Now a lot of people think, "If I just message someone out of the blue, why would they respond to me? Why would they give me their time just because?" So in this lecture we're going to address that. We're going to go over three principles to live by when asking to interview a customer and some two bonus tips that will put you over the edge. To those of you that nodded when I said, "Why would they actually contact us?", well, you know, let's talk about it. It's rather obvious that if you're cold-emailing someone, even if you use the techniques that we covered in the last two lectures, it's still a relatively cold email. They have no personal connection to you. And we all know how we treat emails from people we don't know, right? You probably don't remember them, they go to Spam, or you delete them faster than you could even notice. The thing is, in general with cold-emailing, you have to think of it kind of from the perspective of investment. If someone doesn't know you, they may be willing to read your email, but they are not going to be willing to invest the time to read the entire thing. If they're talking to a friend, and a friend, let's say, emails them, they'll probably read a page, right, for spacing out. But if someone emails them, they have no idea who you are, and they're not referring to any previous conversation, they might give you a couple sentences. This can be overcome, and it's relatively easy. So let's talk about three things to live by, three tenets of cold-emailing. If

you follow these three, you will have a dramatic increase in the number of people that respond. Side note, though. To put this in perspective, we generally have a ratio three to one for use to reference how many people you contact and how many people respond. Obviously, this is a generalization, but set your expectations like that. If you message 10 people and three people respond, don't get mad at the other seven. It's just the case that you generally have to message three times as many people that you want to interview. And also keep in mind, let's say you want to interview 10 people, how many people should you message? Probably 30, maybe more to be safe. Anyway, enough of that. Let's back to the three main tenets. Okay, so the number one tenet when contacting someone in a cold connection type situation is be short. You probably figured this out 'cause it's just talking about how people aren't going to give you a lot of time investment, so being short is incredibly important. It's one of the biggest mistakes I see people making. They'll message someone that they want to contact, and they'll send pages and pages. This happens to me all the time. Somehow people get my email address and send me something, and it's three pages long. It's ridiculous. No one under any circumstance is going to read that. So focus yourself on brevity. Now what do I mean specifically when I say short? How short is short? Should it be like a tweet, or is short like four paragraphs? Short generally means closer to the tweet than the five paragraphs. I like to think of it in terms of four to seven sentences. If you can get it away in four, that's great, but seven is the absolute maximum. Anything over that and you'll see response rates just plummet. Number two, be personal. People do not want to be emailed from a robot. We don't like talking to robots. If I even sniff, or have the tiniest little inkling, that this is an auto-generated email, that dramatically lowers the chances of me responding to it. Now you're not going to be messaging millions of people, or even thousands of people, so there's really no excuse for not adding at least a tiny bit of personalization. The most effective way of being personal is by responding to how you found them, okay? If this is a person you found on Twitter complaining about something, you need to immediately make a connection to that. Say that you saw that they had this problem, you recognize that it's a problem, and then you go into the ask, asking them to talk to you. That email cannot possibly be auto-generated. People are much more likely to respond if they actually think

you typed it out. And you don't have to type every single one out. You can make a semi-template where you're only modifying one or two sentences of the four to seven, but you have to write those one to two. If you found someone who wrote a blog piece and you're connecting with them, then you need to say something about their piece that you had a connection with, that you responded to. Point out some kind of connection between something they mentioned and something you've experienced yourself. And the last tenet of cold-emailing is be valuable. Yes, you are correct in saying that everyone out there for the most part will consider their own self-interest when answering your email and thinking about whether or not they want to spend time talking to you. Now I'm not saying that you have to bribe every single person that you talk to, but you need to show that having this conversation with you is valuable in some way to them. Now that could be abstract things, as simple as "It would help our organization out." A lot of people want to feel altruistic. They want to feel like they're helping. That ultimately is valuable to them. Telling them that you value their input, and there's a high likelihood that you'll act on it, makes them feel like they actually have an effect on the organization. It's not like voting for the president. This actually matters. And yes, of course, you can offer to incentivize them, and that's something that you're almost always going to try at some point. It is very effective, and it is something you can use to quickly get people to agree to talk to you, but it's not the case that you have to grease the wheel for every single person. You can appeal to their better self. Okay, got it? You open up that email client, you better make sure you are four to seven-sentence short, short, short, short. You better make sure that you sound personal, not like a robot that just auto-generated this. You need to connect it to something they did or something you've experienced that is similar to them. It needs to be customized, and it needs to be valuable. Why should they spend time talking to you? You can butter them up with money and gift cards if you want, but you can also just appeal to the fact that they could help you out, and a lot of people are okay with that. Now before you run out there and just start willy-nilly emailing people to see if they'll talk to you, there's two other tips that I want to give you that will really put you over the top as a product manager. Two other things that I really insist on trying to fit into cold emails is, one, make sure that you mention that you're not from sales. That is an

amazing advantage to you. I would say 60% of people that don't respond just think that it's, because if they talk to you, you're going to try to sell them something. So if you say you're not in sales, you're in a different department or a different role, the title of product manager is very innocuous. No one is going to think that at the end of this conversation you're going to pitch them on a timeshare condo in Florida. This is an advantage that is specific to you. And even founders of companies can't do this. They can't say, "I'm not in sales," because if they're the ones emailing people, it's because they don't have any salespeople or have product managers. And founders do a lot of selling, so it doesn't really work that way. The second thing that will really increase your chances of connecting are make them feel special. You really need to emphasize that you want to understand their specific problems and build products that fix their problems, and that they can help you get there. That sort of Neo-you-are-the-only-one type kind of conversation makes them feel like, "Well, I'm the only one who can actually help them. This is really a waste if I don't." Alternatively, you could make it sound like the reason why you're reaching out to them is because they have done something right. Give them the VIP treatment. Make it sound like, you know, "We listen to you more than we listen to other people." So make them feel special. Make it sound like you are giving them special treatment. They have information that is more valuable than anybody else. They will melt for it, especially if they're power users. All right, guys, we've covered the concepts of how to get them to talk to you. In the next lecture, let's actually start doing. I'll open up an email client and show you how these three principles, two tips, translate into actual email communication. All right, see you then.

Practice writing emails

- Hey guys, welcome back to the course. So last lecture we talked about how do you reach out and get these potential customers or customers who you already have to connect with you and give you feedback, it's a bit of an ask to get someone either on the phone or in person. So you need to have some skill with it. So let's get some practice in the actual realm where you're going

to be contacting them your email clients. So I just kind of whipped open this email right here and we can use this as a practicing space. All right at the bottom, remember, the three things short, personable and valuable. We need to be short four to seven sentences, less is more personal. We need to make sure that whatever we say to the person is specific to them. Doesn't feel like a robot is saying it and valuable. We need to highlight that they can get some sort of benefit from talking to us or building off of what we set in the last lecture on how in general people are probably going to read only one or two sentences of what you send them says is a cold email, we need to front load our emails and make sure that whatever we write that's personal to them, we make sure that at the beginning, you can make a crucial mistake, by using your template at the beginning of your email. That's what they read. They think it's an auto system sending it to them and they bail. So the first thing you want to do is write one to two sentences of an introduction. Now, when I say introduction, I don't mean introduce your self necessarily. One of the things that people generally dislike is that in the first couple of sentences, they hear you talk about yourself and not them. That's another crucial mistake people make when reaching out to people, they make it all about me, when in reality it needs to be about the person who's receiving it because they're the ones that are you know, you're asking them to respond. This is where we get personal. And whatever you write here, it needs to be specific. I read your blog post on blank, it was great. I've noticed that you are struggling with x feature and our product, I saw that you had a problem with this, a lot of people have that problem and I'd like to help you. The next thing you want to do is generally write two sentences of why you want to talk. Now this is your opportunity to insert the value portion of your email. Once they get to this section, this is where you kind of hook them in and give them an incentive to actually respond. Now to classic go two ways of adding value in these cold emails is to do one of two things. One, you can say, you know, I think I can help you use our product better in the case of a post-product customer or in the case of a pre-product potential customer, you say something along the lines of how you specifically want to solve their problem. Those always seem to be the go to sentences you see this pretty much everywhere. If you've ever received an email, that probably said that I would like to solve this problem for you. Or I can show you how you can get

more out of our product. Alternatively, if you didn't want to use those two, you can go a different route. Now there are three different avenues or paths I think you can use that are quite effective. One is instead of just saying generically you have a problem I want to solve it. You appeal to pride. Pride is a very, very useful thing to use. When you want someone to respond to you. What I mean is you have the opportunity to say things like you appear to be an expert on Blank you would be able to give us so much valuable information. Everyone likes to hear that someone has recognized you know more information or you are better in some way than everybody else. If you want to feel the pride of someone who's an existing customer, you could say things like, you have been a customer so much longer than all of our other customers, or it seems like you understand our product much better than other customers. And that relates back to what we talked in the previous lecture, make them feel special. That is one technique that is always open to you as a product manager, and it is very effective. The Second Avenue you can go down is appeal to money. You can use this section, this two sentence section to say why you want to talk and then incentivize them by saying that there's some sort of monetary reward at the end of that rainbow who doesn't like money? The third thing you can do is imply association. So what I mean by association is, you can imply that you know people or well situated in a space that the person you're emailing cares about. Oftentimes, what you can do is you can look at their LinkedIn profile, see what space they're in. And you can imply that by connecting with you, you know, a lot of other people that in itself is valuable. Some people do that some people are more than willing to talk just because they think you're a valuable contact. After that we have one last thing that you want to do. And this one is amazingly crucial. The last thing you want to do is schedule a time now, do you say are you willing to talk? No, do you say, do you have any time next week? No, what you say is, are you available Wednesday at 3:30pm Pacific Standard Time? What you're doing here is you're reducing the friction in between them responding and talking to you. If you say next week and they're interested the next time they'll say yes, which day? And you'll say how about Tuesday, and they'll say it great. Propose a time it just goes on and on went from one email to six emails if you just pick a time and say it, they're much more likely to say yes. And you don't run the risk of losing them throughout that process

of trying to schedule them. Don't forget that one. You probably seen that technique in your inbox. Another pro-tip you can use in conjunction with this is use some of those online scheduling software's like Calendly there's a ton of them where you don't even have to suggest a time you just let them click through and pick a time. All right, so let's look at a well crafted email that follows that same template we just went over. Alright, so it says hi, Sam, I read your article on volunteering your professional skills in Guatemala. It was really inspiring I'm looking to travel more and you've got me thinking about incorporating volunteering when I do. So right there that is two sentences. It's personal. It connects it to his article on volunteering in Guatemala, you showed that you read their article, and you understood it and you were responding because of it. Now, the next sentence is what we call the value section. It's just divided for purposes of reading it. It looks like they still used two sentences. But they said, I have a software company trying to improve remote medical record coding, and then it says, I'm not looking to sell anything. But since you have so much expertise with remote coding, I'd love to get your advice on our products, so we don't build the wrong thing. This is a masterful set of sentences. Whenever someone sees I have a software company, dot, they usually cringe but if it's followed by trying to improve remote record coding, which is something you do and probably are frustrated with, it instantly sounds like this could be something that is game changing. Then they move on and we use that technique where they said, I'm not in sales, I'm not trying to sell you something. And then they appeal to their pride by saying you have expertise so what's the value of responding to this email? Well, it is the fact that one you can help them, two we have a specific expertise that would benefit them more than talking to someone else. And three this is the big one, they're building something that could actually solve your problems. And then the last one is, if you're available, I'd love to chat for just 20 minutes Thursday or Friday morning. You know what I'm going to complain about in that one, they didn't specifically say a time or a set of times, but that's pretty good. You also notice that this came in at six sentences, so it's very well within our four to seven guideline. Alright guys, go out there and shoot off some emails in to the wild. Just make sure that you follow this format or at least you stick to the core principles of being short, personal and valuable. If you want to use this template, start with an

introduction then add value, then schedule them, make it short, make it valuable. Alright guys see in the next lecture

How to run a customer interview correctly

- Hey guys, welcome back to the course. So in this lecture, we're going to talk about what are some of the best practices with regards to talking to your customers and running customer interviews. Now you need to talk to your customers in order to make and create products that is unavoidable. But in order to collect the right information and get the right signals for your product, you need to know how to talk to your customers, and what not to do. Now, customer interviews are not as intuitive as you think. You might think that the point of a customer interview is to just sit down, have a conversation, talk to them about your product, see how they react, pitch them on whatever new feature or idea you're rolling around in your head. Now that's not even remotely true. The only thing that was true in that is that you are sitting down with the customer, I would hardly call it a conversation. It is an interview. Now, what's the actual purpose of running a customer interview? Well, it's to get the most real and accurate information as you can from your customer. If you just sit down and start rattling off pitches, and if you domineer the conversation or act the same way as you would in any other normal conversation, you will not get accurate or real information. So I'm going to go over four applicable best practices to help you understand how to run a customer interview and what you should do. Now first of all, let's tackle the misconception that when you run a customer interview, your topic is going to be your product. It's actually not. When you talk to your customers, you want to talk to them about their problems and their needs. You rarely, if ever will talk about the solution. If you're talking about your solution directly with the customer, the information you're going to get, it's going to be limited and it's not going to help you understand the why, the deeper level of why they make certain decisions, why they think a certain way. There's a great quote about talking to your customers. And that's customers might not know what they want, but they can't hide what they need. So when you talk about the solution, you're talking about what they

want, and they often don't know what exactly they want or how in an accurate way to process that information or describe it. But their needs are as simple as it gets. Anyone can talk about what problems they have or what they need. And we care because those conversations are much, much richer than a conversation about the surface of a solution and whether or not it looks nice. The second thing you need to understand is that, you are here to ask questions and not really direct conversation or give your own opinions or your own thoughts. In general, the ratio between how much a customer is going to talk and how much you talking on in these interviews, it should be 95% or 90% to 10% or 5%. It is very, very skewed. You want to set this up so that you ask one question and that question leads to your customer talking about multiple things in detail. The point here is that you want to get them to talk. One of my favorite rules about customer development or customer interviews is the point is to get off point, Basically what that means is that, don't worry about whether or not they're answering the question the exact way that you expected them to, or going on to the topics that you wanted them to. You have to embrace the idea of tangents. Sometimes you'll ask a customer a question, they'll answer it or they might not. And then they'll go off on their own and start telling a story or an anecdote or something that's entirely not related. And you got to understand that tangents are good. Sometimes they're even gold wines. Sometimes the best ideas come from your customers going off the cuff, off the record, saying things that you never would have thought to ask. The third thing is make sure that you create an environment where they feel comfortable talking. That means creating a rapport with them. Don't run a customer interview where you just sit down with them and immediately start asking them detailed questions. Make sure that they are comfortable with you, make sure that you at least feigned some effort that you are interested in them outside of this interview. Make sure that you're comfortable in conversation before you actually start. Along those lines don't ask them questions that would be hard for them to answer in person, or make them feel awkward or nervous to answer in person. Also, when they give you negative feedback, don't react negatively, don't defend the product or the idea that you were projecting. When they give you negative feedback, they're testing you. And if you respond negatively, I know the chilling effect on the rest of the interview, they're much less likely to give you

truthful, honest information in the future, because they don't want to have the confrontation or the argument. When you respond in a neutral non judgmental way. They'll then realize that it's okay to tell the truth. And the last thing is don't force these conversations. Try to guide them, depending on what type of interview you're doing, whether it's exploratory, validation, satisfaction or efficiency. You probably have a focus in mind, you probably have some questions you want to answer and some topics that you want to spend time on. But don't be forceful in forcing them on to that subject. If you allow them to gray use in areas that they actually care about, and then naturally guide them into the direction of what you want to talk about, you'll end up in a situation where they give you a lot more information and they feel a lot more comfortable. You also don't want to risk the chance that in the process of getting to the topic that you wanted to talk about, they could have brought up many, many interesting things you never would have thought of. So if you keep those things in mind, you will have a much better chance of running a successful interview. And if you get to a point or you're having trouble talking to the customer, you don't know what to ask, you don't really know what's going to get them to respond to in a more, open fruitful way. Use this one that I use all the time, my favorite sentence anytime I'm stuck. I don't even know what to say is, that's interesting tell me more. And if you ever find yourself lost in this process, not sure what direction is up and what direction is down, just keep in mind the five W framework. I'm sure you've heard of it, but you probably haven't heard of it used in the context of product management, which is what, why, who, when, and where. That is, you are always as a product manager striving to answer the five questions and get a better deeper understanding of what they mean for your product and your customer. And this questions are, who are your customers? What are their habits? When do they use your product? Where do they need your product? The last and most important one, why do they need your product? Customer development and customer interviews are really just a process of filling in the gaps of your knowledge of the five W's. the more conversations you have with your customers, the more you'll be able to fill in the five W's and have a better more holistic understanding of what your product is, its impact, and why people are using it. All right guys, now that you know some of the bigger picture of what makes these interviews

successful, what are they like? What are you thinking about and how to keep your eye on the prize. We're going to talk a little bit more about the specifics, or we're going to talk about good questions and bad questions. They make a huge difference in your interviews. All right, see you then.

Good questions, bad questions

- Hey guys, welcome back to the course. In this lecture, we're going to talk about good questions and bad questions. Now anytime you talk to a customer, you run the risk of accidentally biasing the information they give you, causing them to lie to you, or the information they give you is hard to understand or, worse, misleading. A good question can bring out so much out of a customer and it can fill in so much of your understanding of your product and how to improve it. Bad questions, on the other hand, are very pernicious. Sometimes you don't know if you're asking them and worse, sometimes you don't know if you're getting bad answers and incorporating that into your product decisions. Let's simplify it. You ask good questions, you'll make good products, you ask bad questions and you're going to probably make a bad product. So how should we ask questions to our customers? Well, there are some rules that are worth following to make sure that you're probably asking good questions and not the bad ones. The number one rule is always ask open-ended questions. An open-ended question is a question that cannot be answered with a simple yes or no or a specific piece of information. When you ask someone an open-ended question, it gives them the latitude and the room to answer it and to give the relevant information they see fit, which is exactly what we want from them. If I just ask someone, "Is it raining outside?" all I get is a yes or no answer. It's one tiny little data point. But if I ask someone, "What do you do on rainy days?" that could be answered in a million and one different ways. Who knows, they could talk for an hour after that. And you remember from the last lecture, we talked about the fact that we want people to go on tangents, we want them to explain, that's where we get the good stuff. So that one's pretty simple. That's what you should do, ask open-ended questions. If you do that,

you can't really fail. So rule number two is where we start with the things that you should not do. Rule number two, don't ask binary questions. What do I mean by binary questions? An example of a binary question might be, "Have you heard of Uber?" That question can only be answered with yes or no. Binary, there's only two options. You don't do that because one, it doesn't draw them out, it doesn't entice them to tell you more information or feel more comfortable with you. And the second reason is that you're just robbing yourself of information. They could have gone on and on and on about a subject adjacent to that question, but instead you asked them something that just ends with a hard landing. And rule number three is don't ask hypothetical questions. Now I know a lot of people like to ask hypothetical questions, they get ridiculous, and they're things you just should not do in a customer interview. An example of a hypothetical might be, "If you were a painter, would you buy "a subscription service for canvases "if it was reasonably priced and the canvases "were high quality?" That question is absolutely ridiculous, and I wouldn't be surprised if I saw someone ask this in a customer interview and I would shake my head vigorously. Hypotheticals fail for a number of different reasons, but the biggest reason is that people operate and often are illogical in fictitious environments, hypothetical environments. And if you bring those questions down to earth, they will actually give you information that's easier to understand, easier for them to connect with, and it's going to be more accurate. Rule number four is don't ask leading questions. Anybody who's watched Law and Order, you know what a leading question is. A leading question is a question that biases or influences the person that's supposed to answer you. Leading questions are often questions that sound more like they're just telling you something. An example of a leading question might be, "If you could save 40% on your auto bill, would you be interested?" Yes, I would, GEICO, but let's be honest, who is going to say no to that question? It's almost rhetorical. If I just ask you plainly, "Would you like to save money," you'd probably say, "Yes." So I know the answer to that before I asked it, so why am I asking the question? Leading questions get the answer that you want, but the answer you want might not be the truth. Rule number five, don't ask questions that might make the customer lie. And I mean lie in the sense of questions that put them in an awkward position and will unduly influence them to give a certain

answer. So for instance, if I ask them, "Have you ever cheated on a test?" That might be something that they might not want to tell me or questions, let's say, of a personal nature. Another example question would be, "Do you think I did a good job with this design?" It's very awkward sitting next to someone, especially someone you don't know, to tell them what they made is terrible. It's the reason why you don't tell your kids that their kid art is really quite bad. If they lie, then the data's going to be junk, but more importantly, if you put them in a scenario where they could lie, you'll never know if they did and so it doesn't even matter, because that ambiguity makes your data, makes your information pretty much junk. So that's it. Focus on open-ended questions and get the most information out as possible and make your customers feel as comfortable as they can. Don't ask binary questions, don't ask hypotheticals, don't put them in awkward situations where they might lie, and don't ask leading questions. There's really no excuse for breaking some of these rules. Bad questions can almost always be transformed into good questions. "Have you heard of Uber?" You could easily just say, "If you need to get across town, "how do you do it?" That'll give you way more information and if they do use Uber or have heard of it, they will mention it. "If you could save 40% on your auto bill, would you be interested?" should be changed to probably multiple questions, but you could ask a question that's better by saying, "If you could save 40% on your auto bill by," and then explain the downside, then ask them if they would do it, that's a better question. Maybe you save 40% because you have to car pool with everyone every day or maybe it's 40% because you can't drive on Fridays or Saturdays. I don't know, give them the whole picture. The last thing I'm going to leave you with is my favorite tip for asking good questions, or at least if you ever get stuck in a situation where you don't know what to ask, always use this one or have it stored in the back of your head, ready to go. It's, "That's interesting, tell me more." "I hate your product." "That's interesting, tell me more." "I'm not sure if I would pay for this." "That's interesting, tell me more." "Why are you blocking the exits?" "That's interesting, tell me more." All right, guys, see you in the next lecture.

Build user personas off your interviews

- What's up, everyone. This is all about building user personas. And you can build them from the interviews that Evan just talked about. First, I want to say this is a bit of a buzzword alert. Because what I'm going to talk about in this lecture is of course user personas and how to make them. But I want you to know that if you go to many companies and become a product manager, they will likely already have a good idea of user personas. And you will likely not be the one making them. If you do have to make them, it's something you will do with the design team. In any case, understanding what a user persona is and what they're used for is really useful to being a product manager. And it's something you need to know. So have you ever heard of the term user persona before. If not, you will when you become a product manager because designers are talking about them all the time. User personas are just aggregates of observed user behavior. So let's say that I'm a product manager at eBay. For those of you that don't know, eBay is an online auction website where you can buy and sell products that people put up for sale. As a product manager at eBay, if we talked or interviewed a whole bunch of users, we would likely find out that there are groups of certain users that behave in similar ways. For instance, there are people who shop once every few months on eBay for a certain type of collectible item. There are also people that come to eBay to look at reviews and prices on certain items but very rarely purchase anything. And then finally, there are probably some heavily engaged users that log in every day to look for a specific item to come available. And they snatch it up really quick and try to resell it. Then on the other side, we have the seller side. So these types of users are sellers on eBay. And there are groups of them as well. So people will sell something once every few months when they need extra cash. And then there's another group of people probably who use eBay all day, every day and make a actual living off of selling items over the internet that they purchased in person. You could imagine that if you were building a product or feature that is targeted at a specific user behavior, it would be really convenient to have a user to refer to as an example, rather than just say, I don't know, this feature is targeted to all people between the ages of 25 and 40 who sell a lot of stuff on eBay every day and make a living off of our site, right? This is where user personas come in. To make a user persona, we interview or observe a large number of

users. Then you find a user behavior that is observed frequently, and pretend that it is one fictional user doing that thing. Then you name that fictional user with a real name. You give them a description like how they like to use your product and what matters to them. And then you even give them a bit of background information. You're actually kind of creating a fake person here. The purpose of user personas is to make it more convenient to talk about certain user behaviors that you're building for. But also to increase empathy towards your users. If you're looking at a bunch of metrics on a screen and see a whole bunch of numbers that are describing a certain way that a large user base behaves, it's sometimes hard to keep in mind that these are real people out there in real life doing these things. User personas make behavior and metrics more relatable as a human. All right, so head on over to the next lecture. And we're going to go over what an actual, real-life user persona might look like.

Real-world examples of user persona

- Hey, everyone. So just so you can see a true, real-life example, here is an actual user persona that we use at SoundCloud. Her name is Becky and she's a mainstream music listener. So you can see that we have her name. We have a little bit about her, like what type of phone she uses and what type of music listening habits she has. And then, we have some preferences on how much she likes technology or what types of listening she really likes to do. So if we were doing this with our eBay example, we would do something like this, but we would focus it on selling things on eBay rather than listening to music. For instance, we may want to make a persona for users that find cool stuff at local garage sales and sell those items on eBay for a living. We could call this user persona Jessica, maybe, and say that Jessica is typically between the ages of 20 and 50, loves antique shopping and garage sales, and relies on eBay to make a living. Most important thing to her could be that she is able to quickly list multiple items that she found at antique shows on eBay and then manage all of those multiple listings at once, easily from her computer. So that's an overview of user personas and like I mentioned, UX designers are going to be the people that are making these

most of the time at your company, but you need to be very aware of these people as a product manager. When you make your user personas, make sure they have a name, a sketch or a photo, and a description about what is most important to them about your product, what they want to be able to do easily, and anything else you think is useful. All right, that's it for this one. I'll see you all in the next lecture.

The product manager and the data diet

- Hey guys. Welcome back to the course. This lecture, we're going to talk about a concept in product management called the data diet. Now, what do I mean by data diet? Well, what I mean and what I'm getting at is that when you are a product manager or an entrepreneur, you do not make decisions in a vacuum. You also are not going to make product decisions with only one single source of information. Let's say that you wanted to go see a movie. Would you be satisfied in just pulling people over randomly in the street, asking them what they thought of a movie and then basing your decision about whether or not to see it off what they said? Well, no and if you said yes, that's because it's a low-risk maneuver or you have really low standards. Shame on you. Building products compared to going to movies are a little different. Building products are a much bigger risk. When you're building a product, there are no do overs and if you make a mistake in building your product, that can haunt your company's legacy for years. That movie in this terrible metaphor would instead last let's say a year or at least one of those ridiculously long four-hour movies and you can't leave to go pee. So you better figure out for sure whether or not you're watching "The Shawshank Redemption" or "Eternal Sunshine of the Spotless Mind." God, that was awful. When the stakes are that high, you probably will look for other sources of information. You might ask your friends or your family, people you trust. You might read reviews online. You might even preview it by watching the trailer. You get the point and the same goes for product decisions. When you make a product decision, you're going to be using a mix of different information sources. You might use internal feedback from your company. You might use data you got from user testing. You might use

feedback that was submitted online. You might use the analytics and the engagement stats from visitors on your page. You'll even base your decisions off the news and market trends and watching what your competitors do. There are a lot of different inputs. Customer interviews, and everything we just learned in this section are a crucial part of making decisions as a product manager but they are only one dish in your data diet. Customer interviews are great but they do have a pair of vulnerabilities and that's why it is crucial that you supplement this information with other sources. The first vulnerability is that customer interviews, they don't really scale. It's only you talking to a customer. You can have one, maybe you can have 10, maybe you're crazy, you have 50 interviews. That's a lot but you could be working with an application that's thousands of users. So it's actually quite risky to take three or four interviews and extrapolate that for the entire group. Who knows if those three people you talked to were the crazy ones. The second vulnerability is that this qualitative data and qualitative data is very important but you also need quantitative data. Qualitative is the fluffy stuff, the why, the what. I like this, I like that. Quantitative is the data, it's hard numbers. 0.25% of people clicked on this button. People stayed on your page for four minutes and 25 seconds, et cetera, et cetera. Qualitative data is nice and it helps you understand but it's kind of left out to dry because it's not precise enough and that's why you need quantitative as well. Think of qualitative information more like a barometer of temperature and quantitative data like an actual thermostat. So keep that in mind. You have a data diet. Customer interviews are just one piece of your dinner meal. You as a product manager will have to make hard decisions about what inputs to use in your decisions, which information sources are relevant and which are junk. In the rest of the course, we cover a lot of these different input signals and we also cover in more depth how you can synthesize this information and make good judgments for your product. All right, guys, see you in the next lecture.

What is an MVP?

- Hey guys, welcome back to the course and welcome to the new section. We're going to talk in this section about MVPs. First off, we're going

to talk about MVPs, what they mean in definition and how the lean startup framework wants us to think about them. Then after we're done talking about the theory, we'll talk about how do MVPs get used in actuality, by product managers. Now, MVP, if you don't know, this stands for minimum viable product. The term was originally coined in a book called The Lean Startup Framework written by Eric Reese. And, for the most part, it's extremely popular. Startups nowadays often follow this religiously, and even big companies have taken a lot of the core concepts and integrated it into their workflow. All right, let's look at the actual definition of MVP, as the book says it, pull it up on the screen. You guys can read it. (Jeopardy thinking music) That was ridiculous, and I'm sorry for how long that definition is, I don't know why it's so bloated. MVP just means the smallest amount of a proposed product that you can build, push out there, and then get real feedback from users about whether or not they are actually interested in it. MVPs are primarily designed to test hypotheses and test assumptions that you have. Now, if you want to be more accurate, you would call an MVP an MVP experiment because actually, an MVP is a process, not necessarily a thing. And that process is very much like a science experiment. You'll start with things like assumptions, a hypothesis, a minimum criteria for success. And then you'll go about testing your hypothesis in a very scientific, measured way. MVP experiments rely heavily on a concept called validated learning, and they draw the distinction between learning and validated learning. validated learning is anything you learn from a customer that's done in a test environment in a way that, you know, for a fact, you're not biasing your customer. And you're studying them in an environment where they have no undue pressure and they act naturally. Learning, on the other hand, is just pretty much anything else you do that's not designed like an experiment, not done well. And you can't trust the results you get from it. Think of MVPs in the metaphor of walking in a park. You're on a path and you reach a fork in the road, forks, three different ways you can go and you don't know where to go. Why are you doing this? Why do people hike in general? Probably because they love trees. Now in this metaphor, you have to pick which path to go down and I'll give you a little bit more information. Right at that intersection, there's a town. And this town has people and these people presumably know which path is the right one to

take. So how do you decide? Well, most people would say ask someone in the town, but actually with an MVP experiment, you don't want to ask the townspeople. Why? Because even if they swear that they know which path it is, we don't know for a hundred percent certainty that they have any idea what they're talking about. Can you really trust those townspeople with their quaint way of life? No, you can't, because it's entirely subjective and qualitative. So, the only way to actually tell for sure is to walk down each individual path until you can see enough of the path to know if it's the right direction. With an MVP experiment, it is the art of walking down those paths, putting in the minimum amount of effort to get to the point where you can say, "No, wrong path." If you're going to build a website that sold shoes online and you weren't sure if people wanted to buy your shoes online, what would you do? Would you go all in, build out a fancy website, hire some people to program it, buy some inventory, which is, this is the equivalent of just sprinting down one path. Or would you try to do the absolute minimum just to figure out first whether or not this is the right path and whether or not people want to buy your shoes online? That's actually a classic example of the MVP. If you guys are familiar with Zappos, the way that they started was exactly what I'm talking about. They wanted to know if people would buy shoes online. And this is back in the day when buying shoes online was kind of crazy. So what they did is they made a very basic website. They didn't buy any inventory whatsoever. They walked across the street to a shoe store and took photos of the shoes. And then when people bought them, they physically walked over and bought the shoes and then shipped them. Once that had happened enough, they knew this is the right path. They knew that this is validated. Now a common misconception with MVPs is that an MVP is some sort of prototype, and when I say MVP, it means, you know, a really basic version of what you're trying to build. And that's just not the case. MVPs are about idea validation at the end of the day and what it takes to get to that point. Well, that's your MVP. In most situations you can actually get away with not building much of anything, just anything you can use to figure out whether or not people are interested in what you're about to build. The other core reason we make MVPs is that we use them as a way to mitigate risk. If you had the option of building a big fancy platform that created forms, or just using a Google form or a basic platform to validate your idea beforehand, why

wouldn't you go with the faster, cheaper, easier option? By picking that simpler version by walking only partway down the path, we save an enormous amount of resources. Time resources, money resources, and things like opportunity costs. The last thing I'll say about MVPs and the actual lean framework is that there's a very strong focus on speed. And that's where the mantra of fail fast comes from. If you can fail fast and build your MVP experiments, run them quickly, then you can run more MVP experiments in the same amount of time and collect exponentially more data, which means that you're more likely to find a product or find a feature that fits, works, and is successful. All right, guys. So, maybe that sounds familiar to you, maybe it doesn't, that is the academic way of explaining MVPs. In the next lecture, we're going to talk more about the relationship between product managers and MVPs. MVPs are different depending on who you are as a product manager and what company you are working for. All right, see you guys in the next lecture.

How do product managers think about MVPs?

- Hey guys, welcome back to the course. So this lecture we're going to get even deeper into the concept of an MVP. We now know the academic sense of what an MVP is. Now we're going to ask the question, "What is an MVP to a Product Manager?" Well, let's first start with what we established in the previous lecture. MVPs are Minimum Viable Products, which are basically experiments that we're going to run to simulate whether or not people are interested in a new product or feature we're proposing to build. This primarily is used as a way to mitigate risk and to save us from wasting tons of resources building out the full thing. Risk is defined here on a number of different dimensions, but the main ones are risk of losing resources. Resources like time, resources like money, also things like opportunity cost. So time, money and opportunity cost. As a product manager, are those three things going to be your top priority? Now this is where MVPs don't really match up perfectly with the role of a product manager. Now, it's easy to understand why lean and MVPs are so important to startups. Startups in general don't have a lot of resources and they don't

have a huge tolerance for risk. Now you might say to yourself, "Whoa, I thought startups were super risky." Well yeah, startups do take risks on building their products but what happens if that product they build is adopted by no one? No one shows up, no one uses it, no one buys it. Well, the vast majority of cases the startup just dies. So as far as tolerance goes, if you've launched the wrong product, and that means that you die, I would say that you have a pretty low tolerance for risk. And so that's why startups hug the leanness of the MVP pillow. It's what protects them from the monsters at night that want to throw them into the dead pool of other failed startups. Now remember what we said when we talked about what a product manager is. Well, product managers primarily exist to manage products. And that inherently suggests that they're part of something that is a little bit more established. In general, if you're a product manager at a startup, you might have all the roles and responsibilities of a product manager, but your title is going to be a little bit different. It probably would be something like co-founder. Now the traditional product managers exist in organizations that are a little bit larger and a little bit more established and therefore have a higher tolerance for risk. Assuming that you already have a product and it's validated and it's has product market fit, it's potentially making money. Any experiment or MVP you run that's tangential to that; if it fails, it often doesn't endanger the main product line, the main source of lifeblood. That's not to say that a failed MVP experiment might not critically endanger the company. But when you're a product manager, you're probably already in an organization that has at least a little bit of cushion. The larger the organization, the higher the cushion, the higher the tolerance for risk, and the more likely you are to be working on a product that's farther and farther away from their core product. Now imagine you're at Google and you are a product manager, congratulations, I get paid a ton of money. Now imagine you run an experiment on a product, you launch it and fails miserably and you blow a bunch of money. Do you think that's going to endanger Google? In very few conceivable scenarios could it ever endanger that company. As long as it doesn't have to do with that oil derrick of money coming from their search ads, they'll probably be fine. So as a PM when you run an MVP experiment, you're going to to try to do it quickly and you're going to try to not be wasteful, but you are probably going to de-emphasize resources as a hard

constraint. If I were to draw this as a graph, on the left hand side, the y axis you would have risk tolerance on the bottom you would have the size of the organization. As it gets larger, they have higher risk tolerance and as their risk tolerance gets higher and higher, they start replacing the things that you're traditionally concerned with in lean and then in MVP with other metrics and aspects. They start caring about other things such as their brand or opportunity cost. Bigger organizations care about things like brand and opportunity costs because they're a bigger deal at a big organization. Brands for big organizations are big, there's a lot of inherent value in the brand that they've built up. Startups generally do not have that problem. And opportunity cost is another thing that's big for a company. A company that has lots of resources, lots of talent has the option to pursue a lot of different avenues. For a large organization, they're going to care a lot less about wasted resources and a lot more about things that startups can't even imagine such as not being the first company onto a brand new platform; or missing a big trend with one of their product lines. That's a big deal for large organizations. Now the last point I'm going to make as a product manager is that you don't have to think like a scrappy two person startup. If you make a new product or feature and it doesn't work you will survive to live another day. And because you are still there, also keep in mind that failed products can actually benefit you in other ways, other parts of your organization can absorb the benefit. Let's go back to the Google example. Let's talk about Google Glass. You remember that one? Google launched the notorious facewear called Google Glass which either turns you into an object of envy or a target of derision. Now as a product manager at Google, would you have launched Google Glass? Now some food for thought. A lot of you guys would say no. It was a big waste of money, waste of time, there is the whole glass whole phenomenon, people really just didn't adopt it. But I'm going to ask you to think just a little bit about the other benefits that could have came from Google Glass. All that money they spent, it barely even dented their balance sheet. It made no difference to the company as a whole; but they did get a ton of buzz, a ton of press. And they enhanced their brand because they're pushing the envelope showing themselves as innovative; and at least hey, they're swinging from the fences and trying something. Alright guys, I will see you in the next lecture.

Seven steps to running an MVP experiment

- Hey guys, welcome back to the course. Alright, so now you are pretty much up to your neck in information about how PMs look at MVPs, and what they are from an academic sense. You understand now that MVPs, they change depending on the size of the organization because organizations have vastly different amounts of resources to work with. So in the next set of lectures in this section, we're going to go down step by step, the process that a product manager would take to come up with an MVP experiment and run it, and then learn from it. And I'm going to explain briefly how the general flow of an MVP experiment is going to go. There are seven steps and we'll start with the first one. The first step in running an MVP experiment is just figuring out what exactly your problem and solution set is. We're not going to jump on this in this section because you should have done that in the last two sections, where we talked about figuring out what problem your product or feature is actually aiming to solve. Now, you don't obviously need to know for a fact that that problem exists and that the solution is the best solution. That's what we're going to use an MVP experiment for to figure out whether or not this is the right combo. But once you do have on the table what you want to try, the next step and step number two, is we need to identify the assumptions associated with this problem solution set. Anytime you launch a new product, come up with a new feature, you are assuming a lot of things. There are a lot of things you just take for granted, and there's a lot of different things that go into the success of a product. So, we're going to systematically sit down and try to think of what are all the things that have to go right for this to be successful. And then at the end of that process, we're going to figure out which of these assumptions is the riskiest? Which is the gunpowder in the bottom deck of your ship that can just sink the whole thing? The next step and step number three is once we have our assumptions, we're going to work on building testable hypotheses, out of them. It's a relatively simple, straightforward technique. We'll take the assumptions and we'll figure out a way of stating what exactly we think they are and then establish criteria for testing them. Building hypothesis around assumption is relatively easy. All

we're doing is building statements that we are confident about and we want to test based off the assumption. In the next step, step number four, we're going to set up what's called a minimum criteria for success. We're going to run an experiment and try to test our assumptions and prove our hypotheses. We need to know what constitutes success, what constitutes failure. If you don't do this step, you have no idea if you succeeded, or you failed. You going to draw a line in the sand, and then later, we can look and say, I know he fell right before the line or no, he actually made it past it. The next step is picking what type of MVP you're going to run. There's a lot of different ways of getting validation from your customers or potential customers. It's just going to depend on what suits you, what suits the feature that you're going to propose, and you're trying to try to build. And it's also going to really matter on how big your organization is. The last two steps are executing the MVP experiment, running it, getting out there, getting some real live data, then iterating on that, collecting the data, and figuring out whether or not you succeeded, what worked, what didn't, and what you would change if you want to redo this experiment. At that point, you'll have a hard yes or no about whether or not this is a viable option and it's worth your time compared to other opportunities you might have. Alright guys, hope you're excited. In the next lecture we're going to start with step number two which is taking a problem solution set, and identifying assumptions that we are making about it. Alright, see you then.

Identify your assumptions

- Hey guys, welcome to the course. All right, so we're going to start down the MVP journey and we're going to start with identifying the assumptions that we're making about our problem solution set. Now, in case you didn't catch me before when I said this, at this stage of the game you probably have an idea of what you are trying to build. We covered that in the other sections so your probably have a decent idea of what type of user, or potential customer, you are targeting and what is the new feature, or the new solution you're going to be proposing. Now, every single time you come up with a new idea, whether or not it's a new business idea or a new feature idea, you are

taking, onboard, enormous amount of assumptions. Like almost anything in life we think we know things and that is built up of a lot of small things that we assume to be true. If you get into a car and you think that it's going to be able to start, you're assuming that the engine is actually working, you're assuming that it has gas in the tank, you're assuming it's not cold enough outside that the spark plugs things get stuck together and it can't start. There's a lot of assumptions into that, even basic, idea. And since all these assumptions have the possibility of not being true and that creating a problem for us, we need to write out as many of our assumptions that we can. All right, so when you start identifying the assumptions that you are making it's kind of an abstract process. There's not, generally, a step by step process you'll go through. Really, you just need to sit down and think hard about what goes into you thinking that your users are going to like whatever you're going to build. You need to recognize that almost every single time you have one of these ideas it's going to be based off of some type of observation, or some type of intuition. In either case you just assume, and take for granted, many many things. Now, there are two different ways I like to try to get into, kind of, the mindset that'll help me identify these tricky little things. The first one is to try to think about The Lean Mantra which is that we're running this like a science experiment and in any science experiment we're challenging ourselves with regards to what we know is true and, so, what we need to do to be successful is to get in kind of a lean Zen mode and keep saying to ourselves, "We don't know anything." - You know nothin, Jon Snow. - Anything you think you know is an assumption. So, what do you think you already know before you've even gotten out there and tried to build something. Now the second way, and I think this one's probably more effective for most people, and I like to write this one out, is in order for this problem solution set to be successful the following must be true. And then just try to fill in all of the factors you think go into this potentially being successful. Google Glass what do you think has to be true for that to have been successful? Well, you probably have to assume that people will wear a camera on their face. What has to be true in order for even me to be successful? It has to be true that people are going to be comfortable paying money for online courses. Whether or not we're at the stage in our development that we trust online courses enough to pay money, that's a

legitimate question. If you're having natural regrets about buying this course I don't blame you, I do blame Cole. Now, in most cases there are enormous amount of assumptions that you could write out for this. All we want to do is focus on the big ones. And I'm going to give you some examples of common assumptions that almost everybody makes and then hopefully this can be a template for you going forward and it'll help you find a whole batch of other assumptions. Now, the first assumption that almost everyone seems to make is my user has X-Y-Z problem. Remember we had the problem solution set, you're probably trying to build something that targets someones problem but how do you know if that problem is real? How do you know if the customer considers it a problem? That is an assumption and it's an assumption in pretty much anything you ever build. If you're building Bluetooth Headphones you're assuming that people really are fed up with cords. If you're building the Snuggie, that big one pieced wizard suit, you're assuming that it is too inconvenient to just put a blanket on you. Now, the number two most common assumption I see is blank matters to my customers. Anytime you do a new feature or business idea, or product idea, you are assuming it's going to be successful because you're going to take some kind of angle, with it, and that angle is something your customers care about. If I'm building something that's, specifically, going to be cheaper than the competition I'm assuming people care about it being cheaper. If I'm trying to build something that is more convenient than the competition I'm assuming that convenience is a big factor in what they care about with the service. So, don't assume what your customer's care about but do write it out. Number three is that users will pay or sign up for this service or product. Now, that's kind of an optional assumption, this is assuming you have a product that charges or signs people up, in some way, to eventually make money. Eventually we all want to make money, and all products will eventually get to a point where it makes money, so you either have to go with one of those avenues, charging or signing them up. So, that's always an assumption. Maybe the problem is a big problem but do we know if it's a big enough problem for them to, say, pay money for it or give you their contact details. If I'm the genius behind the Hot Pocket, I'm assuming that people will enjoy a big bean burrito of lava so much so that they would pay money for it. And number four is that there are no satisfactory substitutes. A lot of times

we just assume that our solution is better than what's available but it's often the case that customers have already figured out a way to solve that problem and there's really no reason to switch. Give you an example. What say we are making an app that told me what the temperature is right now. Now, are there satisfactory substitutions for that? Absolutely. Not only could I go to weather.com or any of the other, hundred, weather services, I could just look out the window or just walk outside. What a concept. So, keep that one in mind and try to test whether or not there are alternatives out there that are already meeting their needs. All right guys, in the next section we're going to talk about risky assumptions and then, after that, we'll do some practice with this so you guys can see some real life examples. All right, see you then.

Follow along: Identify the assumption for Zirx

- Hey guys, welcome back to the course. Hey, I totally lied at the end of the last lecture. We're actually going to just jump in and start working with the real example. And then go over what are some example assumptions that we can think of for this product. Yes, I know I lied, you'll get over it. I'm over it already, let's start. All right, so for our example, we're going to use a service that is somewhat well-known in the San Francisco Bay Area. It's not really outside of New York and San Francisco, I think right now. It might be in Chicago and Boston. But it's enough of an idea that's a little bit out there that we can come up with some assumptions relatively easily It's a good example to use. So the example is called Zirx. It's Z-I-R-X. And what they do is that they build a mobile app that allows you when you are driving your car to hail an on-demand valet person who picks up your car and then takes it to storage. And then you can request your car back at any time. So basically what they're trying to do is on-demand valet. And the idea is that you're driving, let's say downtown. You're going out to eat, you have a car, and in any big city, parking is just a disaster. And garages are typically inconvenient, they're hard to get into, they're hard to find. Especially when you're trying to do a hundred different things at once. And they're expensive. So for all the same reasons you would use a valet, you would use an on-demand valet. But the

advantage of being an on-demand valet is that I can call a valet person, and they can take that car literally anywhere. They don't have to park it in a garage near me. They can take it out to the suburbs where it costs nothing to store. And then all I have to do is just tell them a little bit before when I need it, and it works the exact same way. It's an interesting service, check it out if you guys are ever in San Francisco and have a car and want to do it. They have a lot of funding, and they are launching in a lot of different cities. And they have a lot of competitors, mainly copycats. There's one called Luxe, and there's one called Carbon. All right, so get familiar with them because we're going to do some assumptions. Now that the core problem-solution set for Zirx is the problem is that it's hard to park your car in big cities, and/or, it's hard to find garages which are inconvenient and expensive. But then the solution to that is by using flexible on-demand valets through a mobile application to take advantage of technology for speed and for proximity so they can save money on the storage cost All right, so we're building that product. What did we assume? Well, a whole lot. Pretty much every sentence I said to you about why they work or why they're appealing was just loaded with assumptions. Now let's use the technique that we learned before and say that "Following must be true in order for our idea to be successful." So what has to be true? (mumbles) let's start off with one of the more obvious ones. Remember how I said that parking is hard to find in big cities? That's an assumption. I do not know that for an absolute fact. So I'm going to write that out. Parking is bad, or difficult, in big cities. Maybe it's the case that people don't find parking to be an issue, or maybe most cities don't have this issue. So what are some other examples that we can think of? Well let's use those examples that we learned in the last lecture. Well, most times we're making assumptions about the problem we think we're solving. So remember how I said that people find parking garages inconvenient? That is an assumption. Do people really find them inconvenient? It could be the case that people have no issue with them. They're everywhere, they're so easy to spot. They have signs and maybe those guys that twirl their signs in the middle of the road, whatever. And the corollary to that was that people find them expensive. Right? I said that, you know, there's a cost advantage by using an on-demand valet system, but do people really think that the alternative, which are garages, are too expensive? It might be the case that

people who use garages are used to those prices, and so they don't actually think of it as a problem. Let's move onto the other one. What are things we are assuming that matter to our potential user? I mean, think about the value proposition of this app. Right? We're trying to save them money and make it more convenient. Those are the two biggest things. So we're kind of assuming that someone who's driving their car downtown and wants to park it for dinner or work or whatever cares about price and convenience. And we're also assuming that they don't really care about speed, right, because if we're using the on-demand system, it might actually be slower than if you're using a garage, where you know where it is, or a valet service that's already embedded in the restaurant you're going to. Now are there any satisfactory substitutes? Well we kind of already addressed it when we said parking in the street is a substitute or parking in a garage is a substitute. But we could just say that people are not satisfied with parking their car in a garage. Now let's think about some things that we're assuming about the market. We're assuming that the market that we're going to be out to serve, let's say, has enough drivers with smartphones. That is actually important assumption. If you're going into a market, you want a smartphone app, the vast majority of people don't have smartphones, then it could potentially be a venture that is not nearly as lucrative or interesting. We're also assuming, in a weird way, that the driver can operate the smartphone while they're driving the car. I mean texting and driving is terrible, but I can't even imagine what it's like trying to place a pin on the phone while you're, like, navigating traffic cause you're downtown. So it might be the case that, you know, everyone who uses a service has someone else in the car. Maybe solo drivers is not a great idea? Another area that we're making assumptions about actually is the other side of the market, right? We have to hire people to be the valet. So we're actually assuming that people want to be on-demand valets. And then the last thing I would say is just trust and safety. We are assuming that people are going to be comfortable giving their car to, what they call, a Zirx's agent. Seriously Zirx's a terrible terrible name, but an interesting service. All right guys, so we got ten assumptions right there. Those are all big ones. All right, so the next lecture, what we're going to do is we're going to take these assumptions and organize them by risk. All right, see you in the next lecture.

Find the riskiest assumption of them all

- Hey guys, welcome back to the course. So in this lecture, we're going to talk about how do we take our assumption list, and identify which of them are the really risk ones, and which ones are not as risky. Now the reason why we do this is because it's a function of being lean, and it's a function of running a MVP experiment. Now remember, where it's all about mitigating risk and saving resources, so if we have a lot of factors that have to be true in order for our product or feature to be successful, it makes sense to focus first on the ones that could potentially sink the ship and make the entire thing unviable, versus some of the less risky things that are not really a big deal. You don't want to get in a situation where you spent a lot of time validating assumptions that were very very small and unrisky, only to come up to, let's say your sixth experiment, where you test a risky one and you fail miserably, and you completely scrap the project. That's going to happen all the time, but you wasted so many resources that you didn't have to waste had you just started the opposite direction, with the riskiest one first. In this section, we're going to come up with an MVP experiment structure that's going to target as many of the riskiest ones as we can. But we first got to figure out which ones are risky. So, let's look at the template examples that we used before. Number one was the assumption that our customer has X, Y, Z problem. Number two is that we assume our customer cares about whatever we think they care about. Number three, we're assuming that the solution we're about to create is valuable enough to the user or potential user for them to either pay you, or give you their email address, or some form of information. Then the last one on the template was that we assume that there are no satisfactory substitutes. Meaning that there's no other way of doing this that people find to be okay. So now just out of these examples, let's say you just had a theoretical product idea. Which of these four do you think is probably the biggest deal? As in, if one of these things turned out to be false, we assume this and it's not true, which of them is probably going to be the biggest deal? Which one could potentially just blow up and make all the other ones not matter. So let's just start from the matter. No satisfactory

substitutions. Well there are plenty of products and plenty of services out there that competed with satisfactory substitutions. People didn't even realize they had a problem with it until someone popped up and said, "hey, here's a better way of doing it". People were pretty satisfied with their feature phones. I mean it had a lot of issues, but for the most part if you asked someone who had a feature phone, you know before 2006, they'd probably say, "yeah, I'm pretty satisfied with it". And that was before they came out with the iPhone. So people can typically get dislodged from substitutions if you show them there's a better way on some dimension. Number three, they will pay for it or sign up for it. Well, that's a problem if they won't because it's an issue for your business model. But, is it absolutely a game changer? Something that you know, just drops the curtain, you can't possibly go forward. Well no, not really. 'Cause there's plenty of other ways of monetizing, or growing a successful product that don't rely on the customer paying you directly. For instance, you can work with ads. Ads don't require any sign up, and they don't require anyone paying you any money. It might not be ideal, but it's still a viable avenue. Number two, blank matters to my customers. You know, if you're trying to build something, let's say that is super fast, and you find out that people don't really care how fast it is. Let's say we were, you know Domino's Pizza and we came out with the 30 minutes or less delivery strategy, and we found out that no one cares. It doesn't matter if they get it in 30 minutes, or two hours, or three hours. Now would it be a game ender? Well no, but it would be a pretty significant set back. The thing is that, we have a thing called marketing. And the job of marketing is to convince people that they care about things that they didn't previously care about. So it's very plausible that Domino's could just run an ad convincing people that you want pizza, and that pizza is better if you eat it sooner. Before I bought a 1080p HDTV, if you would have asked me, do you care about the resolution of your TV? I'd say, "What are you talking about, what is resolution?" But after I bought one, and I saw what they could do, yeah all of a sudden, I started caring 'cause that was really crisp. This is actually shot in 1080p not 4K, and you can thank us for that, 'cause if we shot in 4K, you wouldn't be paying attention, you'd be lost in every contour and bump on my face. Now the last one is my customer has X, Y, Z problem. That actually is a big big deal. If we're trying to build something that solves a problem, and that problem does

not exist, we are building a solution in search of a problem, and that never goes well. So I would actually say, the lack of a problem in this scenario is by far, the riskiest, and as a product manager what I would do then is try to design my MVP experiment to target that one. After I prove that one to be true, I would then move on to the next riskiest, which is probably number two. And then the next one, which is probably number four. And then the next one is probably number three. In the next lecture, we're going to find the riskiest assumption of our example, Zerks. You guys can skip this one if you want. Totally not mandatory, but if you do care watch it, otherwise I'll see you in the next next lecture.

Make decisions: The risk/difficulty square

- Hey guys, welcome back. All right, let's talk product management strategy, as it relates to testing out your assumptions. Now as you remember, and as we have established MVP, it's just a technique for testing whether or not a new product or feature is going to be adopted by your user base or potential customers. Remember this? The next step, when you have a list of things you want to add, or test, or build is to break it down into assumptions and build hypotheses around them. That's where we're at right now. Now you have a list of everything you think, how do you decide which one you should direct your effort towards proving first? Well, we know which one is the riskiest, right? So wait, why don't we just jump right in to testing that? Well, it's complicated. And if you were an entrepreneur, I would say absolutely. The first thing you should do is just tackle the riskiest assumption you have. But as a product manager, it's a little bit more nuanced than that. There are other things you have to take into consideration to make the right decision. And that's what we're going to talk about in this lecture. So, I'm going to introduce to you a structured way of viewing your test and your hypotheses. The reality of any situation is going to be that you have assumptions, and God only knows if they're true, and you need to figure out whether or not they are. But you also have to contend with the fact that as an organization you probably have limited resources for these experiments. Now companies have to focus on a million different things, and chances are this little test you're running

is probably not the biggest thing in the universe to them. So, your design team and your dev team are really going to have to ration their time, based on how important the task at hand is going to be. And that's exactly the mentality we're going to use to pick what to test first. Now, product assumptions generally will have two main criteria. One is their risk, and two is the difficulty associated with executing on them. Now, risk in this case means how risky they are to a potential feature or product as a whole. Now remember we talked about how some can sink the ship and others just change its course. Difficulty, on the other hand, refers to how hard it would be to figure out whether or not this assumption is real. As in, how much effort would you have to channel into testing this? It's often thought of in a graph and on the Y axis we have risk and on the X axis we have difficulty. Now since we have risk and we have difficulty, we can divide this up into four different quadrants. So, you obviously have things that are low risk, low difficulty. Then we have a quadrant that's low risk, high difficulty. Another one that's high risk, low difficulty. And another one that's high risk, high difficulty. Now an example of a high risk, low effort assumption, let's say that you were going to make a subscription business for Airbnb hosts. And what it does is it mails them gifts that they give to their guests. They sign up, they say their preferences and it sends them out the gifts. Now, it's really risky that guests might not actually want gifts from their hosts. And that's something that's easy to find out because all you have to do to figure out that assumption is get one owner to give out a couple gifts or, you know, just ask a couple of guests, get them on the phone. Very risky, very easy to figure out whether or not that assumption is true. Now let's look at the inverse. What's something that's low risk, as in it's not going to sink the ship. It's not a huge deal. But to answer that question, it'd be pretty difficult. Let's stick with the same example, subscription service that sends gifts to Airbnb hosts. Well, I could think of one and it's that hosts might not buy gifts or buy items that they would have otherwise had access to in their own city. Well, how exactly would you test whether or not they'll buy these items from you that they have access to? Well, you probably have to build a website, find things they already know they can buy, and then stock those things and see if they buy those things. You couldn't just ask them on the phone whether or not they buy it. That's not a scenario where what they tell you off the cuff, what they tell

you subjectively, it's probably not going to be true. You'll need to see it in practice. Now, as a product manager, you're going to want to focus your effort on what gets you the most bang for your buck. And since you don't drive the singular focus of your organization, which by the way, this is entirely different when you are an entrepreneur. 'Cause as an entrepreneur, you are everything. But you'll have to really pick what is most worth it. Generally the best category to target is kind of obvious right? It's high risk, low difficulty. We can make pretty meaningful progress without taking up a ton of people's time. Now after that, you're generally going to want to focus on things that are still high risk, but now since you've knocked some things out, you're feeling more confident about your project. You can now start to ask for larger commitments from your team. Then after that, you're probably going to want to cross your Ts and dot your Is. And so why not just swipe through the other quadrant that's low risk, low effort, and most likely just end up ignoring the high effort, low risk, probably not worth focusing on at all. Now in general, you're going to want to focus your efforts based off of this graph. And always be on the lookout for low difficulty, simple, easy, things that are not resource intensive, ways of testing out high risk assumptions. All right, see you in the next lecture.

What is a hypothesis?

- Hey guys, welcome back to the course. Alright, so we have a big bad list of assumptions that we've made about our proposed feature or product. And in that big, bad list, we've identified the order and the priority in which we should figure out, hey, is this true? Now your goal at this point is to steadily and methodically start testing these assumptions, one by one, and ruling them out as potential issues that could blow up down the road. But we can't do that until we first solve a new problem we have. An assumption list is just a raw list of things we roughly think need to be true for the success of our product. These assumptions are not precise, and they're not particularly actionable. If I said go build an MVP that tests the assumption of, people are satisfied parking in a garage, you'd probably say, I don't know what you're talking about. We need something we can more easily work with. We to take

our assumptions, flesh them out a little bit more and roll them together into they're much more specific and easier to deal with sibling, the hypothesis. So what is a hypothesis? It's a single written, testable statement of what you believe to be true with regards to the assumption you've identified. I know that's a mouthful, so let's break it down and explain it a little bit more. Let's say that we had identified that people are not satisfied with parking in our garage as a particularly hairy assumption we were making. Now remember, this is from that example that we covered previously. Check that one out if you lost. We would probably think of running some sort of test to see if that assumption was true. It can be as simple as doing street surveys of people leaving garages on a Friday night, or maybe it's an ad we've run that says, parking in garages suck, and we see who responds to that ad. Or maybe we plan to pitch an alternative on how many people will then vote with their feet and start using the parking garages. The point here though, is that it doesn't matter yet, how we get our data and how we run our test. The simple fact is that we cannot construct a test around people are not satisfied with parking. We need to get more specific, we need to identify who exactly we think are unsatisfied, how unsatisfied they are, and potentially things like why we think they're unsatisfied. All of this information is crucial for you to understand. Now, there's a reason why MVPs are more correctly called MVP experiments. And yes, if you say, my God, Evan, look at my MVP, I'm going to correct you because come on. Now because we don't want to waste any of our resources, pursuing our new product or feature, we need to treat this entire process, like a science experiment. And the best way to focus our efforts is on one singular hypothesis, something we can try to prove true. Hypothesis brings clarity not only to you but the rest of your team. If you skip this stage, you run the risk of down the road, forgetting what exactly you're trying to do in the first place. So in the next couple of lectures, we're going to try to cover how to put these together as a matter of strategy. It's going to be up to you whether or not you want to build a hypothesis for every single assumption you have, or you want to to build a hypothesis that if proven true could rule out several assumptions killing multiple assumption birds with one MVP stone. Will cover both, stay tuned.

Follow along: Identify Zirx's hypothesis

- Hey guys, welcome back. So last lecture, we covered how to take your assumptions and then convert them into testable and easy-to-understand hypotheses. And since we've been using that company called Zirx before as our example, let's go ahead and try out making a hypotheses for Zirx. Let's first go back a little bit and refresh ourselves on what some of the assumptions we made about Zirx were. Well, we were assuming a lot of different things. We were assuming that parking in cities was going to be limited. We were assuming that people thought garages were too expensive. Maybe they also thought that garages were inconvenient, and we even had assumptions that were on the supply side, where we assumed that people would even want to park cars as part of our service. Now out of all of those once we went over, we didn't cover all of 'em in just that second, but do you remember the biggest one, the one we were concerned about the most, the riskiest one? The riskiest one was that people will be okay letting an on-demand valet park their car. Well, let's try to think about how we would build a hypothesis that turns this assumption into an actionable version of itself. Well, we first need to define out the pieces of the hypothesis and draw out our handy-dandy template. I'm using the shortened version because we're testing a new service, so we're not really trying to tie this to things like internal metrics or, say, expecting this behavioral change, because, well, we don't have any data to compare the past behavior. We're going to keep it simple. Now what are we going to put in for the who? Well, I'm guessing it's going to be people that have cars and are probably in the age range of 30 to 50, potentially with incomes over, let's say, $80,000 a year. Really, I'm just spit-balling. Realistically, you would probably do some research here, but this is a pretty plausible subject target. Now what's the action we're expecting them to do? Well, it's let an on-demand valet park their car. Now what's the problem we think we are solving? Remember again, we don't have to define the problem, but I think it's helpful. Well, there could be a lot of different reasons, and we highlighted them before in past lectures. Let's just say that it's because garages are too expensive, and that's their problem. We could even add a location if we wanted to add even more specificity to this, but that's completely optional. They themselves actually launched in San

Francisco, so we could say San Francisco, but really, it'd be any city that has bad downtown parking. Well, there you go. We've pieced together a one-sentence, testable hypothesis, and guess what? We got to wrap into that hypothesis several other assumptions that were on our list. See how that works? If this hypothesis turns out to be true in its entirety, then people do think garages are too expensive. That was one of our assumptions. At the end of completing a transaction where they use an MVP to book their first Zirx pickup, well, then that also means that it's probably not too difficult to use the service while driving. Remember, that was also one of our assumption. Now realistically we could go at this in a hundred different ways. We could build a separate hypothesis for each assumption, and then consolidate them down when we got to the stage of deciding how we need to test each one. Or we could just do the biggest assumptions in one big test and come back to the later ones after we've done our first round of testing and if we think it's absolutely necessary. Like I said, many, many different ways of doing this. We, in this case, started off with an assumption that happened to be the riskiest, but it also happened to be something that's pretty low difficulty. So according to our quadrant system, that is where we would start anyway. In the next lecture, we're going to talk about how you measure whether or not your test end up being true. It's a really, really important part of this entire MVP process. See ya then.

What's a minimum criterion for success?

- Hey guys, welcome back to the course. In this lecture, we're moving on to the next part of the process of running an MVP. Okay, so we know what a hypothesis is, and we know how to make one on our own, great. We now have our sights set on what exactly we're setting out to prove. But you may or may not have noticed this, but did we ever define what exactly proof, or define what exactly success in this experiment means? In one hypothesis we said, buy more. So, is that it if they buy more, we win? And the other said, x metric improves. There are three different outcomes to an MVP test you need to be aware of. One is, you find out that your hypothesis is categorically false and overwhelmingly not worth doing. Great, you dump it, it's not really a hard

call. Two, you find out that within a wide margin, your hypothesis is true, no question about it. And three, you're somewhere in the middle. Now here's the rub. 90% of the MVP experiments you're going to run as a product manager or as an entrepreneur are going to end up in situation three, somewhere in the middle. It's going to look like some people like it, and some don't. And the numbers are going to be all over the board, it's probably going to be a mess. So in our pursuit to see if our users buy more, what exactly do we mean? If 2% of our users buy more, or let's say our users don't even buy more but they buy twice as frequently? Is that worth it to say, build something that takes three months and four people to do it? Well, it depends on the size of your company and it depends on an internal calculation you have to make. We need to establish a concept called minimum criteria for success. We need to draw a line in the sand to definitively distinguish between worth it, all systems are a go, and not worth it. Now, why is setting a minimum criteria for success important? Well, because if you forget to set your MCS, which is what I'm going to start calling it from now on, your definition of success, you might end up conflating a validated hypothesis with a clear signal to proceed. They could be two different things. You can validate your hypothesis, for example, in a way that wouldn't make it worth it to go out and start building everything. Now what if you just barely validated? Well, you got to proceed with caution, my young product manager friend. The last thing I'll say about why we use the MCS is that it gives your experiments clarity and meaning. We aren't just trying to improve stuff. We're trying to improve stuff by X amount. So how do we figure out where our line in the sand should be? Well, creating a minimum criteria for success is relatively simple, and like the hypothesis exercise, there are roughly about million different ways to do it. We're going to take into consideration and outline two sections in this process. On the left, we're going to put the cost of building your product feature, or the cost of making the change to your prior existing product. And on the right, the second section, we're going to put the expected reward. What do you expect to see in return? Now the next step is to fill both of these sections with criteria. Whatever criteria you choose, it should probably be criteria that clearly signals not only success, but that your idea is a viable option for your company. Popular ideas that are monetarily unsustainable, are not generally great ideas. I know, someone needs to tell

that to Silicon Valley. So starting in reverse order, let's start on the right, with our expected reward section of this calculation. And we're going to pick a metric that we hope will improve as a result of building our product feature or making our change. If we're, for instance, adding additional sharing features to a social network, then we're probably targeting engagement, right? Or some other aspect we've identified as being crucial to our specific business. So that could be time spent on page. It could be actions per session. It could be number of likes, or something very literal like number of shares. Every experiment you run as a product manager is going to be pointed at something. When we talk later in depth about metrics, we have an entire section on it, you'll realize that all metrics in some way or other are interconnected, and as you as a product manager, you'll have to have your eye on a couple or a handful that you associate with product growth or product success. Whatever metric you put here needs to either be directly or indirectly related to one of those crucial metrics. Otherwise, this is in entirely pointless endeavor and you're trying to inflate what we call false metrics. So put down the potential metrics you hope are going to be moved by your experiment. If you're building a new product, it will probably be things like percentage of people signing up, conversion rate, open rate if you use email as a distribution channel, percentage of positive reactions, even. Things that indicate interest from a potential customer, or it could be literally like how much money does each customer make us? We'll talk in one of the next lectures about pre-product scenarios specifically and how to tweak this for that type of situation. For now, we're going to focus more on adding features and/or trying to improve user behavior. Okay, once you're done writing out on the right side, let's go to the left side. On the left side, we need to brainstorm the cost of doing what we're proposing to do. Every experiment is going to cost something, and you need to have a decent understanding of what's at stake. So let's say you're a company and you're going to add x feature. What does x feature end up costing your company? Well, in this hypothetical, it's probably going to take developers' time, your time, other people's time. It's probably going to take up some sweet hot cash. Keep in mind also that time is money even outside of what you buy, or what you contract to get this done. So we can reasonably assume, no matter what, that there's always going to be some form of a dollar amount to building what it is

that we're proposing. Well, let's try to brainstorm of what could we possibly have to end up buying in order to do this? Well, we probably need to promote the feature in some way, and even if we don't pay money necessarily for ads, there's also displacement of other ad inventory you could have used, or lost opportunity, because you just took up an email that you could have spent on something else. There are other things you need to take into consideration as well, like opportunity cost, which is a little abstract, but yes, it is a thing. If you pursue x and fail, then you could have pursued y and succeeded. You are one successful feature shorter. There are other things like pride, brand, morale, but these things are really up to you to self-identify as important to you and your company and put them down, because it's not necessarily true for every single company. So think about what the effects of running the experiment are going to be. Take into consideration your organizational size as well. Large organizations, as we previously learned, are not going to run scrappy experiments. The calculus for them is just different. They're probably going to go with something a little meatier because they have to put their name on it. So factor that brand aspect, and that aspect that they're going to be around after this MVP experiment, into your calculation. Okay, now that we've talked about how there are basically two sides to every calculation for a minimum criteria for success, and we've discussed roughly what are potential metrics you could put in each section, in the next lecture, we're going to work on coming up with a minimum criteria of success that works for a theoretical position. See you then.

Create a formula for your MCS

- Alright, so we have both sides of the equation outlined, and we've identified both the things we are trying to affect. We've also identified the things that it will cost if we actually decided to go full steam ahead and build out whatever our wildest dreams have in store. Now you at home that are bright probably already know that whatever experiment we run, we need the potential upside probably to outweigh the cost. And since we're setting our line in the sand of success in one specific spot, we needed to identify at what point does the benefit outweigh the cost, and that's how we figure out that line. So let's just

do that. I suggest starting on the left hand side first, so we know exactly what we're spending to make this happen. The left side also is in general easier to calculate cause it's based off of cold hard facts whereas the right side is a little more speculative. Now for demonstration purposes, let's say we work on our favorite big bloated bluebird, the Twitter bird. Let's say for some reason we wanted to add a feature that allowed us to hover over any link that someone shared or tweeted and see a pop up of whatever that link was referencing. It's clunky, it's unnecessary, but hey, it's Twitter. It's not like they ever said no to a feature request. Okay, let's try to figure out a rough cost associated if we pursued this new feature. Let's start with development time, because there will have to be people programming this to make it actually happen. If you work in tech this is frequently going to be one of the most relevant things you have to take into consideration. Now you don't need to be a developer to calculate this but you will have to do a little bit of outside research. You might have to talk to your developers and if you're using a contractor while they're generally required to give you this information up front anyway. Let's make up some fake details. Let's say our feature addition which is massively scale back in its first iteration will take two developers two weeks to develop. Okay, so now we've established what the loss of time is going to be. These developers let's say they make $100000 a year, they probably make more to be honest but, we'll say 100000 make it simple. So that means roughly rounded down to $8000 a month, two weeks inside of a month 4K for each of them. You're spending $8000 to try this just in labor costs. It's also going to take your time and other people's time. So let's just for giggles, assume it takes part time effort from you for two weeks as well. So let's add in another $2000. You generally will get paid roughly the same as developers, that makes the grand total of labor time at $10000 just to try out our pop up. Now let's say that you're going to buy some ads to announce the launch of this feature and also to drive traffic to test it in its first version, so maybe add in an extra $3000. That means our total cost could theoretically be $13000. That is a finite number that we can now work with. Fantastic, now we have a cost baseline really helpful. Now, does this number have to be exact? Well no, not necessarily, but you should be in a position to gather information you need to make a decent estimate. If you're using too fuzzy of math that probably just means that you were lazy. At the same time, don't try

to spend your entire life trying to calculate this down to the penny because if it's worth it, it's going to obviously be worth it. It's not going to be something where you only make it worth it by $10 or $5 or some thin margin. Now that we know the theoretical cost, let's think what else could be built in that time. And that brings us to the other thing than it's going to cost us, opportunity cost. Opportunity cost specifically in this case is probably small because hey, it's not a huge feature. But that still is time those developers could have spent fixing bugs or building something else, or even time spent collaborating on a newer, much larger feature. Regardless we now have a cost baseline, we've satisfied the requirements for the left side of our calculation. From the looks of it, depending on how large organization is, this is probably not a huge risk so we probably have some flexibility on the right side when we start talking about what we need in return. Let's now try to think critically about what metric has to move what amount to justify the cost we've just established. Now big risk need big rewards and small risks are okay with small rewards, that's just generally how this works. So, if we're pulling two guys off the development team and taking 50% of our time plus Add money and the fact that we could be building something else, what's it going to make to make this worth it? This is generally just up to you and your team, you will know what's a big deal and what's not. Now outside of that internal decision you guys make the best way and the best practice for going about this is to dis settle on one specific metric. Two metrics becomes too confusing and it makes it hard to really verify success. So just pick one. Second, try to think past these metrics and what exactly they mean. If you have an app that makes money off ads let's say well then increasing the time the user stays on your site will increase revenue. That's important. If you're a startup and you increase user growth by 15%, could that then affect your ability to raise money? Think through what your organization needs as a whole, and if you can't think of anything with regards to business requirements well, that's a great sign that you're out of touch and you probably need a refresher on your company's vision. Let's try some scenarios for the reward side of our calculation. Maybe we calculate money effect, a direct effect on revenue. Let's say we're thinking that the addition of this feature although limited, will in our test group raised conversion to a paid program. Let's say it's from free program to paid by some amount. That's easy, because then we

just have to figure out what is more worth it? $13,000 or that increase which then gives us more money? Which number is larger? Or maybe you can say, well, we have a year long goal of raising our average minutes consumed by 40%. To keep us on that 40% path we need to see an improvement and at least 4% since that's wonderfully the equivalent of a month and it's going to take us a month worth of time, labor wise to develop this. Now I said a month because opportunity cost, right? It's two weeks working on this feature but if it doesn't work out, you could have spent two weeks on a different one. So it's kind of a double jeopardy situation. The point is trying to align this as some bigger goal if you can. Now guys be aware, this is napkin math when your product manager, you're not always going to have the actual numbers available most times when you do this. Sometimes you're going to be adding numbers to potatoes and then squaring them my smiles and then dividing them by green. They can oftentimes be more abstraction than accounting. At the end of this calculation, you should know what metric you want to improve, how big of a cost is it going to take and then in your head, make a basic tit for tat statement that says, if it's going to cost x, we need to see why improvement or why metrics to justify giving the thumbs up and going ahead with this. Alright, so you now know how to think about and put together a minimum criteria for success. The next lecture we're going to talk specifically about pre-product scenarios and startups. How do you do it if you don't already have internal established metrics? See you then.

Optional: Make the calculation for startups

- Hey guys welcome back to the course, let's talk a little bit about startups or pre-product situations and how this process defers for calculating your minimum criteria for success. Now when you're running the show with an existing product, an existing user base, and potentially existing revenue, you're going to be one, more concerned with certain metrics, and two, less sensitive to certain types of risks and certain types of costs. PMs at a big company generally care about metrics that are just not of concern to entrepreneurs or people who are trying to figure out whether their product should exist in the first place. Average minutes consumed is not very

important if you have zero customers and you're only getting one percent of your people to even sign up for your service. Two different worlds. Specifically in the case of a startup, you're looking for what we call validation metrics. Is my product a good idea? Does it meet a user need? Does my identified problem really exist? What exactly is a validation metric? Well, it's something that demonstrates real interest from your potential customers. Percentage of people that sign up, percentage of people that share your post, average purchase price, how many people open your email. Those are examples. You're also more concerned with the economic viability of the project as a whole. It's hard to justify how much money exactly a feature will make. But a product is relatively easy. So we're probably going to look at things like lifetime value and things like gross margins. So the biggest difference when you're doing this pre-product, it's really going to be the right side of our equation. Rewards are just going to be different for you. You're going to be looking for specific things that signal interest and not necessarily some behavioral change you expected to happen. You're also not going to be as concerned with opportunity cost because if it fails, and you flop, you're probably looking for a new job. So make sure you understand the nuance between the two, because as a product manager, you might find yourself in both situations. All right so at this point in the process, we know what we want to build. Ideally it's whether it's a product or a feature or a big change to the product you already have. We know what we need to test assumption wise. And we know what outcome we want to get out of these tests. All right now let's shift our focus to the ways and the tools that are available to us to get the actual testing job done. When you run MVP experiments, you're always going to present in some way shape or form something that implies your new feature or product is already real and or coming very soon. This obviously is not the case because internally we actually haven't decided yet whether or not we want to build it. Now the extent to which you need to fake this in order to get the information you need is what's going to determine what type of MVP you need. So let's list them out and briefly discuss them.

MVP techniques: Emails, shadows, 404, and coming soon

- Hey guys, welcome back to the course. All right, so at this stage in the game, we know what we want to build ideally. Whether it's a product, feature or a large change of the product we already have. We have to know what types of assumptions we need to test to get to the truth. And we also need know roughly what outcome we want to see out of these MVP tests. Let's now take a slight turn and start talking about the ways and the tools that are available to us to get this testing job done. First of, when you run MVP experiments, you're always going to present in someway shape or form something that implies your new feature or product is already real and/or coming very soon. The reason why we do this if you remember is because we need validated learning. We need our potential customers or users to be in real live situations where they don't know they're being tested to see how they actually behave. That is the data we crave and that is the data that we have to rely on. Now, it's obviously another case that you have built what you are saying you have because internally we haven't yet decided that. Now the extent to which you need to fake it in order to get the information you need, is what's going to determine what type of MVP you need. So let's list them out and briefly discuss them. In the following lectures we'll cover each one a little bit more in depth. Talk about pros and cons and when to use them. And in the spirit of faking it. Let's start and let's start with the ones that are the most fake and we'll get towards the end where we talk about the ones that require less fakery. Alright, so the first MVP technique is what we call the Email MVP. It's the simplest one out there. All you need to run an email MVP is an email client, some of your email lists, user base, and maybe some basic wordsmithing skills. As an established company, you already have a list of users, and most likely you communicate with them on a frequent basis. I can't really think, think of any company that doesn't use, email marketing in at least some way. So your users are already used to buying things or signing up for things or engaging with the content in the emails you send them. They're already ready to have a test run on them. By taking a small segment of your user population and emailing them a pitch for a new feature or a new product. You can see exactly how they'll react. In a simulated this is real "scenario". Whether or not this is a good idea as a PM is a different question. And we'll cover that in the lectures where we go more into depth. But right now we're just covering what they are. Product managers that

are pre-product also known as early founders can use this one quite effectively. Even if you don't have an email list already, you can gather a list of emails from potential customers. I.e people that fit your target demographics or your target customer segmentation, and send this out in much of the same way. You'll probably take a much more personal tone to avoid getting flagged as spam, but it's very viable way of testing. And it is literally the simplest way of doing this. If anyone's familiar with AppSumo, they sell new products all the time in this way. In fact, they're kind of like a daily deals website, but for marketing and entrepreneurial tools. They routinely test their new products by sending out email MVPs before they acquire or build them. They generally just say, "Hey, this is real. "Come sign up." And if they're even more brazen and I've seen this, they'll put an order button in the email and see if people actually buy right out of the email. All right, so that's an email MVP. The next one we're going to talk about. It's called the shadow button MVP and this is one that's very specific to product managers. A shadow button MVP generally requires more resources to do than an email MVP, but it's still on the whole very easy to pull off. Now, what companies will do instead of building out a new feature, is they'll put a button within their already made product that supposedly links the person to this new feature. When the person clicks that button, it registers that they did it. And either one does nothing and just looks broken or two shows text saying, "Sorry, this is coming soon." That is in essence, a shadow button. Again, people don't know if it's real. So when we observe their behavior in this scenario, we can generally consider it valid. People act entirely differently when they think something is real versus if they don't have any reason to think, it's not real. You'll hear this over and over again. Oftentimes you'll see shadow buttons be used in midsize startups that are really trying to triage they're nice to have features. Which log-in would you offer at first, let's say you're a startup deciding between email log-in, Facebook log-in, Twitter, Google, and let's say LinkedIn log-in. Well, instead of implementing all of those, which is a decently large task, you just put fake buttons for them and record if people actually try to use them, if they don't use them, then you know it's not worth putting in just yet. Now is the shadow button MVP, a good idea? And anytime I say this, Cole just cringes. Again, we'll talk about that in another lecture. All in the MVP's depend on your

organization, your size, and some other factors we'll discuss. The next two types of MVPs I've actually lumped together. There's one, which we call the 404 page. The two, which is called the coming soon page. They're really similar to each other but they're just slight variations. And that's why we'll discuss them together. In this MVP tactic, what you do is again, act like you're adding a new feature or a product. And when the user navigates to the page that is designated as the new feature or product or section of your site, it either displays a 404 message, 404 meaning server cannot find this page. You'll find this any time a server's not working, or you click to a page that's old and it's no longer there. Or a page that says, thank you for your interest. The product is coming soon. And then often we'll ask them to sign up or display interest some way. Now you might be thinking to yourself, "Hey, this is way too scrappy." "My big company and our beautiful brand "and blah, blah, blah, blah, blah. "We can't use this possibly." Well, guess what? Guess, who uses this technique every single day to test demand for new products? Amazon. Amazon.com is the king of using the 404 page or the coming soon sign up page. Amazon uses as extensively to see if their existing user base is interested in a new side project or even a new category of goods. They track the interest against their criteria of success. And there you go that's how they know where to put their resources and where to expand the Amazon empire. It also works great for them because they have so many random things attached to their brand that has nothing to do with the core value proposition. So it kind of makes sense. Oculus Rift, if you guys have ever seen that virtual reality goggle system. We have an activity on it actually. Is another great example of a company that launched in a very similar way. They just put up a pre order page. I'm sure you've seen companies that have launched with Kickstarter pages. They did the same thing. It's the same thing as a coming soon page. We're coming soon, sign up or pre order, or get your foot in the door by paying us money. The difference with Oculus Rift was that when they did that, all they had was a very rough prototype. And I don't honestly know if it even worked and the only other thing was as a way to pay in advance. So that pretty much is an MVP. All alright, guys, let's take a break now that we've learned three of them, try to digest that one. Don't worry. If you feel like you're a little bit lost on some of these. We're going to cover them in more depth later. Feel free to pick which

ones you want to dive into. Take a break, we'll start with the next batch in the next lecture. See you then.

More MVP techniques: Explainer, fake landing page, and pitch experiments

- Hey guys, welcome back to the course. Let's dive right back in and start talking about more MVP techniques. Now we're going to start talking about ones that require more resource investment, look a little more real, less scrappy, but still pretty scrappy. So the next one we're going to talk about is called the explainer MVP. This is little bit harder to do because it requires some resources to make. Explainers, if you don't know what I'm talking about, are those videos that explain what an app does or what a new feature or feature set will do. They come in two different types, you have tutorial style explainers and sales style explainers. In a tutorial style, you'll see someone manually explaining over a screencast much in the same way that companies use tutorial screencasts to explain how to do certain tasks in their product. In this version, someone will pretend to use a new feature or product and explain what it does and why they made it, but they haven't actually built it yet. It's all smoke and mirrors and actually the way it looks real is in editing. In editing, you create this illusion that you have a working product by animating certain things and having them, for instance, navigate from one static page to another static page. They're not connected, they don't work, but you'll just cut the video so it looks like they do. Sounds a little nutty? Yeah, it does. But this is actually a fairly common way of pre-validating before you build. True story, this is actually how Dropbox started. Dropbox, not familiar, multibillion dollar company that lets you store anything in their data servers through a desktop and browser client. When the founder of first started Dropbox, he wanted to test to see if people wanted this idea, this cloud storage idea, where you could sync your files very seamlessly to servers somewhere else, and have access to your files anywhere you go. So what he did is he just made a fake video of him using his product and put it on YouTube. The responsible was huge so he knew it was worth building. The other type of explainer is the sales related explainer. And this is just more when you make, more like a fake promo video that pitches the product or the

feature using, who knows? It could be one of those whiteboard videos with a super fake looking hand that comes up in the middle and writes, or you could use an animated one or even a talking head video like this as your promo. A decent example I actually saw the other day was a company called Frame.io that makes a collaborative web app for video editors and producers. We actually use it to make our courses. They made a video pitching a new set of features and added all sorts of crazy animations and CG. But actually, those features had not been added yet at all. They just said it was launching very soon and to sign up for a notification. Presumably, if not enough people signed up, then who knows? Maybe they just wouldn't launch the feature and maybe it would just disappear. Either way, no one is the wiser. The point is that this MVP works because real products generally will have promotion videos of some kind. And real features will often have tutorial videos showing how to use them, so you just kind of slip in an explainer video and there you go, that's your validation. The next MVP we're going to discuss is called the fake landing page pitch experiment MVP. This is when a business will create one singular page that pitches all of the benefits of a new product and then has a call to action somewhere on the page. Since you might not know what some of these terms are that I just used, let's break it down just a little bit. Landing pages are a term used for web pages that are just one page. They're a marketing tool so that companies can run ads and have a very controlled experience when people move from that ad to that site. It doesn't generally make sense to have an ad for, let's say, an eCommerce store that just added the new iPhone SE. And when people click on the ad that says, we have the iPhone, they go to the homepage of your eCommerce store, where nowhere on the page are you talking about the iPhone. The person generally will get confused and will probably leave. So that's one of the primary uses of landing pages, although they are pretty versatile, and essentially, if you want your users to specifically do something like sign up or pay or fill out a form, you're probably going to use a landing page. Now the other term I used was pitch experiment. Now this is just a nifty little term worth knowing although you might not ever actually use it in conversation. It's a term from "The Lean Startup" framework. A pitch experiment is basically when you have an online experiment where you pitch for a new product or a new feature to an online audience and you see

what they do. Landing pages are a natural fit for this because there are one completely focused page that you can deliver a product or feature pitch directly to the user, whether they like it or not, they can't ignore it. As far as MVPs go, the landing page pitch experiment is kind of a gold standard, at least for entrepreneurial MVP test, got an idea for a product? Create a fake product page and drive traffic to it. See how they react. Product managers with existing user bases can also use this technique quite well. Instead of getting external traffic, you're just going to drive your internal traffic to a page through an email, internal ads, push notifications, whatever. Now, two classic examples of a landing page pitch experiment are one, a company called Buffer and two, a company called Basecamp. Let's talk about them. Buffer, Buffer is an online service, if you haven't actually heard of it, that helps you manage your company's social media messaging campaigns. What are the main things that will do is help you schedule when your messages out in advance so you don't have to have someone sitting at the computer writing all of your Facebook posts. Now Buffer has started off by creating a fake page, pitching what their service would do and then just drove traffic to the page. They actually set it up in a way where they put up pieces of what they thought about adding, and then when they saw what things people interacted with, they kind of just use it as like a puzzle piece, jigsaw puzzle put together what their product should actually end up being. Now Basecamp, pn the other hand is an online project management tool. It's something that's often compared to a Asana or Trello or any other popular project management tools. They didn't necessarily start with a landing page. They actually came out of a web development studio called 37signals, and it was a side project, but they are very well known for using landing pages to pre-validate any feature or new side project. Anytime they have a new side project, they make landing pages that say, this is what it does, this is why it's awesome. Here's some more information, some photos, and you should buy it or sign up for it.

Even more MVP techniques: Concierge, piecemeal, and Wizard of Oz

- All right, so the next MVP we're going to cover is the Concierge Service MVP. Concierge is, for those of you that didn't grow up in a mansion, it's a service where you get one-on one-support from someone who manually walks you through some task. It's like a concierge desk at a fancy hotel. If you call the concierge, you can ask them to do pretty much anything, and they'll try to personally help you do it. So in this MVP, instead of making the feature or product, what you can do is launch an informal offering to a small subset of users or potential users. Tell them it's a beta program or just tell them it's a new thing that you're planning out and just starting to do, then you can manually help them accomplish the task your proposed feature or product would do. The idea here is that if you are manually there to help them accomplish their task, you can see firsthand whether or not what you are doing is helpful and necessary. Let's give an example. Let's say Amazon wanted to build a bot and that bot could take written requests and then go and fetch things it thought the customer might want. Sounds like a really expensive thing to build, right. Well, what you can do instead of building that complex algorithm is just find some power users and say, "Hey, we're launching a new service, "tell us what you're looking for, "and we'll come back with personalized recommendations." And then through email, just help them fetch the items. See if they like it and see if they continue to ask for more things and also see exactly what they plan to use the service for. That then tells you who to focus on and what areas to prioritize when you launch. A great example of a Concierge MVP in action was a company called Rent the Runway, startup long time ago, very successful now. They'd rent out really fancy dresses to women who want really fancy dresses, but don't want to buy them forever, they just want to wear them once. They launched with an in-person concierge service on college campuses, where they physically helped women pick dresses, get them, stock them, organize them, return them, and then charge them. It was a hit. It was easy to figure out what people wanted. They built the product around their feedback, and the rest is history. And the next MVP we're going to talk about is the Piecemeal MVP. A Piecemeal MVP is where instead of building out your product, you take what's available in the form of out-of-the-box softwares, and by piecing those softwares together, you can match the functionality you need to test the basic version of what you're trying to build. Now, there are tons and tons of

platforms out there, and that's why this works. You can use a Weebly or a Wix or a Squarespace to create a page that can take orders, send emails, send text messages, really do a lot. You can stick softwares like Formstack, or Typeform, or JotForm to create forms that can do advanced calculations and have advanced features. You want to send text messages and have entire systems built off that you can use Twilio. You can use Recurly to manage subscriptions. There are just tons and tons of services out there for almost anything you can think of doing. Now with some creativity, you can take these things and stick them together into a Frankenstein that gets the job done until you can then justify using real resources. Groupon's a great example of a company that uses MVP. Groupon's that company that for so long just bombarded your email inbox with flash deals on things from spa packages to skydiving to that restaurant you've vaguely heard of. Well they didn't want to build out the complex system to take orders, manage their clientele, the actual stores, and generate the coupons, so they instead created their site on WordPress, and this is back when they were called The Point. The WordPress site then would email every time there was an order, and then they would use Apple Mail and a thing called AppleScript so that when it would receive an email, it could just run some basic logic on it and generate a coupon that would then autosend back to the person who needed it. All right, so we're at the last MVP tactic that we're going to cover, this one's called the Wizard of Oz MVP. The Wizard of OZ MVP is an MVP that from the front looks to be completely made, but all the tasks that computers and automated systems would normally be doing is actually carried out manually behind the scenes by an individual. It's just a fact of engineering that when you build a web or mobile product, the largest portion of your effort is going to be dedicated towards making the server-side logic, i.e., the parts of the website that make it work, but no one ever physically sees. So in this MVP, you take advantage of that. You develop a front end design that looks entirely functional, but you just have someone ready on standby to fulfill whatever the user thinks should be happening in the app. So let's say you had a social network, and you wanted to start a feature that matched people based off of their interest in their profile information. Instead of building out the complex algorithm for matching people and then seeing if people want to use it after the fact, you could build out the button that says match me and then every

time someone clicks that you manually had someone look through their profile and match them to someone, email them the results, wala. Now, why do we call it Wizard of Oz? Well, did you watch the "Wizard of Oz?" I mean it's a classic, but remember when they get through the entire yellow brick road only to find out the great Wizard of Oz was actually a man behind a curtain pulling levers to make special effects, to create that giant emerald face that was so scary. So this is the software version of that. The classic example of this was Zappos. They created a online shoe store, but they didn't stock any shoes. When people order through their online store, they walked across the street and bought it from an actual brick and mortar store, then mailed it. That way they didn't have to build any server-side logic, they didn't have to buy any inventory, and they could pretty much say they offered whatever the people down the street offered. All right guys, that's the whirlwind tour of all the MVPs that you need to know about. The following lectures, we're going to talk about each one in depth. You can skip some of them if you're not interested in the specific one, otherwise I'll see you then.

Email based MVPs

- Hey guys, welcome back. Let's talk a little bit more in detail on the concept of that first MVP we talked about called the email based MVP. It's covering what the pros and cons are. You are after all going to have to decide whether or not this is a good idea given your situation as a product manager. You could have guessed it, but this technique is best suited for smaller organizations that do not yet have much brand anxiety as I call it. The pros of this are simple. You can do it very quickly, you can limit your testing group to a small, but statistically significant group of people easily. And you can segment your users as long as you've collected secondary data on them. What I mean is, if you collect, say usage statistics or basic profile information, you can easily tie that to their email address. And then when you need to say launch a test to your power users, you can then pick only the email addresses that fit that specific group based off of their usage stats. Okay. So it's cheap, it's fast, it's targeted and you can limit the damage. Oh, and also anyone can do it. No bugging your developers that run

this one pretty much ever. Now the cons are that they can come off sloppy. It's not standard practice to ask people to say, buy something straight out of an email. So you might run the risk of having it look a little bit odd. You also might dent your brand if you're not careful to match the general tone of how your normal email marketing campaigns and conversations happen. If you do decide to go down this path and try it out, here are some tips to make it worth your while. Number one is be aware of what your audience expects of you. Are you a luxury brand or a bargain outlet? Different audiences have different tolerances for this technique. The example we used before was AppSumo. They can get away with it because they sell flash deals. So their normal product offering often looks like it's rushed anyway. They also sell to entrepreneurs who are used to being in beta groups. And they're used to seeing pre-launched products and MVPs. Number two, try your best to match the production value of the emails your company usually uses. In your normal email campaigns, you probably have templates, designs and probably some embedded images or videos. If that's the case, try to match the same. Add some images, add some video and by all means, try to use the exact same design. I'm sure your designers will lend you the template file. As for images, you don't need to bug anyone. There are a slew of easy to use tools that can help you crank these out really easily. My favorite by far is Canva. Check it out. I put it in the resources. Number three is trying to at a minimum pair this with a landing page or another MVP technique. For instance, a concierge service. You can pitch the idea in the email and then have a link that goes to the landing page for more information. Or in the email, ask them to respond, state their interest, and then run them like they're in a beta program through a concierge service. Email by itself is kind of tacky. So you can optimize it by adding in other techniques. If you think in the email, MVP will fly with your crowd and your company will go for it, well, then try it out, but be careful as to not make it look too cavalier. All right guys, in the next lecture, let's talk more about shadow buttons. See you then.

Shadow buttons

- Hey guys, welcome back to the course. Let's talk a little bit more in depth about the other MVP that I call the "Shadow Button". All right, now shadow buttons are fake buttons put on an app that just track when they're clicked and that's literally it. If you're scratching your head wondering where you've seen this before, you might not remember. They're pretty subtle. Here's one I actually found the other day, if you guys ever go to uncommongoods.com trying to log in in the sales page with Facebook. What you'll see is a image popup or a text popup and say, "Thank you for voting to add Facebook login." Great example of a shadow button. Shadow buttons are good because they give you great data. That's pretty much as close as you're going to get to showing that someone is in fact interested in your new feature. It's also pretty easy to do, your developer can probably do it over lunch. The downside though, is obvious. It doesn't look good. No one's ever interacted with a shadow button and said, "Wow, that was awesome!" They probably think what was that? Is this broken? Because that's what it looks like. It looks like your site is broken. Now that's okay if you're maintaining a product that has a reputation for having bugs and has a user population that's just gotten used to it. I'm not saying be complacent about it, you should fix your bugs, but let's just say if Udemy for example had a new feature, when I clicked on it, it just did nothing, well, I wouldn't even notice, because let's be honest, I probably ran into 15 different bugs and broken things before I even got to the shadow button. Shadow buttons are sneaky, great data, but they create a universally negative response from your users. It's just a question of whether or not the user cares about the polish of your product. If you do decide to go this route, let's cover some best practices. Number one, do it just like Uncommon Goods did it like you saw, put a message that says thank you for clicking and acknowledge what it is. That softens the blow because intentional bug is way better than a bug that they aren't aware of and simply can't fix. Number two is limit the amount of users that see this as much as possible. You can calculate what number of people need to see it to be statistically significant, be as close to that line as possible. In summary, it's easy to do, it's very valid data, generally not a good thing for your brand, so have a hard think about whether or not you think your users are tolerant enough to be okay with this type of testing.

404 and coming soon MVPs

- Hey guys, welcome back to the course. Let's talk about the best practices of using the 404 page and the Coming Soon Page, type MVP to test out your new feature product idea. Chances are you've seen these before but you probably don't remember them, why? Well, because it generally just feels like nothing out of the ordinary. As for the 404 MVPs, these are some of the hardest MVPs to spot because how do you even know if it was a test or not? It could just be that the page was broken and Coming Soon pages are nowadays everywhere. Half of which never materialize into anything at all. Now, how do we choose between running a 404 or a coming soon page? Well, just ask yourself this question, If I'm the user, would I rather think the page is broken or that the page was misleading, which of these two options is least evil for you? Whatever you say, that's probably the way you should go. The companies think they can get away with 404 of the ones that generally have tons and tons of pages, Why? Because if you see a 404, you can still jump to another product and keep browsing, not a lot of damage done. Amazon does this all the time because they have 10 bajillion roughly other pages you can look at. Clothing companies run variations of this all the time. They'll add new items, but make them out of stock and that's totally normal, no one notices this. The companies that generally get away with Coming Soon page MVPs are often companies that are smaller and where the audience is aware that they are infact new. Those companies are already working with their early adopters and those of you guys that completed my MVP course know that early adopters generally have very tough skin when it comes to this stuff. They generally revel in the fact that they are cutting edge, So in a weird way, Coming Soon Pages actually don't phase them at all they feel better about them. Now, if Walmart.com did this, it would be considered universally tacky on the other hand. So think about your user expectations and which of these result in the least negative outcome, can they get over the broken page and will this Coming Soon page pick their interest or deflated. If you decide to go this route, let's talk about some tips and best practices. Number one is design it. What do I mean? Well, there is a world of difference between a blank colorless page that really does look

like your site is broken and the clever well-designed 404 page that actually might enhance your brand. Check out the resources, there are actually lists of 404 sites that have done this really well. Some of these 404 are actually really funny and it endears you to the company and allows the company to continue a conversation with you that they otherwise wouldn't be able to. The same goes for Coming Soon pages, put some effort into them. Don't just use generic templates and don't forget to proofread whatever you put up. My best suggestion here is use a tool that is designed specifically for Coming Soon pages, the two main ones being Launchrock and KickoffLabs. Those two sites will help you pull one together very quickly and even let you import your own design assets and make it look like a seemless continuation of your product. Sometimes you can even tell they use a platform for it. Number two is shorten the horizon on these as much as possible, you don't want to see the same user seeing the same 404 or Coming Soon page, that's when that's annoying turns into I'm angry. So try to limit the link Of these tests. Bye guys see you in the next lecture.

Explainer videos

- Hey guys, welcome back to the course. Let's talk about explainer videos and how best to use them. First off, let's answer the question, why do people use this technique versus others? Well, it's simple actually. Video converts much better than just text pages. This is what every company on the planet is figuring out and it's the reason you're seeing more and more videos embedded everywhere, including your Facebook News Feed. Another reason we might choose this is because an explainer video can do a really good job of getting across a product that needs in-depth explaining. Some products are just inherently complicated, which makes pitching them difficult. You can cover way more ground in a shorter time through a shnazy video. The next question is when should we use these? Well, an explainer video, as you remember me saying can be a lot of work to create. They often need videographers, editors, and potentially custom graphics. On the flip side, they can really excite your user base if done well. Let's look at a couple examples. - [Narrator] Hello, this is Joe. Joe runs his own business selling

local goods, but Joe has a big problem. He's got too much to do, and he can't find qualified help. Everywhere Joe has looked, he's only found unqualified employees. Then one day, Joe called The Solution Company. Over the phone, The Solution Company was able to analyze Joe's needs and match him with qualified candidates. Within days, Joe had a wide selection of qualified, pre-screened workers to choose from. Thanks to The Solution Company. Joe can now sit back and relax, knowing he's got the right people for the right job. Do you want to find the right people for your business? Call The Solution Company today. Let The Solution Company provide you with pre-screened and qualified candidates to grow your business. (upbeat music) - So how do you balance this exactly? Well, like anything it's going to come down to your organizational profile. Companies that rely heavily on video already use video frequently with their users, and they're going to be much more likely candidates to use this type of test, why? Well, for one, you probably already have the infrastructure to create half-decent videos without a significant amount of work and outside recruitment. Another reason is that if your users are used to seeing video, then they're more likely to respond to your video. If you use a service like Wistia, you can see how many of your users watch the entirety of your video. Wistia, by the way is just a platform that lets you embed videos, and webpages, emails. It's really anywhere and it gives you detailed analytics about how people engage with it, very similar to the YouTube analytics system. If you run a content website that rarely uses video content, throwing an explainer video for a new feature is probably going to get ignored and skew the results of your test. Same goes for websites that simply have older demographics. Old people on video are just not really a thing. My grandma for the most part thinks the iPad is the work of the devil and videos just don't really register with her. As a funny aside, check out our Oculus Rift activity, if you haven't already. There's a video of old people using Oculus Rift and it's pretty hilarious. - My God, I'm not going to electrocute myself, am I? - A snorkeling mask? - Something you would have in war-time situation. - It's a space mask, I don't know or something you wear for the Ebola crisis. - This could be some kind of mind-control hypnotic thing you're going to try to pull on me? - Oh my? Walking without using my feet. - It's very pretty out here. I mean, it's all very fake-looking, but it's beautiful. - Oooooh! Let's not do this. - Bullshit!

(beep) - What happened to the tracks? (laughs) Which is geographical track? There were no tracks. - Other things to take into consideration are your size and your user's expectations. If you have no online presence at all, ie, you're in that entrepreneurial pre-product scenario then you can be as scrappy as you want. My favorite way of making explainers in this case is to use just cheap softwares like PowToon, GoAnimate, or even just hiring someone on Fiverr to make a video for me. If you haven't set the bar for quality yet, you can get away with whatever quality you can afford at the moment, that flexibility is fantastic and embrace it. If you're larger, however, and you have already dabbled in video in some way, be aware that you do have a quality bar to get over. If you make slick fun videos that your users are used to, then out of nowhere, you create a poorly produced screencast with low quality voice over. The difference in what users expect and what they find will taint their perception of whatever you're trying to pitch. The key here is to try to be as ordinary as possible. You don't want to stick out either way, relative to the other video content you use. Explainers are an interesting way to go but they're definitely something you need to internally assess before you try it. The bonus of using explainers is that in general video is just more engaging with your users. And this increases the likelihood of being shared, which is great. Downside obviously, though, is that it could just simply not be your demographic.

Piecemeal MVPs

- Hey, guys. Welcome back to the course. Let's talk about one of the last types of MVPs that we covered, the Piecemeal MVP. Now, Piecemeal, if you remember, is when you piece together other softwares to create the functionality that you need. To better help you visualize what this concept is, because it's kind of confusing, let's go over an example. Let's say you had an online course. Let's say you wanted to add a certificate of completion so that anyone who finishes a course would get emailed a PDF that says, "Yay, you've finished the course." (children clamoring) Well, instead of building an auto PDF generator and setting it up to know when the user finished the course, maybe you just add a link to the end of the course that takes them to

a form. They can then input their name into that form with their email, and then they can get a PDF certificate sent to them that way. How you can make this work is by, let's say, using Typeform, which is a form software, to create an online form with a public access link. Then you could have Typeform submit that information into a Google Sheet. Here I use Google Sheet. It's just Excel, but online. Which can then trigger something like Zoho Invoice to create an invoice with the information and send out an email. There you go. It was one, two, three steps, and you've accomplished your goals. If you wanted to pursue this option, the best way would probably be to physically write out what your feature or product needs to do, and then use those functions as just search terms in Google by adding things like "software" or "online" at the end of your search query. You would be amazed at just how much you can get done with available softwares that are out there. Once you mark down what you think will do it, then you just need to find what combination is going to work the best and cooperate together. Typically, Piecemeal MVPs are best-suited for when you want to add some functionality to your product that's relatively common. If enough products and people start using, let's say, text message reminders in their app, eventually there's going to be an online software that tries to simplify adding that process, and then makes money by saving companies the hassle and the money of doing it from scratch themselves. That's great, and can be a seamless integration for you. When a Piecemeal MVP is not generally a good idea to pursue is if you have, let's say, multiple functionalities you need to implement, because in general, it's kind of a pain to get off-the-shelf software to cooperate with each other. They can do it. That's the hard part, though, especially when you're chaining them together. It can get really complicated. Oftentimes, you'll just end up running into dead ends trying to find some obscure software that needs to send some obscure information to another obscure software. It can become a hairball really quickly, and you'll wish you didn't get into that. If you do for whatever reason decide that this is what you want to do, here are some tips. Number one: Piecemeal MVPs are becoming quite popular, and online softwares are so common that there are systems cropping up that can actually help you facilitate interactions between all of these softwares. The two that are worth checking out are Zapier, which I can never pronounce. It's "zap-ier" or "zape-ier." And the other one is

IFTTT, which stands for, "If This Then That." In both of these, you can tell it to say, you know, "Every time a new note with this text "is created in Evernote, send information to MailChimp," and then it can trigger MailChimp to do something else. With some persistence, you can actually daisy-chain together a whole list of these different things, and you can get quite a bit done. And these softwares will make it a lot easier for you. Number two, as a product manager, it's rarely the case that you're going to get away with sending people from one external app to another and another. Look for softwares that allow white labeling, as in, look for ones that allow you to put in your own images, put in your own colors, and put in your own fonts. Or even better, look for the softwares that can integrate into your existing product so it looks like it's part of your product. That lowers the chance your users abandon the process, and in general, it just looks a lot more professional, which is a problem with Piecemeal MVPs. Sometimes they just don't look that great, and people can tell that you're just sticking a bunch of services together. All right guys, so that's Piecemeal MVPs. Moving onto the next lecture, we're going to talk about Concierge MVPs.

Concierge service MVPs

- Hey guys. Welcome back to the course. Let's talk about concierge MVPs. As a refresher, concierge-style MVPs are where you manually help your users accomplish their goals as a means of validating whether or not they have a need for what you're doing in the first place. This is different than the Wizard of Oz MVP in that you don't actually build out anything. Just some way of connecting your users to you to get their participation, and then secondly, you are up front in this scenario about the fact that you don't yet have this feature. Those are two key distinctions between the two. Concierge MVPs are very common ways to test the needs of your users, even in big companies. The advantages are that you can run them almost in secret. All you have to do is email a batch of users and manage those relationships directly. You don't even have to add any fake buttons to your website or build any fake pages. Another advantage is that a concierge service is kind of like a hybrid between customer development and MVP testing. You did get to see

how they react and whether or not they're going to use your service in a meaningful way, but you also get the opportunity to ask why and see what their internal reasoning is. The cons of this type of experiment are that they're really management-intensive, and they can be really time-absorbing. In most cases, you yourself will want to run the task, because you know, you're the one who knows what questions to ask. Which then can add some serious logistical issues for everyone, cause you have other responsibilities. You're also going to burn a lot of resources on one user. You are just going to have to swallow the extra costs because concierge services and concierge MVPs are just very rarely profitable in and of themselves. The only cases where a concierge might actually make you money is if you're selling luxury items, in which case you can justify spending a lot of time on one person. So, how exactly do you run these experiments? Let's try an example to help conceptualize this. Let's say you're a project manager for Uber. You know what Uber is, right? And I probably don't have to tell you what it is. You want to know if adding, let's say, a wheelchair-accessible car option to your app is worth pursuing. You're not sure how many people exactly need it, and whether the drivers will get enough money to stay on the system if they're solely focused on serving that one type. So, what you could do is pick a city, take a handful of users, and email them talking about a beta program for handicap-accessible cars. You could also tell your regional managers to ask around and forward any accounts they can find that might benefit from the service. The users that sign up then get a phone number to text whenever they need pickup. It's really that simple. You could drive the car of have one person drive the car and you operate the text message. You didn't actually have to develop anything to do this. Then it's up to you to just track how often people end up using the car, whether or not they decide the wait is even worth it, and whether or not they just end up switching and using regular cars. Also, the plus side is you get to talk to them during the ride and hear what the pros and cons of the setup are. That's a realistic scenario in which you could pull this off. If you want to assess whether or not this is going to work for your organization, you need to think about the value of time. If you're too in demand and you simply can't be 100 percent committed to a concierge test, then it might be the case that you try some different way of VIP-testing. Similarly, if you need bigger data sets to justify your decision, you're

just not going to get that with this technique unless you're planning a giant, expensive operation. So, that is a concierge service. It's cheap to run, but it takes a ton of your time. Think deeply about whether or not this something that is going to fit for you. Generally, earlier in your company's history if you're a smaller group, that's generally where that's the sweet spot for concierge tests. But big organizations use them all the time. All right guys, see you in the next lecture.

Optional: How do big companies think about MVP experiments?

- Hey guys. Welcome back to the course. Now, I know after watching the lectures on all the different techniques you can use to MVP your feature or product, you might have been repeatedly asking the question to yourself, "Can I do any of these in a big, well-known organization? "What about our brand?" Well, it's a great question, and so in this lecture, we're going to talk about the answer. So, how do big organizations run MVP tests? Well, remember how we talked about the relationship between risk and the size of your company? Well, if you fell asleep, let me just recap it for you. This is the gist. The gist is, larger organizations have the resources to weather failed products. It doesn't mean that they want their products to fail necessarily, but if they do, it's not the end of the world. So, big organizations generally have a different risk-to-reward calculus when it comes to launching new products or new features. A startup's main risk with a new product launch is that they use up all of their resources. That's the biggest risk for them. But the risk big organizations care about is comprised of a lot of other things, one of which is the risk to their brand. So when you run an MVP test, you're going to have to decide how much risk you can tolerate. A general rule of thumb is that the larger the organization, the more concerned they will be with brand and the less concerned they will be with resources. Pre-product startups, remember, care the least about brand, because what does it matter how strong your brand is if you're dead and you don't exist anymore? Now remember, when you run an MVP test, there are numerous ways of tackling this, and they all exist on a spectrum. An email MVP is probably the easiest on resources, but the most reckless with your brand. A Wizard of Oz MVP

probably is the most resource-intensive. After all, you have to actually make the front end of your supposed product, but it's the most protective of your brand. People don't even notice that it's not officially your version one. If done well, it won't look to the user like anything is different. But generally, companies are going to reach a certain size where they're beyond these tactics. In those cases, as a project manager, you'll be expected to build an MVP that is really more like what most people incorrectly assume an MVP is. It's a really, really basic version of your product, and product managers in that type of situation are calling it an MVP just because they're used to calling it that and everyone else knows what you're talking about, even if it's the case that you make Eric Reese cry. Your job in these roles is to think about how exactly you can build out your features, but in a way that doesn't expend any unnecessary resources. You can't rely on parlor tricks anymore. You really have to build at least something. You can't really half-step it. The most common way that product managers tackle this issue is through a type of MVP we didn't actually cover. This type of MVP is called the one feature app. A one-feature MVP is exactly what it sounds like: when you build out literally only one core function of your product or feature, and then test its reception before slowly adding on more. To do this well, you need to take every function in your feature, or every feature in your product, and list them out. After that, you rank it based on the value to your user. In 99 percent of the cases, one function or one feature is going to stand out way more than the others in terms of value, so you just build that. Remember Foursquare? It was this app where you could check in to local businesses or locations. You could win prizes. You could network with other people, gather badges. You could do about a million different things that were all location-based. The core of Foursquare's idea, though, was, can you guess it? To let people check in to their locations and let their friends know this. So when they built their MVP to see if anyone wanted to use this, they just built literally that function. You could connect your social networks, and then you could check in. The check-in would send out a text to your social networks. They didn't add photos. They didn't add hashtags. They didn't add badges, because although these things were really secondary in value to the checking-in process, which was the number one thing they cared about. As a project manager, you're going to be constantly making hard decisions about what to build and what not. It's

really no exception in the land of MVPs. You'll have to gauge your company's tolerance for risk. Do they care more about resources, or more about brand? And then figure out the best way to optimize your resource expenditure with the information that you need to gather. It's exhausting, but I know you can do it, so cheer up. See you in the next lecture.

Evaluating results and learning from them

- Hey guys, welcome back to the course, I hope you didn't run out of gas in the section because knowing how to design and put together an effective MVP experiment is going to be crucial to being a successful product manager. As a product manager, you're going to hear the term MVP, and you're going to be expected to follow roughly the same process that we followed in the section. Coming up with an idea designing an experiment, setting up your expectations. And then running a test is how product managers decide what to add and what to go forward with. It's your flashlight in the dark, so to speak. When you run your experiment and gather your data, you're going to then take what you have and compare it to your minimum criteria for success. If it passes your bar. Congratulations. If it doesn't, then you need to see if you can figure out why it didn't work. Regardless of whether it worked or it didn't work. This is the time to sit down and evaluate what you need to do next one thing that has too be understood by any entrepreneur or product manager is that MVP experiments will primarily return quantitative data. Quantitative data if you're not familiar with the term, it's kind of the opposite of what we were doing when we were interviewing customers and doing customer development. Quantitative data is going to be data that has numbers, numbers that indicate behavior, maybe percentages of users that did this, how much time someone's spent on your page, total signups et cetera. When we were interviewing customers, we were getting what's called qualitative information. Qualitative information and qualitative data cannot be expressed in numbers. It's verbal, it's not a scale from one to 10. It's a measurement of feeling strongly about something or who is this user. And here is a crucial lesson you have to learn. If you want to make the right call. You need to make sure you're collecting both types of

data. Numbers will tell you what happened. And it will give you unbiased records of what your users did or did not do. But you can and will not know why they did what they did without also speaking to them the what without the why can be incredibly misleading. Let's say in a hypothetical we created a landing page experiment that targeted hardcore marathon runners that displayed a brand new shoe that was designed for running long distances. When you put it out there into the world and drove traffic to it. You see that, you know, 14% of users signed up to be notified of when it gets released. In fact, some of them on a separate page actually paid for pre-orders. Those cases you decide those are great numbers and you give the project a green light, you bail out the shoe and you bail out an advertising campaign to market it for the first time. Let's say that you advertise in a marathon running forum and of all the people that clicked your page only 1% ended up buying the shoe. What exactly happened? And you might think this is an exaggerated case. But it actually happens all the time, because you've never collected qualitative data on those 14% of users that signed up. You never found out that the users that signed up were actually marathon runners at all. They were casual runners, and they bought it because hey, they liked the design of the shoe. So as a product manager, if you want your MVP experiment to bear fruitful results, you can make sense of make sure you incorporate regular conversations with your users that are involved in your test, incentivize them to talk on the phone, install live chat boxes, you know, now how to get this done. Because we got a section on customer development. This is primarily a discovery process and throughout your experiments, you're going to find out things that will surprise you. Maybe your feature ended up being popular, but not at all by the people you thought would originally like it or maybe your user behavior didn't change the Ideal metric, but it did change an entirely different metric. All of this is gold to you because you're finding out the truth about your competitors needs and your competitors preferences. At the end of this process, though, you need to make a call do you make the feature and give it the green light? Do you scrap the whole idea and forget that you ever thought about it? Or do you think you should rerun the test but on a slightly different hypothesis? These are the decisions you have to make as a product manager. But now you know how to

run MVP tests which are the primary way you're going to get the information you need to guide you in making those decisions. I'll see in the next section.

Introduction to Wireframing

- Hey guys, welcome back to the course and welcome to the new section on Conceptualizing Your Ideas. If you want to build a great product, you first have to conceptualize it. And any great product, it's a result of a long process of development and refinement. No product can stand on its own without a well thought out and well-built foundation. And with that in mind, we're going to learn one of the crucial skills for conceptualizing your idea and communicating it, wireframing. Now what is a wireframe. Some of you guys have heard of this before, feel free to skip this lecture, but for those of you who do not know what it is, wireframe is a visual guide for a website, web app, or mobile app, that basically lays out a rough structure for where the content is going to go. When you're making a web app or a mobile app or just a regular old website, what is the first thing that you do, long before you have anything actually built and anything to actually play with? Well, first thing you usually do is you wireframe it. Now why do we make wireframes? Well, when you first come up with your product idea or your feature idea, you're in what we call the design phase. In the design phase, everything's kind of up in the air. Feature ideas are thrown around, you don't know exactly what's going to go in it, it's very abstract, it's very amorphous. Wireframing is usually the first step you will take in corralling that information that's in everyone's head and what's been communicated only verbally up until this point onto something that is definite and precise. Think of it as the first rough translation of a product idea. And specifically when you start wireframing, you're going to start wireframing with what we call a low fidelity wireframe. Low fidelity in this case just meaning not very exact, not a lot of detail, not very precise. So it's fidelity being accuracy, low accuracy. These versions are very basic, they're created really quickly, and they're really just meant to test broad concepts and define the overall vision for what we're trying to build. The idea here is that we're going to paint with broad strokes and this is going to allow us to then go and do some user

research, collect some feedback internally, as well as from potential users, and then over time, we can slowly define more and more of the product or feature, I mean you really start to figure out what it does, how it does it, and what it looks like. Generally, teams have a very iterative process with wireframing and as they gather more and more feedback, they're going to add more and more fidelity, they're going to add more detail to the actual mockup. We'll go from things like broad sketches of a circle that just says button to a circle that actually has shape that says sign up or whatever the button's actually supposed to do to a button that is actually designed. You'll go from things like writing scribble text, like Lorem ipsum or just gibberish to writing out a rough outline of what the content should say to writing out the actual finished copywriting that should go on the website. You'll go through that process and eventually, you'll land at the top of the high fidelity curve and your mockups at that point, if you took a glance at them, you might not know that they were not functional. They'll have graphic design, they'll have the layout and the information pretty much decided. Now do product managers themselves wireframe? Well, it actually just depends. If you're on a smaller team like a startup, you probably will be required to to some wireframing. That just because small teams, you're going to wear a lot more hats and it's kind of an all hands on deck type situation. If you can do it, they're going to ask you to do it. Now as you move up organizations and they get larger and larger and larger, your role's going to become more defined and it's going to become more narrow. In those type situations, you might not be asked to wireframe, it kind of just depends on the company, but you will need familiarity with wireframing. Typically, what's going to happen with a product manager in a larger company is that you will be asked to sit down with graphic designers and together create wireframes and you will have to contribute to it, but you won't have to be an expert or be super polished with UX skills. Another common scenario is that you might, yourself, have to sketch out an idea that you want to propose, because you don't want to take up resources from other people who can make it in a more fidelity way. So if you want to be effective as a product manager or as an entrepreneur in web or mobile technology, you need to know how to communicate complex feature and product ideas and wireframing is the way we do that. You don't have to be a UX guru. You don't have to grow a goatee

or anything, but you will be expected to know the basics and maybe in a pinch, whip up some wireframes yourself. So in the first half of this section, we're going to focus on getting you up to speed with wireframes, show you how, with a very little time investment, get in, get out, and learn to be dangerous. In a worst case scenario, you now have a skill that you can impress your coworkers with and you don't have to bug people every time you want to draft something up. All right, see you in the next lecture.

Wireframe, Mockup, Prototype

- Hey guys, welcome back to the course. So our last lecture, I introduced you to the concept of wireframing. Why we do it, what it is, and how often are you going to use it as a product manager and under what circumstances. Now that you're comfortable with wireframes, we can actually introduce more concepts now that are related to wireframes. Wireframes are kind of the entry point if you're wanting to conceptualize your product idea or communicate it. It's the most ubiquitous way of communicating. But wireframes can have more detail. They can have more interaction. And they can do all sorts of different things. At which point we actually call them something else. And that's the title of this lecture, which is wireframe, mockup and prototype. As a product manager, you need to understand what makes them different, what tools you will use for each one, and in general, how do they relate to each other? I know let's first talk about wireframes, and I'm going to explain this on a scale of fidelity. Remember the concept of fidelity, fidelity, meaning accuracy. Wireframes are at the far left, and the first thing we start with because they're the simplest. They're also called skeletons because they're very, very simple. The way I like to think of wireframes is that they're like the blueprint of your design. You're almost always going to make this first. And the reason why you do it is because you want to tackle some basic questions around structure, layout, and organization before your team iterates on visual details. Wireframes are on the low fidelity side. We are going to worry about technical and visual detail later. There's a time and place for it and it is not now. Visuals and specifications and technical details are very

important, but you want to make sure that you have the structure and the layout hammered down first. Some of the tools we'll use in wireframing might be Balsamiq. And we're going to use Balsamiq in this course. Balsamiq is incredibly popular for low-fidelity wireframes because it allows you to make them very quickly and a mock kind of sketch fashion that looks cool. It's fun to play with, and people generally respond positively to it. Another popular tool used for wireframing is Axure. If I can remember how to draw the logo. Sometimes I get confused with the Windows Azure. This is Axure with an x. Axure is a big application. Oftentimes you have it at the enterprise level because it allows teams to collaborate and has a lot of features baked into it. It's by no means a lightweight solution. Another one is Omnigraffle. That one is still used. I'm not entirely sure why it's kind of clunky but it has some interesting features. Just usually skip that one. HotGloo is another example of a popular wireframing tool. I mentioned this one because this one is actually, browser-based only. You'll notice this with a lot of tools for conceptualizing your product. A lot of them are browser-based. Why? Well, it makes it more convenient for you to store your information with them and not on your hard drawing where something can go wrong. And also you can then access that information anywhere you are. It's also much easier to share that way, share your wireframes with your teammates without having to send them an email attachment which are notorious for just sucking. And last one I'm going to mention is POP which we're actually going to use in this course. It's really, really cool. Checkout that lecture. POP allows you to take sketches that you've made and turn them into digital copies, and then you can manipulate them. All right, now what generally happens after we have wireframes? Well, that's the concept of a mockup. Now, mockups get confused all the time with wireframes, with prototypes. A lot of people don't even know what a mockup necessarily is. Mockups are actually static displays of what the final product should visually look like. The reason why we use mockups is because it gives you the opportunity to make decisions about things like colors, typography, and in general style. (mumbles) visual representational draw the same version of the wireframe, but now I'm going to add in color, and I'm going to add in some people that we can just imagine or photos. I'm going to fill in the buttons and what they look like. And we already know the structure. Now I'm just filling in

kind of the pixel specific details. Now, if you ever have trouble remembering what tools to use for mockups, well, remember who generally does a mockup? Generally the mockups are going to be done by the designers. Designers are the ones that are in charge of visual detail. And so this portion of the process makes sense for them to be the ones kind of at the helm. And what our designers use, typically they're going to use Photoshop. Photoshop is incredibly powerful. Kind of clunky when it comes to making mockups quickly, but the vast majority of designers still use that. Another option is Sketch, something that's been gaining a lot of popularity kind of as a competitor to Photoshop. Allows you to draw things faster. That's kind of their focus. It's the final cut pro to the Adobe Premiere, if you guys are in video, and that means anything to you. Also not to be left out, a lot of designers will do their mockups in Illustrator. That is a legitimate way of doing it. Kind of clunky because the Illustrator forces you to make your designs in vector, which basically means that you're going to be drawing your designs. And the designs are only going to be held in space relative to the other points of the design. I know that's kind of confusing on vector means, but think of it like this, Photoshop is pixel based, Illustrator is vector. Photoshop, when you draw something, you draw something and when you stretch an image, it stretches out and sometimes it gets blurry. But with Illustrator, because it's vector, there are no hard to find dots in the image. It redefines the image as you stretch it, because the image is only concerned with the relationship between where this point is and where that point is. That was totally extra guys. I'll put some stuff in the resources if you want to read more about that. Another tool you can use in back around to is Axure again, and another one called UXPin. Now I added these ones because they allow you to wireframe, at least Axure does, but Axure also has an image library and templates that you can easily drag and drop into your wireframe to convert them into mockups. All right, and then once you have the design hammered out, what do you usually do then afterwards? Well, now that you know, the visual design looks like this, the structure looks like this, the layout is going to be exactly the way you have it on paper. Now you add interaction. Now, if wireframes handle structure and mockups handle visuals, what do prototypes handle? They handle usability. Prototyping is actually the first phase where you can play with what

you've created. Even if it's in a very limited regard. Prototypes allow you to explore the user interface that you've put together, start looking for potential problems in the user flow and get a decent idea of what you think is going to work best. Now for a product manager, prototypes are essential if you want to do things like usability research. It's one thing to show a potential customer or someone you're doing a user test with a mockup or a wireframe, but it's an entirely different thing if you show them a prototype. A prototype can show you what they think of it and how they interact with it. With the other two, you just only get their first impressions. The best part of prototypes is it allows teams to do usability research. If you're an entrepreneur, prototypes are pretty cool because a lot of times that's all you need to recruit people and raise money. But keep in mind guys, prototypes, and when I say prototype in the web and mobile world, I mean something that I can interact with, but that interaction is going to be very limited. It's basically going to be hotspots where I press something and it takes me to a new page, or maybe, maybe I can do some basic gestures if it's a mobile prototype. Yeah, a very popular tool for making prototypes, believe it or not, is Keynote. If you use Keynote, which is the Mac comparable to PowerPoint, and you can use templates and libraries like Keynotopia, what you can do is you can make a lot of images on the slides and then connect the slides to other slides. So if I click a button on the Keynote page, it links to another Keynote page. And with Keynote, you can load that onto a mobile phone or onto a web browser. Pretty crafty I might add, but keep in mind that you're going to be very limited by how much interaction you can add with Keynote. Another one is POP. Now look at POP, how it was a wireframe tool and it's a prototyping tool. Well, because it allows us to add interactions to our wireframes. It skips the visual steps. So we're just working with skeletons, literal wireframes, and moving them around and adding basic interactions. Other ones to be aware of, again, are Axure. Axure is a massive product. It handles all three of these stages. No wonder it's decently popular. Proto.io is one that's quite popular. It's worth mentioning because it focuses solely on mobile apps. If you're doing mobile apps, look into proto.io. And the last one is InVision. And InVision is another massive product. Very popular that allows you to create multiple layers of interactions into a page that you can import mockups into. InVision gets my vote for prototyping because it's probably the best I've used. All right, you

now know what's a wireframe, what's a mockup and what's a prototype. You know that they're related to each other. And that wireframes are the least accurate, the lowest fidelity. Mockups are in the middle and prototypes have the most fidelity because they have interaction and they look exactly what the finished product is probably going to be. Wireframes are structure, mockups are for visuals and prototypes are for usability and interactions. Keep this in mind because I just saved you a whole lot of hurt for your first meeting when you walk in and you call a wireframe, a mockup and someone just slaps you. All right, guys, in the next section, we're actually going to get started with sketching. So we're going to jump right into it. I'm going to talk to you about what sketching is. How does it relate to wireframes? It is a wire frame. And how you can do it now easily in five minutes. All right, see you then.

Jump into Sketching

- Hey guys, welcome back to the course. So in the last lecture, we learned what wireframing is and hopefully you remember what that is, you don't remember? Well, they're visual mock-ups of a potential website, mobile app, or even just a feature and they're these things that we sketch out and they look like wires. And we do them because it helps us communicate the vision of the product. It helps us make sure that everyone's on the same page and it allows us to get feedback internally and from potential users. But how do you start wireframing? Let's say you wanted to make a wireframe. How would you actually start? Do you need thousands of dollars of software? No. Do you need thousands of dollars of equipment? No, actually you don't. The easiest way to start wireframing is available to literally anyone and it's a lot more simple than you think. It's literally just pen and paper. That's what we do. I'm not actually joking when product managers, when UX specialists, graphic designers, project managers, you name it, when they want to start and actually draft out their first wireframe they reach for a piece of pencil or pen and paper. Google does this, they sketch. Microsoft, they do it as well, they sketch. Facebook does the same thing. Pretty much everyone starts by drawing it out. Now, why is that? Well, because when we sort of wireframing, the idea here is that we want to do it

very quickly. It's supposed to be a rapid task. We don't want it to take a lot of time. We don't want people to think it's real or think that it's well-defined or something that we've been working on for a long period of time. We also really don't want to get attached to it. So when you write it out on paper, you're more likely to be able to let it go and throw it away, if you need to pivot or try something else. At this stage of the game, your idea is the loosest, it's the least defined and we want to stay flexible, so we use something that's very temporary. So, you guys at home grab a pen or a pencil and a piece of paper and let's draw some wireframes. Can't draw? Well, guess what? Neither can I. I've never been able to draw anything more complicated than a stick figure. True story. So, we're standing here looking at a blank sheet of paper, we know what wireframes look like, right? We also know what websites look like but that doesn't stop us from getting this kind of deer in the headlights effect. The possibilities are almost too infinite. What do I even start with? Don't worry, I'm going to walk you through a bulletproof way of getting out of that first kind of analysis paralysis and it's really quite easy. So, ask yourself a question. What is something that no matter what you draw, you're going to have to actually put down on the paper? If you guessed browser, you are correct. So actually, let's just draw out a browser 'cause that will help us get something on the paper and all of a sudden the paper went from this big to a very smaller section, much more manageable. So, draw out your browser. You guys can pick what browser you want. I'm going to pick Chrome and I'm going to put the three little colored dots in the top left corner, put some tabs and put, you know, a search box and pretty much everything else you see on Chrome. Doesn't it feel all of a sudden kind of less overwhelming? I've been doing this for years and for some reason, it always feels better after I draw the browser. So again, now what do we do? We have a browser but we have to put stuff inside the browser. Well, what's another thing that you probably will have to draw out regardless? Well, I'll give you a hint. Most websites are going to have a logo of some sort and they're probably going to have a navigation bar. So draw those out. Just put the logo in the top left corner or in the top right corner and draw an old navigation bar. In the navigation bar, you can say like one of them, is home and then just leave empty spaces for other things. Now, I know this is a little traditional. A lot of websites now don't have navigation bars or

they're floating text but for the most part, we all have some form of this. Alright, so it's starting to look like something. Now at this point, you just need to close your eyes and try to envision what you imagine it looking like. Open your eyes, look back at the wireframe and you probably still don't have any idea. So we're going to move on to the next thing. Alright, the next thing you need to do and this is probably something you've answered in your head, but you haven't yet written it out or consciously done this, is write out the question and answer the question, what is the point of this page? Why are users on this page? What is their goal? And what do you think their outcome of being here is? Try not to over think this. Write it out in the simplest way you can. It's probably going to be something along the lines of it's going to be to let users do something, something, something. Alright, so that was easy. So let's move on to the next exercise. Write out, users should be able to do and answer that question to yourself. Once you figure this part out, you're pretty much halfway there. So try to think about, what are all the things that your users should be able to do on this page? It's different than the primary function of the page, which is the main point of why they're there. But they're also secondary things they should be able to do. Let's call these functions and write them out. Don't worry, if you miss something, you're always going to miss something and later it'll come to you. These are sketches, they're not supposed to be taken that seriously. Alright, so let's say that we're building a page that shows users a lot of different videos that they can watch. So let's answer the question. What's the actual point of that? Well, the point of that would be to let users see a thumbnail of all of the available videos such that they can choose which video they want to watch. So I would just draw that out. I'm just going to draw some boxes and says video, I'm going to put video, video, video on it and that satisfies the first question. I now have established visually what the point of the page is. Then move on to the next one. Function one. What was the first function? Well, let's just take a guess. I would guess that maybe users need to like certain videos and some way of communicating that, you know, they like it, so let's draw a little hearts onto the bottoms of the thumbnails. Alright, good. So, let's check off that function. Now, let's do function number two. Let's say users probably should be able to search, they should probably be able to search that list, it shouldn't just be a random list that they don't get to curate or play with. So, let's put a search box. It's really

simple, it's just an oval with a little symbol in the middle. Great, we are satisfied, function number two. Function number three. Let's say that we need to add pagination, as in there can be a lot of search results, so we probably assume users might look at page two and page three and page four. Decide how many of the thumbnails to display in a page. So I'll put up at the top right, same way that you see it on all websites, show, you know, 10 results, 20 results or 30 results, et cetera. Now if you get stuck at any point in this, you're thinking, my familiarity with the web or mobile is just not good enough to do this. My suggestion is to always just hop online and start playing around with some major websites. Go to the websites that you go to all the time, play around for a long enough time and you'll probably run into a function or a page that has similarities, that'll get the creative juices going and a lot of times you can just outright copy them. If that doesn't seem to work for you, try to think really hard of a website that you know of, that has the same solution or problem that you're trying to solve and if you can think of that, then maybe you can find a website that might have that function. And worst case scenario, I always just go to my favorite website www.dribbble.com and it's just a website where graphic designers put up really good designs of web layouts, buttons, forms, everything and it just kind of helps you get in a creative mood, reminds you what good web pages or simple web pages should look like. So now that you know the process, I need to instill one last thing in your head. This is the most important thing you need to know as a product manager doing wireframes. And that is, its function not form, okay? You are not the UX person, you're not the designer. Even as an entrepreneur, no one is going to judge you because the user flow of your website is a little off or you sketched out something that doesn't have a comparable and it might look weird. That's not the point. The point is to get the function on the page. The form is something that someone else will figure out later. Alright guys, in the next lecture, I'm going to practice in front of you, making more sketches. If you don't want that, skip on to the next lecture where we talk about POP, alright, see you then.

Sketching out a mobile app

- Hey guys, welcome back to the course. Last lecture we got introduced to sketching. Sketching is just a simpler form of wireframing. It's actually the first form you will use in the vast majority of organizations and teams out there. First start by taking pen to paper. So we covered a pretty good way of going about making your first sketches, but in this lecture, let's get some practice. So this time I'm going to sketch out some wireframes. And if you guys want to follow along, continue the rest of the lecture. If you think you got it, skip to the next. So I don't hear you complain. Alright. This time let's not do a website or web app. Let's do a mobile application, so let's draw out an iPhone. All right. So what kind of app do you think we should try to make? I think what I'm going to try to do is let's make a mobile application for ordering groceries. Okay? Grocery delivery. That's going to be the topic of the app. The idea is, you know, kind of like a web van, or you can get on your phone and an order groceries from nearby stores to where you live. All right, now that we have our iPhone template, what should we start with? Well, I'd say an easy one to start with is a navigation bar. I'm assuming that we're going to need a navigation bar so I can put some space down at the bottom where I can click some buttons. Later on, we'll fill it out. You know, another thing that we could put instead, we could actually instead make a login screen. Pretty much any app that's not a demo or any app that is complete. And something you take seriously has a login screen, and login screens are also called splash pages. So let's make that splash page and that login page. And I like doing this sometimes first because it gets the juices flowing. All right, so let's draw out the login screen. This is a pretty standard template. I'm going to draw out. If you guys want to use something else again, just go out and find your favorite app. It literally takes four seconds. You just hit the app icon and see what the first thing you see is. What's the login screen? I'm going to put the name in the middle and the logo at the top left, like we always do. And then how do you think they should be able to log in? Well, I think it would make sense to log in with Facebook login and email. Cool. Now we have a finished screen. Now we should ask the question. Should we link these two screens in any way or at least draw some form of relationship between the two? Yeah, probably. So I'm just going to draw an arrow from the login screen to the main screen and not just shows that, you know, the login screen is first and it's a linear process going from left

to, right. This is actually how most people, when they're sketching will do it, they'll just use arrows. And then often right next to the arrows, what button they're clicking or a description of the interaction, or maybe an animation. We don't need to do that. And also guys keep in mind, this is supposed to be rapid. That's why we do sketching. No one's going to judge you based off of your ability to draw or your ability to, to make something that looks crisp and clean and make sense immediately. Remember, function, not form. All right, great. Now we have them linked and for our purposes of showing other people, that's perfect. 'Cause I'm going to show them and it makes a lot more sense to them, pretty much immediately. Okay. What should we do on the first page? Again, we're going to have to answer those questions that we asked ourselves as part of the exercise in the last lecture. Remember the first question? It is, what is the point of this screen? Well, the point of the main screen I would assume is to let people actually find a grocery store and start picking items. I think that speed is probably important. So in the first screen, they should be able to do both those things. All right. So let's draw it out. Let's make it so that we can select a store that's near us 'cause we obviously don't want to order from the same store over and over again. That's part of the appeal of the app. Then we'll put a little box where we can display the store names, just so you know which one you are actually operating. And then let's just draw in some boxes to show where the favorite items are most common items for that store would display. All right, awesome. So we actually have a decent wireframe right there, but what's put some more detail into it. Now let's answer the second question. What should users be able to do? Well, I think that users should be able to visually see the grocery items just so they can really quickly see what's available and that's easier for them. So let's draw in some actual apples and bananas and grapes and that little circular oranges. You got it, an orange. I think users should also be able to see what the price of the items are. And I don't need to put a range or anything. So I'm just going to put dollar sign, dollar sign, because I think that the price, let's just put it right underneath the image. I also think that since grocery stores have so many items and I'm on this screen, it wouldn't make sense to just show me the, the popular items, regardless of whether or not the user asks for it. So let's put a search box. So I'll just draw out a little search box. Cool, check, check, and check. The point is to be able to find a

grocery store and start the process of ordering items, other things they should be able to do, they should be able to search for their groceries, see them and know what they cost. Okay, so now let's say if I clicked on one of the grocery items, what do you think would happen? Well, I'd probably go to a page where it gives more information. Why? Because, a user needs more information about an apple or a pear or whatever they're buying in order to buy it. They're not all commodities. We need to give more information. All right. So let's make a new screen and we can connect these just like we connected the other one. So this one, we'll just call this the detail page. And then the detail page, we'll just draw a big image of whatever it is. We'll just draw an apple 'cause I'm great at drawing apples. And now the user gets a big picture of what it is. And so that saves us the time of answering the question of what's the point of this? Well, the point of this screen is to display the items so they can see it and give them information so that they can purchase it. What should they also be able to do? Well they should probably also be able to read about it. So I'm put some squiggly lines for text. They should probably also be able to buy it. So I'll put a buy button. I'm going to guess that you should probably be able to choose how many of them they're going to buy. So we'll put a little number box where they can manipulate that. And there you go. Now you have it. That is a pretty decent wireframe right there. There's a enormous amount of screens I could make after that, but I'd bore you guys to tears. Notice how though we went about it in a very systematic way. Very logical about it. Always ask yourselves the questions that we went over in the last lecture. That'll get you out of a rut. I always suggest starting with really easy intermediary screens or elements that are always on the page to get the juices flowing, to get you comfortable. All right, guys, in the next lecture, I have a treat for you 'cause we're going to use a system called POP, which is a mobile application that allow to take sketches and turn them into prototypes. Interesting, right? All right, see you in the next lecture.

Using POP

- Hey guys, welcome back to the course. Welcome to this bonus/optional lecture, were I'm going to talk about how you can take your seat sketches, add interaction to them, get them on your phone and take them to the next level using a system called POP. As always, if you're not interested, you're too busy for this stuff this is optional. Skip the lecture, I will see you later. All right, so, in the last two lectures, we got into sketching showed you how to do it, how to think through it. And then we practiced by creating a wireframe set called InstaFood. Now the idea behind sketches are that there are rapid, they're very easy to do pen and paper, that's it. And you can do them really, really quickly. So, if you have an idea, you want to communicate it, you can sketch it out and then show it to someone that instantly bring your communication and your conversation to the next level. It's also incredible useful for things like validation or user testing, because on the fly, if you want to ask about something or get someone's feedback on it, you can just whip it up, show it to them. But one of the limits of being able to do that quickly is that you hold the piece of paper and explained to them, Oh, this button goes over here and trust me, it's going to have like a thing that does this and it makes sense. It's especially troublesome when you're trying to convince someone to mobile apps, because you're holding a piece of paper and you know, if you draw an iPhone, like the way I draw an iPhone people think you're crazy. All right, so, enter a POP one of my favorite apps for sketching and collaborating. All right, so, you can check it out online, if you want to at popapp.in. I don't know what the.in suffix is. But they'll talk you through it show you how it works. It's free to use and you only pay if you start using it for tons and tons of projects, at which point you should probably pay. Anyway, it works for iOS, Android, and Windows phone. So, it pretty much covers everybody. Sorry, if you have a Blackberry, but let's be honest. If you have a Blackberry, Now, the idea behind POP is that you have bigger problems. we're going to use our phones to take a photo of our sketches, then it's going to import it into the POP system and we can add basic user flow and interactions. One thing that does make this easier for you guys, though, it was to go to the pop.in/sketchpad. You can download these sketchpads they have that are really clean, really to SPAC and for all the major phones, it'll make this process a lot easier, at least look a lot better. All right, you ready? Grab a piece of paper, draw out your wire frames and your

sketches, grab your phone and let's use some POPs. So, you'll open POP, and then it's pretty simple. You just want to create a project and I'll just talk you through this as we go. We'll name it, I remember ours was instaFood, okay. Now, what it wants me to do is say, what device that my designs are for. If I want to do landscape or portrait, I want to do a portrait with an iPhone 5 actually drew an iPhone 5 at least that was in my head when I was using it. Yes, this is an iPhone 5 and it is yellow. Create your project and then just hit the plus button here, cause hitting the plus button will ask us to add screens. So, the way we do this, as we take photos of them, who've taken photos before you can throw them into Dropbox or your Creative Cloud, or it's already stored on your phone or you can just take them right now. All right, so, then all I have to do is take the photo of the app that we just drew before actually drew out a couple more screens, so that you guys can see the interaction a little bit better. And then what you want to do is just place it on the image and line up boxes and take the photo. All right, and then all you have to do is just keep taking photos. You don't have to hit next and save and move on or anything like that. And so, we can just keep moving on. Then the third screen that we drew right there. All right, cool, now I have a couple other screens and I'm going to add, I'm going to call in some extra light because we're in the learning den, also known as a cave. And let's add some more screens. I made a screen for what happens when you hit email login. Obviously, it doesn't just automatically send you to the store front. Facebook login would do that though. So, we need to take a photo of that. And then I made it another screen for checking out. So, after you hit buy, it should add it to a screen, so you can see your subtotal and whatever, whatever you need a very basic screen for that. There, okay, when you're done hit next, you can view what you've done great. Looks great, looks great. You have the option to crop If you shot, shot it off a little bit wrong, like see how it's misaligned, so I can put it right back on where it should be. And I could also just, you know, I could cut it, so, it's like this. If I want it to be shorter or taller or whatever done. Okay, I can add brightness too, as well. If you guys are also in a cave and you might want to add some brightness. Okay, now, I see my five screens. Now, I just need to add-in relationships between the screens. We established this before, right? You have a login screen and you click Facebook login it takes you to the

storefront. If you click on one of the items you want to buy, it takes you to the detail page. And then if you click buy, it's going to take you to the shopping cart. So, I'm going to reflect that in the app. I'm going to go to the first one and just click it. And all you want to do is click on the button or the area that you want to be tappable. So, I'll just click on the Facebook login. And then I'll draw this little box. So, it has a wider area like touch with my finger great. Now, I link it to the storefront, great. I'll add one more. This email login and we'll go to the email login screen, I made, there we go. I realize now that I didn't put a login button for the login button, so I'll make that a dead end, but we'll move on to the next screen. So, this one, I would label this, the main screen up here just by tapping the title. I could call it the main screen done, and then I can add in the buttons that way know should work. So, I'll just do the apple for our sake link to the detail page, great. Moving over to the detail page, see how fast this is it's really, really simple. Double-tap the belly button, make it a little bit larger that's going to link to the cart page great. Okay, now I have a project that's set up it's ready to go. Now, what I can do when I'm talking to someone and say, Hey, I have this idea for InstaFood or at least this is the way I envision a grocery delivery app. Maybe I work at a startup that is making something like that, like Instacart. And then what I would do is I'd show this to them, but not like this. I'd show it to them by hitting play. And now play basically makes a fake version of our app. Now, the sketches look like something on your phone. So, I find wanted to login, I would hit login. Oh, and it takes me to the storefront and I would be talking them through it, right. I'd explain what the top button does and explain what these images are not everyone's going to be doing this with a chisel tip Sharpie. So, there's going to be a lot more detail to this. I do it with the Sharpies because I like to, yeah. Oh, they're fun. I'll click on the apple, explain to them how that works. Oh, I clicked buy and buy and I go to the cart page, great. That is the gist of POP. What you can also do is you can add in comments. So, what you can do is you can share it with people and then if you want to add in commenting, you just double tap in one area, leave a comment this is great. And then someone can read that. And then when I want to share it, I just go back up here hit share, and I can send out a link. So, I will text it to myself. And we can see it on our browser there you go. it works the same. I can read an introduction that describes what it is or I can just play right

through it. It works the same exact way. Check it out, cool. Then I give descriptions, yada, yada, yada. Anyway, so that's one way you can take your scratches to the next level. Awesome, awesome tool for rapidly on the go taking two-dimensional sketches and then putting them on your phone and putting them in a way that you can explain them, share them, collaborate on them. And instantly connect in a better, more meaningful way with whoever you're trying to explain this to. All right, hope you enjoyed that one. We have plenty other tools in this course at the end of this course, you'll be a master of so many different things. I'm kind of envious, even though I guess I'm teaching the course. So, it doesn't really make sense, whatever don't think about it, see in the next lecture.

Intro to Balsamiq

- Hey guys, welcome back to The Course. So in the last lectures, we learned how to sketch, we learned how to think our way through basic wireframes, and we even learned how to use POP so that we can add some basic interaction to the sketches we had before. Now, it's time to upgrade, and add a little bit of fidelity, tiny bit of fidelity. We're not going to make markups yet. What we're going to do is, we're going to learn how to use Balsamiq. I think as a prospective product manager, it really helps to have one of the wireframing tools under your belt. So I picked the tool that is probably the most popular for low fidelity wireframes, and also happens to be probably the fastest as well. That's the reason why I like it. at the end of this lecture, you'll know how to get into Balsamiq, and be dangerous, make pretty much whatever you want. It's a very simple application. It's not going to take us a ton of time to get up, and running in it. If you're the type that does not like tutorials of softwares, and someone showing you what you should care about, and think about, first, skip the lecture, see you in the next lecture. For the rest of you guys, let's jump in. All right, so this is what Balsamiq's website looks like. They offer their service in two different ways. One is a web version, and one is a desktop version. Now, there's pros and cons to both. The web version is cheaper, and the desktop version is more expensive, but the web version is a subscription, whilst the desktop app is one payment, one

time. For you guys out there that don't know if you want to buy this stuff, don't worry, they have a free trial. You can try either one you want. If you don't want to save something on your computer, go for the web app. If you're fine saving stuff on your computer, I prefer using the desktop app. It's faster, it's easier to use, in my opinion, and you don't have to wait for your browser to constantly update. All right, so take your pick, download whichever one you want to do, I'll see you as soon as you get into it. (mumbles) we like fast forward to the future, okay. All right, so this is what the desktop application looks like. It's very similar to the web application. In fact, I'm not even sure there's anything really that much different about it. Now, Balsamiq is very, very simple. There is just this screen. Not only is this application fast, it is very, very simple. Another one of the good things about it. People often fault Balsamiq for being simple, but I challenge you to find something you can't wireframe with Balsamiq. They are simple, but they've really thought out pretty much everything in their app. All right, so the way that this works is pretty much like any editing tool that uses a canvas, and then has tools, like Photoshop, like a sketch. If you're not familiar with those, don't worry. Right here is the canvas, okay? This is where we're going to take elements, and add it to the canvas, manipulate them, and create our wireframe. Now this canvas here is not defined, as in there are no boundaries to it. I can scroll up and down, and just go on forever. There's really no limit to where I can place things in this area. All right, so at the time top, you can see all of the elements that you can use. They have them sectioned off into little groups here, which is the way that you find them. There is no search for some reason. I believe they removed it from a previous version. So you're going to have to get used to these categories. All right, so they have categories for pretty much everything you would want. Now, if you want to add something to the canvas, you can drag and drop, or just double click. If you want to do add an iPhone, we go over to the iOS section, which holds a bunch of elements for mobile applications, and I can just double click on iPhone. All right, so now that's in the canvas, it's really easy. I have boundary boxes up here if I want to shrink it, or grow it in any way. It'll tell me how many pixels it is if you want to get exact. Over here, what we have is called the Inspector. And the Inspector allows us to add modifications to our elements. I can manually pick the size of it, or the positioning, notice how as I move this, the position changes. It's an

absolute position. I can change it between things like the iPhone 4, iPhone 5, I can make it a landscape, versus portrait. I can even add in some really helpful stuff, like a top bars, and lower bars, which are really helpful because remember the first thing we do with iOS at least, we always add a navigation bar. So that saves us time. Also, it gives you the option to change the layering of your canvas, and your wireframing. And let me show you what that means. Let's say I wanted to add this iOS keyboard onto my iPhone, let's make it line up, great. Now what you have to keep in mind is that these two elements are quite literally stacked on top of each other. It's like paper. That's the easiest way of thinking about it. Right now, the keyboard is on top because I created it second, but if I wanted the keyboard to be behind the iPhone, for whatever reason, I would just Send it Backward, Right? So when I hit, Send it Backward, it goes back one layer behind whatever is directly behind it, or I can send it to the bottom, or I can send it to the front with these buttons. All right, pretty simple, right? So let's start throwing some more elements on the page. We have the option of adding things like forms. These are incredibly helpful if you ever need to create a form. Incredibly tedious, but with Balsamiq, you can do it pretty easily. Buttons are over here, very easy to change the text just by double clicking them. So we can change it to, "Buy now," or whatever it is. You can change the color, you can add icons, and I'll talk about icons in just a second. You can create a form then by just throwing down a bunch of stuff that you would normally see into, in a form. And notice this one thing, as I move elements on the canvas, it will bring up guidelines, little grids, and this is incredibly helpful for aligning things. Specifically, if I was making a form, I would want everything to line up on the left, right? Well, I can do that really easily without having to zoom in by clicking the second element, and then just lining it up right there with the left end. I can also line it up with the middle of the element. That's tricky, but there it goes. If I want to do center alignment, or with the middle of the design, which actually happens to be right there because it's the middle of the canvas. All right, let's see what else we can add in it. Let's throw in some tacks right there, we'll call it a form. Let's see what else we can add in. Let's say we wanted to grab some numbers, maybe turn something on and off right there. Line `em up, and there you go, you have the most basic form ever made. Now, I also have the option of adding icons. Now, I said I'd get back to these. What an icon is, it's

just going to be a blank space whenever you add it. But if you double click this magic thing, it gives you an enormous option of icons you can choose from. Now don't worry, it's not going to send these icons into a box. It's going to take the place of the box. We have hundreds and hundreds of icons here. You can pick the size that you want, and it'll give you a preview. You can also search over here for whatever specifically you wanted. I want to get really specific. I'm looking for a rupee. There we go. That's actually the symbol. Great, now I can use that for whatever I was planning to use it for. Similarly, I can add icons to things like buttons. So in the button right here, over here, I can add an icon, okay? So if I wanted to add rupee to that, there we go. Now we added a rupee, and I can also change the size to make it fit, so it doesn't stick out so much, and there we go. All right, so that is the basics of Balsamiq. There's not that much left to go over. You can add certain things like states, two buttons. See this button. When I say a state, a state could be clicked, or unclicked, for instance, or grayed out. I could put the state. It could be selected, and focused, or disabled. Say selected, see how it changes the color. that's helpful if I'm putting wireframes in a row, showing a sequence, or if I want to get really specific saying, when this button is depressed, then you see these other elements on the screen. Over here in the Common section, you'll see all the common things that you would see. Image boxes are right here, notice how before, in wireframes typically you won't have the image, or sketch out the image, you'll put a box in the right image. But if you wanted to, you could put something in this. I like doing it like this, clicking it, and hitting Enter, and then putting image in the middle, so you can see that, "Hey, this is an image." You can also just double tap on it. If you wanted to add an image in that container, it'll insert an image into the container, and constrict it, and then you can expand it to your heart's content. If you ever want to duplicate something, it's relatively easy. I just use copy and paste. You can do it up here, Edit, Copy, and then Control + V, and you can copy it right there. And then you can use grids for alignment, yada, yada, yada. The last two things I want to show you are one, annotations and two, then how do we export? Now, Annotations in Balsamiq are controlled in a really kind of a goofy way. We just use it through stickies. Now, go over to the markup right here. If I wanted to add something onto the page, like a comment for someone else reading this, right here, I could put a sticky, and

then put a description saying, "This is the screen that does this," or some comment that I want other people to notice. And then I can use arrows right here. If I want to change the arrows, I have in the Inspector, ways of changing it. Changing the size, the length, which way it points in which direction. In case I wanted to say, "Point the sticky in that direction." Thought it'd go like this, and there you go. When you want to save your creations, you have to export them. And you'll have to export them either as an image, or a PDF, or you can share it, and use a share link online. That's great if you want to share it with other people, otherwise you'll have to export it, and send it to them. If you want to create multiple markups, by the way, you just hit like this, and then you can create another markup. And you go here, it'll show you all the markups you have made. When you then export, it'll put all of them together into one long PDF, and it, if you have a lot of them, it'll create a table of contents at the top. Alright guys, that is for the most part, all you need to know to get into Balsamiq and get dangerous. There's a lot of other things I obviously didn't cover, but you'll figure them out pretty quickly. That is 90% of it. All right guys in the next lecture, we're going to practice this just a little bit. If you don't feel like you need practice, then just skip onto the next lecture. It won't hurt my feelings. for that, see you guys, I'll see you in the next lecture.

Building YouTube in Balsamiq

- Hey guys, welcome back to the course. So in this lecture, we're going to get some practice with Balsamiq. We're going to create a website from scratch using all the tools we covered in the previous lecture. Now, for those of you guys out there that don't want to follow along or don't feel like you need practice, skip the lecture. And I'll see you in the next lecture. For the rest of you guys, let's get started. I remember what I said in the last lecture, balsamiq is incredibly simple and incredibly fast. So simple and so fast that in the 10 minutes we spent in the last lecture whenever it was learning how to use this system. That was all we need to create an in depth website. And to prove my point, let's go ahead and make a website. To prove my point, I can... Let's go ahead and make a website. I'm going to make a

copy of YouTube. When I speak after YouTube, I'm not talking on the homepage, I'm talking about what the screen looks like when you're actually viewing a sub-page. I would go to all, go to the browser window. Here's in the browser window, I can type in YouTube. And there we go. See, it shows it up in YouTube. Let's expand the size. Yap. Okay, now let's make something like where you'd watch the videos. Where's the media player, go to media. Now media is where you're going to insert pretty much anything you want like to be associated with images, here is a Progress Bar, here's a Street Map, trying to pull up things here. That's like a Google map Street Map. Here's a video player. So let's say that this is, where you watch videos. So let's say we want a search bar at the top, you can use the search bar, let's called it a search box, right? Here's at the top where you will be searching for videos. Let's see, I love this about balsamiq, you can just use icons and they have a lot of the logos that you would commonly use, like they have the Twitter logo and the Instagram logo and pretty much everything you'd want for the most common social platforms. You can do that like now we have our logo, YouTube. Let's make that little section over here where it shows you suggested videos. What I would do for that 'cause of thumbnails, I'll use an image. Images just come out like boxes, which is perfect for what we want. Just call it a video so that our developer or graphic designer knows what that is. With image placeholders, what you can do is you can add borders and you can actually upload images insert them into balsamiq. So what you do is you just double tap it. Well browse for a file. Everybody loves cats, cats. This is a cool feature I like, we're going to have to just sketch it. If I don't hit Sketch It, it's going to insert this image with a whole color, it's going to look kind of weird the whole black and white thing going on. So I'll just say Sketch it out. Let's try to put it back into the size that you expect. Yeah. Okay, so that looks weird. And we want to put it, let's put a border on it a little better. There you go. Now looks kind of like one of those videos you'd have suggested to you. Now let's add the text that you would go with next to it. So again, this is how I do it. Remember string attacks. And I'll put it as the title of the cat video. Super, super original. And since it's the title, I want to make it bold. Let's increase the size just a little bit. Okay. Now you can use your keyboard to move things around. So if I just go left, It's going to move at one pixel left, right up, down, hit point. And then right below

that, see now if I want to put text right below it, all I want to do is hit duplicate, put it right below it and change call by. And we'll just call it the cat, person, person. 'Cause everybody knows a cat person but nobody knows a cat person person. Let's see, and we'll make it smaller with the bolding you go. And let's put a view counter. Here we go. Say, that's usually just a number. So say a million people watch this video. And usually that is a little bit smaller. I think it's pretty believable. So what I'm going to do is I want to create this little grouping and I want to make it go all the way down the screen, which is typically what you'd see in YouTube. So what we do, click it. We're going to group these together. Remember how we do group, We had group now they're together. Now I copy my group even exit out doesn't really matter. Copy my group. Voila, there we go. Smash the grids up. And there we go. These are suggested videos. Now maybe I wanted to put a container around that, something light. I'll just go to shape again. A shape, I am just going to pick a rectangle right here, right here. And I want to expand it over. See you notice how it covered it. Now the balsamiq has a layering, kind of like Photoshop. If you use Photoshop before it has layers, the way that you would want this to fade to the back, is you would use this little subgroup, this will send it back one, this will send it all the way to the back. So this will be the bottom of your... Imagine this is kind of like paper laying on top of each other. This would be the bottom piece of paper, you click this. If you click this, it's going to bring it all the way to the top. So it's going to cover anything behind it. I just want to send it back and tell it's behind the cat videos. Alright, so the rest of it really is just kind of tiny little details. It's up to you how much detail you honestly want to put in here. We can tend like this is a, here's a placeholder for the image, whoever made the video can add in text again. So the name of the video, Super Secret Webcam video, 'cause they want it to be a secret but they also want the entire world to see it and let's put this realistically much larger. Okay. Let's move this down, add some space, you can add those thumbs up and thumbs down and just pick an icon. Put it over here. Can even just search for thumbs, thumbs on, there you go. Run it as small, makes more sense, right? And you can even copy it right next to it and then just find another icon. Thumbs again, some thumbs down, like the video. Yeah, well then just stick someone's face on it. Because it's a webcam right? You get the drift now. You can put an enormous amount of detail into

balsamiq if you want to. But as you've noticed, that really did not take me a lot of time and I have a Wireframe of a video, you're obviously going to put more time and effort into whatever you actually build yourself. But once you actually get acquainted with it, hopefully by now you are acquainted, you can really move at lightning speed and you can really produce these very, very quickly. Now if I wanted to make the next screen at YouTube, I'll just go up to File, clone current mock up and look, I have the same exact thing and I can just move from here and start deleting things wherever I want and changing things around. Now when you're done and you're happy with what you got, all you going to do is just go up to File and you can save it as whatever you want it to be. Make sure to back it up. If you guys have an older computer, make sure you don't have tons and tons and tons of tabs open. It can really kind of slow down your computer. But otherwise, just export it to whatever you want. I like PDF. That's one of the easiest to view or you can just do it as an image. That's what you end up sending your graphic designer or your developer and they will thank you 1000 times over

Introduction to metrics

- Hey everyone, welcome back. This section of the course is about metrics. And this is one of the most important topics that we will be covering in the course for a product manager. As a product manager, basically your entire day, your entire role actually revolves around metrics in some way or another. So we will cover metrics in general. Like what are they? why are they important? Then, how do you pick them? We're going to go over some real life examples of metrics at real tech companies and then we're going to talk about some metrics frameworks that will help you pick and understand your metrics. So pay close attention because this is a big section. So there's a very famous saying that goes, "What gets measured gets managed." And that is by Peter Drucker, who is a very famous old school business consultant, management consultant, that sort of thing. And he was actually most active in the earlier parts of the 1900s. That is so true and metrics are really all about that. So have you ever been driving down the road and seen one of these radar automated speed limit signs, the ones where it sets the

speed limit, but then it also uses radar to show you exactly how fast you are currently going? These signs have been proven to reduce speed for people even more effectively than under threat of getting a ticket from a police officer. The reason is because the feedback is instantaneous. So when you get tickets from the police, you're probably only getting one every few times that you actually speed and a police officer catches you and then decides to give you a ticket. So, depending on how much you speed is however, many times you speed and getting a ticket in that period of time. But with these signs, every single time you go past a speed limit sign, you're getting direct feedback on this is how fast you're going. And this has been proven again to be way more effective than just be being ticketed for reducing speed. The reason that this happens is because of a human nature of a science called feedback loops, which means that the more frequently you're getting accurate feedback about something, the more effectively you're going to manage that. This same exact principle is behind the success of the fitness tracker industry. So people that wear a fitness tracking arm bands or watches, or heart rate monitors or whatever, they're getting instantaneous feedback all the time about their calories or the way they slept or what they've been eating. And this helps people lose weight or become healthier and this is the exact same reason because they're getting really quick feedback every single day or even every hour, if they're looking at their device pretty frequently and this is that science at work. As a product manager we can use the same science to really help our products be successful. And this whole feedback loop process through metrics, is really the core of agile development and your role as a product manager. So, what are metrics? Well, metrics is just a fancy term for a number that describes what's going on with the product. It's a measurement of something. And sometimes you'll hear the word KPIs, which stands for Key Performance Indicator and this is, you can use these two terms interchangeably to be honest, you'll hear both of them depending on the company you're working at. I think KPI sounds a little bit corporate so we just used the word metrics, no big deal. Some common examples of metrics are things like; monthly active users, returning users, churn users, things like even App store reviews. Maybe things that people do on certain platforms like Facebook or Twitter, it's how many posts are they made? These are all metrics. It's just a number saying how many

times someone has done something, or even in terms of, let's say a Netflix or a streaming music company, they have metrics for how fast between the time that the user hits the play button to when the video starts. And in this case, that measurement, they're trying to decrease to be as small as possible to have the fastest service basically. So metrics are different depending on the type of the product manager you are, the industry you're working in, the company you're working at. And we're going to talk a little bit more about that. As I keep saying, metrics are a huge part of your life as a product manager. And as you know, product managers do a lot of different things that we've already covered. But one of the most important things is that you must accomplish a goal. And in order to basically accomplish that goal, you have to first say, what are you accomplishing. You have to do what we call defining success and you do that through metrics. So you basically say, hey, these are the numbers we're going to try to hit in this metric, by building this thing and if we do it, we're successful. And if we don't, then something's wrong. If we're getting close, then we know we're moving in the right direction, but this is our feedback loop as a product manager. So back to the real world example on a fitness tracker, if a person is trying to lose weight, then they're going to have a goal of eating under a certain amount of calories or burning a certain amount of amount of calories per day. Exact same thing for us. We have to say, here's what we want to hit and when we get there, that means we're successful. Again, this is called defining success and this is something you've got to do before every single endeavor you or your team does, or even your company does. You have to know what are you actually trying to do here before you start doing it. So here's a real world quick example from, let's say Facebook as an example. Let's say that Facebook is trying to increase the amount of time that people spend on Facebook. Well, that's like a company level goal and now let's also say that, from this we know, we want to build a feature that's going to basically encourage people to stay longer. But it let's say we are a product manager on a smaller team. And we know that people who comment more on Facebook, where they make comments on posts and stuff, they tend to longer than the average person. So they looked into the data and we know that. What we want to do in that case is look at building a product that, or building a feature rather, or improving that product to basically encourage

people to comment more. And we would say, okay, we're going to go for a 5% increase in comments per person on average, which will in turn affect the average amount of time that everyone is spending on Facebook in general, on that top company level metric. So next going to take a look at some real life examples of metrics at some companies you may have heard of before. I'll see you in the next lecture.

Real-life examples of metrics

- Hey, so welcome back. And last lecture, we talked about just the basic intro to metrics. So what metrics are, they're just a way to measure stuff. It's a number that is a measurement that tells you something about your product. And it's a huge integral part of product management, because it's the way that we are basically knowing if we're doing things right or not. So in this lecture, we are going to chat a little bit about some real world examples, and it's basically just going to be me talking about a couple of companies and some of the metrics that they track. And then afterwards, we'll talk about a couple of similarities between these companies. So the first example is Twitter, and they have some growth metrics that a few other companies do as well, which are total new users per month or per week, or however, they can slice it up however they want. Then you have monthly active users, probably also daily active users, and then probably activated users as well per month. And we'll talk a little bit more about that a little bit later in this section. So then they have some engagement metrics as well. Things like the multiple logins per day, probably time spent looking at Twitter, either the app or the website. Then they have things like number of Tweets sent and then per user, you know, on average. And then they have probably the average number of likes and the average number of retweets. And then probably also the average number of follows. So they have a good idea of the social activity going on. And then they would also have probably the number of messages sent, the private messages in their direct messaging portion of the site and the app. So the second example would be YouTube, and YouTube and many other companies are going to have the same type of growth metrics that we already talked about with Twitter. So you have your monthly active

users, you've got your new users, your activated users, your daily active users, but where some, a lot of tech companies end up differing is their engagement metrics. Again, we're going to talk about the difference between growth and engagement in another lecture. But for now, just understand that most companies will have similar metrics and then they'll have some that are completely different from any other company. So on the YouTube example, they probably have stuff that Twitter doesn't, like video views, first of all, per user. Like how many videos did each user watch on average? And then they would have some criteria inside of that. Like, okay, they must have watched at least 30 seconds of a video in order to call it a view. And that's an indicator of quality or a conservative indicator for them, basically letting them know that this person is actually watching this video. Secondly, for YouTube, they will probably have a average viewing time overall per user. So you could have someone that watched 20 videos, but if each one of those videos was one and a half minutes long, that's not as valuable as having someone that watched one video, which was three hours long, like a documentary or something. And notice that these are specific to YouTube. Okay, so then we have our third example, which is the favorite example, Facebook. And Facebook's got some really interesting engagement metrics. Again, on the other metrics, the growth ones, they're going to have the same sort of stuff that most people do, most companies like, you know, new users and monthly actives and so on. But engagement-wise, they have some pretty interesting ones. I've seen some stuff online that talks about them having metrics like position clicks on the newsfeed. And that basically means they are tracking which position on average did a person engage with in their newsfeed. So let's say you have you open the Facebook app and you have like something at the top of your newsfeed from one of your friends and you have, you know, 10 other things under that, and you engage, or you click on, or you like, or you comment on the one that's like the fifth position down. Well, they actually think that's bad. They want you to engage with the one that's the first, and the reason they do that is because it's a measurement for them of relevancy. So they, basically, in their mind, they want you to see the very, most relevant thing, the thing you want to engage with or comment on, or like at the very top of your newsfeed. So of course they track where you're clicking or where you're engaging or where you're watching the videos

at, because of it's too far down the page, then they're not doing a good enough job of ranking your newsfeed based on your interests. So other than that, they're going to have some metrics like of obviously the message is sent, and then time spent on site. And the interesting thing about Facebook is that they also own a WhatsApp and Instagram, things like that. So if you leave Facebook and go to WhatsApp or leave Facebook and go to Instagram, that's okay with them because they own those properties as well, and you're still basically inside of their universe of companies. So then they're also going to have the number of likes that you make on some, on average, per day, on other people's items, in the newsfeed or wherever. And then of course they are also going to be tracking things like the average number of likes that you as a user get per post. So on average, some people have a higher average like count per post, obviously if they're more popular or whatever, more famous or celebrities, that sort of thing. And then the interesting thing about that is from what I've read, they actually try to keep these numbers relatively stable. And they also don't want anyone to not have any likes at all, or any comments at all, because if you post something and you're sharing it with your network and then no one likes it or comments on it, then you're less likely to come back and post again. And of course they want you to do that. So what they'll do is apparently sort of hold your item in the newsfeed in front of people longer so that, you know, people will see it and maybe actually interact with it until you get to some minimum threshold. And at that point, you know, then it'll start moving down the newsfeed as a normal item. But that way you're like, "Oh, hey, I got some likes. Like, I'll come and do this again." So that's pretty interesting. Okay, so just to recap real quick, each company is going to have a lot of the same metrics as others on things like new users and active users and all of that. But when it comes to engagement, they're probably going to have some metrics that are specific to their company or their industry or what they're trying to get you to do. And they're really tracking those things closely. So I know I've been talking a lot about growth versus engagement metrics, and those are different categories. And in the next lecture, we're going to talk about these categories. So I will see you there and pay attention, 'cause it's an important one.

Metrics of all kinds

- All right, everyone, welcome back. And the last lecture, we talked about some real world examples of metrics from companies you've probably heard of. There was YouTube, and Twitter, and Facebook. And you might have noticed that when I was talking about some of these metrics, I said that here are some growth metrics, the new users and the monthly actives and that sort of thing. And then here's the engagement metrics, which are things that were a little bit more specific to each company, a lot of times the growth ones were the same, because they were the more generic metrics, and then the engagement ones were very specific to the platform. In the Facebook example, we had the newsfeed position engagement number, and on YouTube, we had the number of views that were over 30 seconds on average per person. In this session, we're going to be talking about these different types, these categories of metrics. And there are a number of categories of metrics out there. And there's no official definition for any of these. But this is in my experience, how they're usually sectioned off. So in practice, there's a whole bunch of these different categories of metrics. And these are not, there's no dictionary definition for this and you're never going to have the same ones across companies. But there's sort of general categories that in my experience, I've seen at the companies I've either worked with or worked for. Some of the most common and the ones we're going to talk about here are growth and activation as one category, engagement, retention, then user happiness, and then revenue. And those are really the the big buckets. So we'll talk a little bit about each one of these, get some real life examples and define a little bit further what they mean. The first category is growth and activation. This is pretty self explanatory. This is a bucket of metrics that basically tracks and measures, informs you how your company or product is growing. So some examples of some growth and acquisition metrics are going to be total new users per month or per week, then you're going to have new users by source, we'll talk a little bit about that. And then activated users. So if you're working as a product manager, either on a growth team inside a company, or you're a product manager at a smaller company that's also responsible for growth, then you want to know obviously, how many new

users you have. But you really want to know, where are these users coming from? Are they coming from the SEO optimizations you did, which is search engine optimizations? So that people can find you via Google and search engines and stuff like that. Or is it that they're finding you on an App Store? Or is it that they're finding you on maybe a blog somewhere? Or perhaps they're finding you because one of your users shared some content from your platform or your app out to another platform like Twitter, and they click the link and like, "Hey, what is this?" These are all very important to track and that would be new users by source. So we're tracking where they're coming from. So activation is a big one because there's a really big difference between a person that has been considered someone who's found and maybe installed your app and an activated user. So what usually happens is a company or product manager will say that an activated user is someone that has not only downloaded the app or gone to the website, but they've actually signed up and they've performed a particular type of action like maybe making their first tweet in Twitter's example or something like that. This is hugely important because if you get a million people to download your app, but only 500 of them actually get logged in and activated. Well, that's not so useful. Now, the second one is retention and retention is tied very closely to the growth and activation metrics. It's simply a way to find out who's coming back, just how many people were here last month using our product that are now here this month using it again? Some examples of retention metrics are things like a metric called actually retained users. And one other one is called resurrected users. Again, returning users is the number of users that are returning, that are here again, they are continually either week over week or day over day or month over month using your your service or your app, or your product. And then resurrected users is basically something that we use to describe people that we've gotten back. So let's say that there was a user that was using our product last month, and they're not here this month. They haven't been using our product this month, but next month, they do come back because we sent them either push notification on their mobile phone, or we sent them an email and said, "Hey, did you know we have these features?" or something like that, then if they do come back, we can say, hey, that's a resurrected user. The implication there is that they were, quote, dead or they were just gone, so we

are now resurrecting them and bringing them back into our product. So in terms of how those might actually be measured in number sense, let's say that 100 users downloaded our app last month and got activated. Then we see this month that 60 of them are actually here again, this month using the app, we would say that our user retention rate is at 60%. We know that there's something wrong that 40% of people did not come back, those people did not find it a good enough product. And that's how we'd measure it. Then for the resurrected users, let's say that those 40 were now considering gone like we've sort of lost them because they're not here this month. But next month, we send them an email or a push notification. And let's say all 40 of those people come back into our app, we would say that our resurrected user rate was 100%. Because remember, we have 40 people that are considered dead or gone. And now, we are resurrecting all 40 of them, means our resurrection rate is 100%. Hey, by the way, you might be wondering about these time periods, the one month or the two months or that sort of thing, all of these time periods and actual definitions are really up to each company and or product team to decide the definitions on. Basically, the only thing that really matters about your metrics is that you're keeping them consistent. So if you're measuring something one way in one period of time, you have to keep measuring it that way in the future in order to be able to compare it to the past. And you want to keep these things the same for as long as possible. Alright, so the next bucket of metrics is engagement, engagement is going to be one of the most common things for product managers to be tracking that work at consumer product companies. Engagement metrics, we've already talked about it a little bit. It's things like Twitter, how many times that user tweet on average per day, how many times that they like something, et cetera. Engagement metrics are usually tailored per company to encourage a specific type of behavior depending on the goals of the company or your industry, basically. And I'll give you a real world example really quick, I know that YouTube tracks views, they count someone has viewed if they've watched 30 seconds of video, 30 seconds of a YouTube video. Facebook, on the other hand, tracks their video views, I think based off of three seconds, so if someone's watched for at least three seconds. These are different. Of course, Facebook is going to report way more views that way, because it's pretty hard to shut off a

video in three seconds if you're just scrolling past it. But these metrics, as you can see are tailored to the types of metrics that each company wants to report both externally and internally for reasons of quality and accuracy, and also being able to be easily compared to stuff they've tracked in the past. The next bucket is user happiness. This is pretty self explanatory. It's just a way for us to quantitatively say, how happy are our users? So examples of that would be something called NPS scores or number of customers that have written a complaint to the customer service department. And then even something like the App Store rating for your particular app. And like the Android or iOS App Store, NPS means Net Promoter Score, by the way. And it's something that was developed a long time ago, that's pretty ubiquitous from company to company. It's basically a survey that goes out to users that tracks how likely they are to recommend that service or product to other people. You might have actually seen these before. If you've ever gotten a survey that says, "Will you rate us on a scale of one to 10 of how likely we are to recommend our product?" The secret behind this though, by the way, is that, I think anything under a seven is basically counted as a zero. So if you can't get a user to at least say, I'm at least like a seven, eight, or nine, or 10 in terms of a scale of likelihood to recommend this to other people, then you're basically losing out and it's going to throw off the average. This is because companies want to kind of set a high bar for customer happiness. Overall, happiness metrics are some of the most difficult to get because it's not as easy as just looking at a database for number of times someone clicks something. But they are very important. And they are even more relevant in industries or companies where there's little competition, or where people don't have other options, or it's a total necessity. Some examples for that would be a cable provider, or an airline. I know most people out there hate their cable internet service or their internet service at home, but they don't really have a choice because there's not very many choices available. So the cable companies, they see a lot of use, obviously, but it doesn't mean the customer is happy. And when they say, "How happy are you to the customers?" they're going to say, " Not very, you guys suck." Alright, so the last category of metrics were going to talk about is revenue metrics. Revenue metrics is pretty self explanatory, how much revenue are we making? But there's a lot of different ways we can look at this. And these are very important for product

managers that are working at companies where revenue from customers is a really big deal. Or product managers that are working inside of consumer companies, where the consumers are not paying anything, but maybe they're being advertised to and that's how you generate revenue. An example of a revenue metric is LTV, which stands for Lifetime Value. And that basically means, we pick a period, let's say a long period of time like one year, and we say that's a customer lifetime. And we say over one year on average, how much revenue does each customer generate us? So within that space of one year, we're getting, let's say 300 bucks from each user on average, that would be our LTV. Another example would be a metric called CAC, which is Cost of Acquisition of Customer. Sometimes it's CCA, which is Cost of Customer Acquisition. It's all the same thing. It's basically how much money do we have to spend in marketing or advertising, or even salaries for salespeople to on average acquire a customer? This is very important to track because we want to keep this number as low as possible. One other example that a lot of business to business companies put a lot of value into is a metric called MRR, which is Monthly Recurring Revenue and its counterpart which is ARR, Annual Recurring Revenue. This is basically how much revenue are we making per month or per year and this is of all of our customers combined, all of our users combined. So this is for a subscription service, let's say Dropbox, for example, they have 100 users, and each one of those users is paying $10 a month, well, then their monthly recurring revenue is going to be 1000 bucks. So you can see how that works. This is, of course, a huge indicator of value for subscription businesses like Dropbox. Alright, so let's recap really quick, there's a ton of metrics out there. And some of them are going to be the same across companies, a lot of them are going to be different. But the important thing to know here is that there's different categories of metrics. And the purpose of these categories is to ensure that we're covering what's called the customer lifecycle journey or the customer lifecycle. So I have metrics to know how many people are finding us and then signing up and then using us and then I have a metric to measure how happy they are and then I have a metric to measure how much money are they making us? And finally, sometimes people even add, how much are they referring us to others to account for sort of a growth, like a viral growth. But yeah, we want to cover basically, the entire lifecycle for each individual

user. Now, if you're at a larger company, you might work on a team that is only focused on really one of these categories, or one of the metrics inside of that category. But if you're at a much smaller company, you might be a product manager that's responsible for a growth metric, and a retention metric, and a engagement metric, and so on and so forth. Alright, so in the next lecture, we're going to get even further into metrics. We'll see there.

How to pick good metrics

- Welcome back, everyone. In this lecture, we're going to talk more about metrics. We're going to talk about how to pick good metrics, what makes a good metric in the first place, and then we'll go into a couple of other details about how metrics roll up into other top level metrics. So, firstly, let's talk about how metrics roll up into other metrics. Let's say that you're working at Facebook, and Facebook's overall metric is to get people on the site and leave them there for a long time. You want people to spend as much time on Facebook as possible. Well, that doesn't mean that if you're working on say the photos part of Facebook or the newsfeed part, that your metric is going to be the amount of time spent on Facebook, or maybe even on that page. it's likely that it would, but sometimes it's not the most important metric. Like I've mentioned before, There are sometimes things that we know that we can track on our individual team that makes the overall metric for the company increase. The example that we've used is the comments on Facebook. If we know that people that leave more comments on Facebook are more likely to spend more time on Facebook or one of its related apps, then we might be working on a team that just tries to increase the number of comments on average per person. This is because we know that it will increase the overall metric, which is the metric that the company has a goal for as a whole. So that's it for that part. I just want you to understand that even though a company might have an overall metric, that you as a product manager might not always be tracking that exact metric, you might be tracking something for your team. The second thing I want to mention is the difference between exploratory metrics and reporting metrics. Exploratory metrics are things that you're not always tracking, or always really telling investors, or your boss, or

other people about how they're doing, but they're there, it's tracking an analysis available to us just to go poke around and see what user behavior is like. This contrasts with reporting metrics, because reporting metrics are the things that we are tracking over long periods of time to ensure that our product is doing well and heading in the right direction. So an example of an exploratory metric would be something where let's say I'm working at Twitter, and the overall goal is to increase the amount of time spent again, on the website or in the app. An exploratory metric might be the number of times someone clicks a certain button somewhere because we're not really interested in keeping track of this over time, but if we as product managers with our team, our engineers, and our designers want to kind of try to figure out a solution in order to keep people on the site longer, we might go look at the buttons they're clicking, and the pages they're going to so that we can tell when they go to this particular page, all of a sudden they drop off and they don't come back. So maybe something about that page is bad. And these sorts of things are just metrics, and just data we look at to help us inform us on other decisions. These exploratory metrics are basically just data points that we go in and look at, kind of hunting for a little bit of a clue so that we can know how to potentially build a feature to hit the reporting metric and make that increase. So for the purposes of this course, and for the rest of this lecture, we're going to be talking about reporting metrics. And these are the metrics that you're tracking over time to make sure your product is either getting better or getting worse. All right. So what makes a good metric? Well, let me first say that if you're working at a company or you join a larger company, I bet they're probably already going to have a bunch of metrics per team that they are tracking, and that you can already begin to try to optimize with features and changes and fixes and that sort of thing. But let's say just for the thought exercise that you don't have any metrics to track, and this feature that you've been brought in to work on with your team is completely new. So you have to come up with some metrics. Well, what makes the metrics good? If I pick some metrics, how do I know if they're good? The first thing a metric should be is understandable. And that means you don't need to go and create some crazy metric, like the number of times someone clicked this button twice and then went back and clicked this other thing and did this thing, and then call that a particular metric that you're

tracking. It needs to be something that's relatively simple that you can explain just by telling them the name of that metric. It's usually like something divided by something else to get a percentage, or the number of times this person or this group of people did this thing. It's usually pretty simple. So keep your metrics understandable. That brings me to the second point, which is metrics should usually be a rate or ratio. And this goes back to what we were talking about in regards to monthly active users versus just total number of users. You want to pick a metric that is not something that's going to look like a hockey stick no matter what you do. Imagine if say Twitter or Facebook or something just basically tracked the number of users on their platform or the number of people that have ever signed up. That's pretty silly because maybe some spam accounts sign up, maybe some people sign up and hate it and leave. What really matters is total number of users, and then from there you take the total number of users that have actually logged in in the last 30 days or something like that. And we call that monthly active. So it's a portion of a larger number, and this is the sort of stuff that you want to be doing. Let's give another example. Let's say we're working at Airbnb. And for those of you that don't know, Airbnb is a company that allows you to basically rent out your house as a sort of independent hotel to people around the world using their platform. Would you want, if you're a product manager at Airbnb, to be tracking a metric like total number of nights booked? Well, actually you would want to know that number, but that number is not going to be very effective in knowing whether or not you and your team are doing a very good job. This is because that can be effected by a whole bunch of stuff. It could be effected by publicity and marketing, it could be effected by any number of other things, maybe the economy, et cetera. But what we could track is the average number of nights booked per person in a month, or let's say six months, whatever might work for the data that you have. This is basically measuring a ratio, which is super important because the numbers are much smaller and they are more descriptive of whether or not your product is good. The third thing I want to point out is correlation. You don't want to look at two metrics that are correlated and then assume something from those that isn't true. I'll give you an example. So there's an old sort of metrics joke that basically says if you look at revenue for ice cream sales, and then you look at a metric for the number of people that drown in a pool and die, those

numbers, those metrics, if you looked at them on a graph, they would track each other pretty closely. And you know why that is? This is because people eat ice cream in the summer, obviously, and they swim also in the summer because it's hot out, but you might be led to believe if you're not paying attention, that when people eat more ice cream, more people drown, which makes no sense at all. So you have to be careful about these things. So the next point I want to bring out is that you should pick a metric that is actually able to be changed, and with relative ease, or you should position your metric so that when you do make changes, the metric is changing either going up or down. Here's an example of that. Let's say that you have a particular type of user that let's just say, only has like one or two hours available at all during the average day. And there is no more time than just those two hours in the day. In that case, you probably wouldn't want to try to maximize the time spent in your product or in your app, beyond the two hour mark, or maybe not even measure at all. This is because there's some external force again, in our thought experiment, where the users only have that amount of time per day. In that case, you may want to measure something like the number of times per week that they're using your product, or maybe the number of times that they log in. In that case, you might want to measure the number of times per week that the users are using your product. Are they coming back really frequently? We're not going to care about how much time they're spending per day, because we assume once we've got them near the two hour mark, they're not going to be able to spend more time in there anyways, depending on sort of the nature of the product. One more quick example for that is let's say you're running an eCommerce business and you sell something that people only buy like they just only buy once per month, like 90% of people only buy once per month. Let's say some sort of fancy clothes. In that case, if the users are coming in once per month, and it's very unlikely that they're going to come in more than once per month to use your product or buy something from your eCommerce platform, then it's going to be very difficult to get them to come in more than that. And of course we would want to try, but it's going to be something that involves a lot more than just our product team, it's probably going to be marketing, and all sorts of other stuff. So what might we track instead? Well, in that case, we might just try to get them to spend more

money for that one time that they do shop with us once per month. So let's do a really quick recap. We know that metrics should be easily understandable. We know that people should be able to know what they're about as soon as they read the sort of description of that metric. You shouldn't pick metrics based on an erroneous correlation, like the ice cream and drowning deaths example. You want to pick metrics that actually do impact each other and are not just tracking each other due to circumstance. You also want to have a ratio or a rate rather than just a number that might go up when marketing does some sales or when your company gets a lot of publicity. And we want to make sure our metrics are with relative ease, able to be affected. So we don't want to really try to change a number or metric that our users are very unlikely to behave differently within. All right. So that's it for this one. And next, we're going to talk about some metrics frameworks to actually help you think through the process of choosing metrics and brainstorm a little bit. See you there.

Using the HEART metrics framework: Part 1

- Hey everyone. Welcome back. So in the last lecture we talked about what makes a good metric. And all of that, and the talks before that very useful in terms of understanding metrics. But now you might be wondering, "Gosh, there's so many things that I can pick from, "how do I even get started thinking about what metrics "are important to me as a product manager at a company, "or me as an entrepreneur. "Which ones should I even think about tracking? "And is there any sort of cheat sheet for common metrics?" What we're going to talk about in this lecture is a metrics framework called the HEART metrics framework. And this isn't exactly a cheat sheet, but it does help you think through the process of which sorts of metrics you should be thinking about to make sure that you know what's going on in your product. So this metrics framework was originally popularized by a girl named Kerry Rodden at Google Ventures. And she's a UX researcher and does some really fantastic work. So why should we use the HEART framework? Well, it's an acronym. So if you know it's an acronym, you know it's good. Executives love acronyms. Secondly, metrics are harder than you

think. We've been talking about a lot of stuff, so it's nice to have a little bit of a framework to help you come up with the metrics you should be thinking about. So this again is just a really simple way to think about the customer journey. And by that, I mean, what I've mentioned before, which is the user has to first of all be aware of your product, then they have to come in and download it maybe if it's an app, then they actually have to sign in if you need them to sign in. And there's just this whole journey that you have to get them activated, they have to come back. And then from there, there's a whole bunch of different things you should be thinking about, like how much money am I making off of these users? Are these users actually liking it? Do they like it as much as they use it? And so forth. So what we're doing here with the screen is I'm going to show you a table. And the reason I'm going to show you a table and a real life example of this is because if you look up this HEART metrics framework on the internet, there's actually not too many good sources. In fact, the first Google result is this one how to choose the right UX metric for your product, which is the original post that Kerry made describing this framework. However, there are no graphics associated with it so that's why we're doing this because I want to show you a real life example. So let's come up with a fictional company. And what is really hot right now? Well, we know Uber, the car ride or ride sharing service is a really hot topic right now. Everyone loves Uber. And maybe some people don't, but anyways, everyone knows about it. How about Uber for X? If you don't know what I'm talking about, there's this trend recently where ever since Uber came out, people have been having these on demand location-based apps for pretty much everything, like dog grooming and all sorts of weird stuff. So it's pretty funny that there's an Uber for pretty much everything. What are two other things people like? How about alcohol? Everyone loves alcohol. Then how about lemonade? Everyone loves lemonade. And if you don't, you should get into lemonade, it's pretty good. And I would say that coming from the Southern United States originally where the lemonade is really good. So let's imagine that our fictional company or product here is Uber for spiked lemonade. And for those of you that don't know, spiked means alcoholic. So basically this is going to be an on-demand location app for the delivery of alcoholic lemonade anywhere you are. Sounds pretty good. So the HEART framework works off of the acronym HEART, and they stand for these five

things, happiness, engagement, adoption, and retention, and then finally we have task success. We're going to talk a little bit about each of those. So happiness is how happy is your user. Remember when we were talking about the fact that you may have a ton of usage from your product, let's say if you are a cable company, you know, and the cable companies are notorious for having unhappy customers, but sometimes there's no other choices there's no other options. So if you were working at a cable company for internet, for instance, then you might have a ton of usage, but that's no indicator of how happy your customers are. This is why we need to talk about customer happiness. Then we have engagement. And we've talked about this before, how engaged is your user in the short-term? This is different than over the very long-term. It's not if they're coming back month over month, but are they coming back within short-time spans? The A stands for adoption. So how many users have actually tried your product? Have they even logged in? Maybe they've heard about it yeah, but have they tried it? Have they gotten in there and placed an order? The R is for retention, which we've talked about before when we talked about growth metrics. So retention is, by the way, it could be considered an engagement metric depends on where you're working or how things are organized. But retention is basically, are your users returning over a long period of time, most commonly it's are they returning every single month? Then we have this concept of task success, which is the final letter in this framework. And this is all about you saying, "Okay, there's one thing at least the most important thing "that someone should be able to do with my product. "Are users doing it? "Let's pick a metric for it and find out." So that forms the left side of the table. So then across the top, we have three things. So there's three columns in our table. It's goals, signals, and metrics. Goals is just, okay, what do you want to happen? What is our goal? Do we want to maximize or minimize the number of times a user does a particular thing, or another example is, do we want to minimize the amount of time they wait? There's something that we're trying to get better and better up to as close to absolutely optimum as possible. That's what goals is. The signals column is really simple. It doesn't seem so, but it's literally just, what is the actual thing that we need to measure in order to know if we are getting closer to that goal? The third column is the easiest, and that is the metrics column. It is basically how do we take the goal and the

signal and express it as an actual metric over time. And that would be the thing that we're actually looking at as a number at the end of the day to know if our product is doing well or not. So together the left side and the top look like this. But really quick, I wanted to mention something about this framework since it is an acronym. And that is that as you can see the HEART, the word HEART spelled out, leads it to be spelled out in terms of happiness and then engagement and then adoption then retention, and in fact, if we were to order it in terms of how a customer actually uses our product, it would look a little bit something more like this, which is first, they adopt something, then they have a successful task that they've done that thing we want them to do, then they have other signs of engagement. Finally, they are either being retained over the long period of time, or not. And then the one on the side over the bottom is the happiness, which is just, is a customer happy? And the reason this is the last thing is because you can gauge this at any point in time, you could do it the first time they've ordered something, or you could do it six months later, whatever, it's really up to you, it's a little bit separate. But in any case, I wanted to call your attention to this just because it's an acronym, it may be a little bit confusing, but they just rearrange the letters so that it spells out HEART. Okay, so let's take our Uber for spikes lemonade company example, and let's go down the goals column first and list out what our goals might be. Okay, starting with happiness. What is happiness for the user of our app here? Well, we want to maximize the drinker satisfaction with our spiked lemonade. And the delivery service that brings it to them. So for the engagement part of the framework here for our product, we're going to try to maximize the order value or the number of orders in a shorter period of time. And the way we might maximize the order value for instance, is eventually doing some add on stuff like little umbrellas in your glass or extra fast delivery or something like that. So for adoption, we want to maximize the number of people that have adopted the product. And for us, that's basically going to be the number of people who have ordered at least once successfully with our product. For the retention part, we're going to do the standard, okay, every month we want to have users come back. So out of our total user base, we want the highest percentage possible to come back month after month. If we were able to get 100% of everyone that's ever used our app to come back and do it again and again every single month, that

would be ideal. So that's what we're maximizing here. So task success could be a couple of things. We could minimize the number of abandoned carts and that's the people who put stuff into the ordering cart and then, you know, delete the app or turn it off or just exit or whatever, and they decide they didn't want any spiked lemonade. We want to minimize that. And then we might even go so far as to say, we take pride in how fast you can get the spiked lemonade, not only through the delivery, but how fast you can do at any app. So let's say that we want people to be able to get that in under five minutes from the app, so let's maximize the number of people who complete their order in that five minute window.

Using the HEART metrics framework: Part 2

- So let's go on to the signals column. So what signals, what things are we going to track in order to know about happiness? Well, how about the app store rating? That's a pretty good signal for the happiness of people using the app. If you don't believe me, just go to any app, any popular app, look at all the ratings and people are saying, "Oh, I love this thing," or, "I hate this thing." This is a really good number to judge happiness from. Secondly, we could track these scores from the NPS survey. Do you remember what that was? Net Promoter Score. That's just kind of industry standard way of measuring user happiness through a score, through one to 10 that they say I am very likely or not very likely to recommend this to a friend. So how about for engagement? What signals should we track there? Well, how about the number of orders per customer? Let's say in a period of a day or something. And then how about the order value? So we want to get them to order as much stuff in one, basically order as possible, as many dollars worth of product as we can possibly get them to do. So what do we need to track for adoption as a signal? What is that thing that we need to be able to extract from a database or whatever? Well, we know that we need to know the number of people who have ordered at least one glass of lemonade using our app. That's what we are calling adopted. These, these people have adopted our product because they've tried it at least once. That's the thing we need to track, who has done this at least once. Now, on the other hand, we

need to be able to find out how that compares to people who have not adopted the product, but they have downloaded the app or downloaded and then uninstalled it for instance. So we also need to know the number of people that have downloaded it in total, because we want to be able to compare these two, remember. So the signal for retention is just going to be, we just need to know who has ordered Spiked Lemonade. And we actually also need to know the time periods when they have done that so that we can say, okay, this was in one month and then this other time they did it in the next month. But we're assuming, because this is signals. This is the thing we're tracking. This stuff is in a database somewhere. Databases always carry what's called a timestamp. So we always know when a particular thing happened in a database. So for task success, what should our signals be there? Well, how about the number of people who did not complete an order? What do we need from that? Like what in the database should we track? How about just the people that have added something to their cart and they never actually went and clicked on the finalized purchase button. And if you remember, we were concerned with the people that are ordering within that five-minute window, because we want our app to be super quick to order stuff. So we need to know the amount of time that people are spending from opening the app and saying, yeah, I want to buy lemonade all the way to the time where they have successfully clicked that finalized purchase button. So the last column metrics, this is the easy one. We just take our goal and we take our signal, which is a thing that we were actually physically going to track in the database. And we just say over a period of time, this is the metric. So for the happiness one, we said that we were going to track the app store rating. So for happiness as a metric over time, we just say app store rating over time. What's the app store rating from month to month? That's our metric. Then, because we're also sending out those NPS surveys, like I mentioned, we may also want to say, okay, how many people have given us a perfect 10 score? What percent of total users that have actually filled out the survey have given us a perfect score. Those are two really good indicators of happiness. So for the engagement as a product manager at this company or a CEO, we're going to want to know what is the average order value of all of our customers per day? By that, I mean, let's say we have 100 customers and all of them are placing orders in between $1 and $10. If the

average is $5, then that's the number we're seeing when we pull up our metric screen. And we want to basically move that closer and closer to 10 or further than that if we're shooting for a much larger number. Then we want to know the average number of orders that the users make per day. So let's say again, we have 100 users and on average, like 90 of them are placing one order a day, but there's some power users. There's 10 people that order like five or six of these per day where we take all of that and we average it out. That's the number that we'll be tracking over time. So then for adoption, again, we want to know some sort of metric coming from the goal we have and then the signal, which is a thing in the database we're tracking, what do we know about how many people have adopted this? What does adoption mean? Well, we already said that it is people that have ordered at least one glass of lemonade. So what we're going to do is we're going to take the number people that have downloaded the app in general, and we're going to find out what percentage of them have actually ordered successfully. Notice how we divided those two things. We basically say people that have made successful orders versus people in general. And we're going to call this metric the new users to customers rate, or let's say adoption rate. So then for retention, we're going to want to new our retention rates. That's going to be our metric because we know, again, the only thing we were tracking with the signals column was the, for retention, it was just okay, who is ordering? So what people, maybe we assign them an ID in the database or something. And again, we, that the database has a time associated with it. So we want to know of the people that were here last month, did they return this month? We're going to divide those two and we're going to basically call them the retained users. And now it will be our retention rate. That's our metric. Okay, so task success. If you remember on the goals part of the task success for this app, we said that users should be able to check out really quickly because our app is really easy to use. And if you remember on the signals part, we said in the database, we're going to track how long people spend in the app and specifically how long they spend between the time they basically open it and indicate they want to place an order to the time that they actually complete that order. So for the metric that we're going to track here, it's just going to be the average time it takes everyone that uses our app to complete their order. So if we have 100 users

and they're taking three minutes, four minutes, 4 1/2, some are taking six or seven, we take the average of that, and we use that as our metric. And we try to get that average as low as possible. And then for minimizing abandoned carts, we wanted to know, remember in the signals column, we wanted to know the number of people that indicated they wanted to start an order and then the number that actually completed the order. And we just do the math there, and then we have a ratio of completed orders and we have a ratio also of incomplete orders. And we said, we were going to try to minimize that. So that's it, that's all of the stuff. And if you put all this together, it's just a really big table. So that's why I showed it to sort of one column, one row ahead of time, because it can look a little daunting at first. So once you've got this all sort of filled out, you can just take a look at it. And it's a really good guide to thinking about the kinds of stuff that you should be potentially tracking. However, there are some tips I want to give you on using this framework because there are a lot of open questions I find when people use this for the first time. First of all, people always ask whether or not you can use this for a product or an entire company, or just a small feature. Well, you can use this for anything. This framework, the HEART Metrics framework is very flexible. So you could use it at a small company in order to think about the customer life cycle journey through the entire product that the company has. And you can also use it a huge company where you're a product manager on a team that is only working on a small feature because even features have adoption rates in this sort of thing, and happiness with particular feature, et cetera. So you can use this for either, it's useful for both. Second, this metric framework is used for reporting metrics, not exploratory metrics. Remember exploratory metrics were just data points we have in our database that we can go query like whenever we want to basically find out a little bit about user behavior, but it's not stuff that we're saying to ourselves, this needs to be on a dashboard. So this framework is for the stuff that is on a dashboard, is the stuff that you do report up. You report it to your investors, you report it to your executives or your boss. That's what this framework is for, the reporting metrics. Thirdly, it's important to know that you don't have to use all of the metrics that you've come up with when you're doing this exercise and putting that together this table. The reason is because it's pretty overwhelming. There's a bunch of stuff you'll end up with in the

metrics column where you're saying, "Well, I have to track all of these." Well, it's not always the case that you need to track every single one of these, but it's a good exercise to sort of list out the things you might track. And then you may only say, Oh, I only want to track engagement and I only want to track happiness, but the other stuff like task success or, adoption, I'm not going to track that, someone else will, or it's not important to our company at this time. So don't think you have to use everything, just pick the ones that you really think are very important for your business or your product. So the fourth thing is actually really cool. And that is that the signals column can actually be used as a sort of guide to what you need to do on the engineering side to get a hold of these metrics and put them in the database so that you can track them. Let's say again, that you don't have metrics being tracked and you've got a new product, or you're working on a new feature at a company and you say, all right, we need some metrics 'cause that's how we're going to define success. And that's how we're going to know if we succeeded and you don't have anything. Well, basically you can just do this exercise. You can take the signals column because remember that's the stuff that we want to track. That's the actual data we need. And you can just hand basically that column to your engineers and talk it over with them and be like, here's the signals that I need from the database. And then we basically turned them into a ratio and express them as a metric. But it's a really handy way to show engineers what kind of data do you need from the backend in order to know whether or not you're doing well? So that's it for the HEART Metrics framework. It's a really handy tool, more like an exercise actually to get you thinking about metrics in a sort of a way that falls a customer journey. And on that note, the next lecture also does the same thing in a little bit of a different way. It's another metrics framework and it's called AARRR Metrics, Pirate Metrics. It's pretty common in Silicon Valley. So check it out and I'll see you there.

Using the AARRR (Pirate) metrics framework

- Hey, and welcome back. In this lecture, we're going to be talking about one more metrics framework that will help you out with deciding on which metrics

you should look at tracking for a new product or business. This framework is a little bit lighter than the last one. A little bit lighter than the heart framework we just talked about. It's pretty short and sweet and it's pretty popular in Silicon Valley. This framework is also really commonly used with software as service businesses. So the framework is called AARRR as in like AARRR in the pirate. Kind of weird, right? But the guy that came up with it, Dave McClure, he's a venture capital guy out in Silicon Valley where actually he's a partner in 500 startups which is a venture capital firm. So he knows a little bit about what he's talking about. He came up with this framework himself because he was investing in companies. And he said like these are the metrics that I like to see. These are the ones that you should focus on. The reason it has that weird name is because it's A-A-R-R-R, a five-step process. Acquisition, activation, retention, referral, and then revenue. This framework is pretty popular with entrepreneurs or people doing much smaller businesses. So the general idea here is that for each of these five stages, you pick a metric that you want to measure. And again, just like the heart framework, this is a way of getting us to think about the life cycle of our user. We'll go over each one of these really quickly because we've talked a lot about all of this so far. Just want to make sure that you're aware of this one. The first one acquisition is just like it sounds. We need a metric for us acquiring a user. That means the user went to our website or downloaded our app. One of these, they have arrived. They know about you and now they are indicating their interest to begin using you by taking that step. The second one activation, I've talked a little bit about it already. It just means that the customer that we've acquired is now activated. So they've signed up or they've entered their info on our mailing list, or they have integrated the app with another app that you're considering to be an important step. It just means that they are now active. And you can define that however you want. Some people just say, okay, if they've made an account they're activated. Other people will say, okay, they have to make an account, and then they have to do one small thing so that we know that they are now officially active. Let's say adding a profile picture or connecting their phone book, contact list to find their friends on your app, or something like that, pretty easy. The third one is retention. It means just like we've talked about before they come back, they're repeat users. Notice how there's not so much

emphasis on engagement here. This is because the framework just assumes that you're maximizing engagement in the first place and that if you're having someone being retained well, they're going to be engaged otherwise they wouldn't have come back to be retained. So an example of trying to optimize your retention rate would be to send the user a notification to their phone if they haven't logged in like a week or two weeks, or send them an email drip campaign. When you sign up for those emails and then they email you like once a week, and then they email you like once every two weeks, and then once a month, they're just trying to get you to come back. And the fourth one is referral, which means are users happy enough with your app that you're going to refer to others. And this doesn't just mean user happiness. In fact, it's focused the way this thing is built or the way that it's framed from Dave McClure's mind is that you can optimize it beyond just making the users happy enough to talk to their friends in real life about it. In fact, you could put a feature in where they could share it with their friends and maybe get some credit for doing that. You might've seen this with Dropbox where when you sign up and you invite a friend, you get some free storage, this sort of thing. So referral that's easy. They're going to share it with others. And the last one is revenue. That means you are making revenue from this user, and that is the furthest step down the funnel. So if you're a consumer company that means eventually that they're maybe buying a subscription or they've bought like a pro version or a full-fledged version of your program, your app, or if you're a company that's free to use, then maybe they are seeing advertisements and you can gauge that per user, you're making this much money off of them. The goal here is just to quantify the revenue per user that you're getting. So that's it, just those five stages. And the way this framework is meant to be used is you just use those five stages as a guide. So for each one of the five, those five steps down the funnel, you just think of a metric that you can measure that will tell you how you're doing on each one of those steps because each one leads to the other. You're not going to get revenue from someone unless they are first activated. So you have to get a lot of people acquired. A lot of people activated and then go all the way down the funnel. And you want to get that bottom bucket revenue as big as possible. So you optimize the metrics all the way down. It's just an easy way of thinking about that journey for the user. So

if you want to learn more about the ARRR metrics framework or Pirate metrics framework from Dave McClure just Google it. So if you're in Silicon Valley or making a startup and you want to get some investment money or something, this is a great place to start because these things do matter a lot to venture capital investors and other investors in general. Tip for this lecture. It wasn't very difficult ARRR, Pirate metrics, Dave McClure, good stuff to know. I promise you if these two metrics frameworks, and you tell people about them or you're in a job interview and you mention them or anything like that, they're going to be impressed. I'll see you in the next lecture.

Tracking your metrics in practice

- Welcome back. So by now you should feel pretty confident about metrics, about what they are, why they're important, and a few ways you can go about choosing your own metrics. And likely if you're joining an established company on a team or something like this, you will probably already have metrics to track, and they'll probably already be actively tracked in the back end. So in the case that you're working for all these companies and the metrics already there, you'll probably be working with an insights team or analytics team, or even a data science team to get the metrics that you're looking for to report. But let's imagine for a moment that you actually have no metrics and you've decided what you want to track using the earlier lectures, but you have no idea. Well, in this lecture, I want to talk about a couple of my favorite tools to track metrics and they're really easy to use. So the first one is probably one of the most popular analytics tools out there. It's Google Analytics, and this is a really easy thing to install and will give you all sorts of tracking information on your website, and they even have a component that delivers on mobile tracking. So five stars for Google Analytics. It is awesome. I highly recommend it. Secondly, there's a really cool tool for websites called Crazy Egg. And this gives you all sorts of click data on where people are clicking and that sort of thing, but also gives you these really cool, like heat map things where you can see with colors like a heat map of where people have been clicking and moving their mouse and scrolling. It is pretty cool and gives you a ton of insight. The next one is

Kissmetrics and it is an awesome solution for both web and mobile, and it provides a ton of data, and you can set up custom metrics and do calculations on the fly and look at all sorts of crazy graphs. It's really good, but it's not cheap. So the fourth one is similar to Kissmetrics. And in fact, it's a big competitor. I think it's probably about 50-50 split between the people who like Kissmetrics versus Mixpanel, but it's also not cheap, but it's extremely good to use for both web and mobile and has some really cool features like a calculator, and you can look up individual user behavior. You can even hook it up to stuff like email and push notifications, so you can send users messages to try to retain them. So the fifth one I'll recommend is Optimizely. And Optimizely is actually more of an AB testing tool, but it also provides some really good click tracking and that sort of thing that you can set up to see how many people click where and where they're going. It is really good and it is super easy to use compared to a lot of other AB testing tools out there. I highly recommend it and you can actually just go demo it for free on their website. So the last one I want to mention is one of my favorites and that's because it's not a tool itself for metrics, but it's a metrics hub. So if you imagine you have all these metrics tools out there and they all integrate with either your mobile app or your website through an API, which we'll talk about later, but usually have to put a piece of code on your site or in your app in order to track stuff. And the problem is if you switch metrics tools, then you will usually have to switch out that code and then to make it even worse, let's say you have one metrics tool and you have a year of using that, and you have all this data in there, and then you switched to another one to try it out. Well, now you've lost all the historical data because you're switching to a new tool. Well, Segment solved this problem because it's like a hub, so you connect other services to it, And then you connect your app or your website to Segment, and this way you can basically plug and unplug any other solution, and it will just automatically send the data to the tool that you would like. So it's really good if you consider setting up with Segment first and then trying out a whole bunch of other tools. It even keeps a backup of your data. All right? So that's it for the metric section. And I hope you paid close attention because this stuff is super important as a product manager or an entrepreneur. Everyone needs to know this stuff. So in the next sections,

we're going to get even deeper into the world of product management. It's time to get our hands dirty. I'll see you there.

Introduction to epics

- What's up everyone, welcome back. So in this section we're going to get into the real nuts and bolts of product management and managing a project day to day as a PM. This lecture is a little bit longer but it's extremely important, so I really recommend you pay attention here. So before we start talking about building we need to understand that the things companies build come from an overall vision that's established by the CEO or founder. So the question is, how does this vision turn into things that get built? Okay so we know that the CEO and founder or executives, they have this vision that they want to achieve about where the company is going and really what the purpose is in two, three, or five years. In order to achieve the vision, a company has to have goals that they want to reach to achieve it, to get there to that vision. Like vision the goals are decided by the executives but then they are defined further into metrics usually, like increase new users by five percent this year. So once these goals are established the management teams come up with what are called initiatives. These are things that the whole company is going to do and build in order to reach those goals. An example of initiative would be, "Translate the product into three other languages this year." Underneath initiatives we have what are called releases. Now don't worry to much about releases right now because it's not really the same at every single company, but really what it is is just a grouping of new features or functionality that's released to the users, to the public, on one particular date or time period. We can continue our example and say that, "If we translated the product into three additional languages, we might decide that we have one release in February where all three languages are available to the public," that would be our release. But underneath releases there are features and functionality that get released, the product management world calls these epics. Underneath epics there things called stories and requirements, but we're going to talk about that part in another lecture. Lets finish off this lecture by diving deeper into this concept of what

epics are. First we must start with a statement of things that we're building to solve a problem and clearly show what features or functionality we intend to build in order to do that, this is what we call an epic. We learned in previous lectures about the concepts behind minimum viable products, customer interviews, and wire framing. We use these tools to help us make the right decisions in solving user problems with features. But once we know what we want to actually build, how do we actually get it built? Well that's what all this stuff is going to be about and that in particular is where epics come in. An epic is really just product management lingo for a grouping of one or more features or functionalities that we want to build. It's the fundamental brick that we build with as a product manager and a product team. Just so you know, usually a product team will build out three to five epics during a quarter, like a quarter of a year, just depends on the company size and all that but in general. This really just depends on how much effort the epics going to take and that sort of thing. In real life an epic example would be something like, translate the app to Spanish, or implement photo sharing in direct messages, or allow users to upgrade to a pro membership. This stuff might sound a lot to you like a feature, and that's because it is. One of the reasons that we use the term epic instead of feature is because not all things we do as a product team result in a new feature for an outside user, so the term reduces confusion. There are some other reasons as well but we're not going to really get into it here, it's something that will become apparent as you continue. For instance we could have an epic that is called, migrate our data to a new database. This isn't really a feature for end users, like the people our there using our product, but it's something that internally our engineers might need to do and we might need to do it to make sure we can scale or that we can support more users in the future, so we still got to do it and it would still be called an epic. Epics are also defines as, a piece of work that takes longer than one sprint to build. If there's something smaller that you want to build and it's going to take less time than one sprint then it wouldn't really need its own epic, it would just need a user story. Now don't worry quite yet about user stories because we're going to talk about that in another lecture, but just know that epics are above user stories, they contain features and functionalities and user stories inside of them. So epics are the fundamental groupings of features or functionalities that teams build. In the

next lecture we're going to talk about what they actually look like in real life and how you might go about making one.

Let's get into epic specs

- So now that we understand what Epics are in concept, let's talk a little bit about how they work in reality. Most project management software gives you the ability to create an Epic and then add features to that Epic. This really depends on the project management software you're using, but in any case, Epics always have documentation with them and describes what it is you're building and then why. This documentation is called an Epic Spec Sheet. And sometimes referred to as a requirements document. You can make one pretty much anywhere like Google Docs, your project management tool or anything else that you want. It's just got to kind of be out there in the public for people at your own company to look at. The purpose of an Epic Spec Sheet is to allow anyone in your company to read it and understand exactly what you're building. It also serves as a guide for your own team as you build things. So what do these Epic Specs look like? Well, they usually have main areas, an introduction, the product requirements, the design requirements and then the engineering requirements. As a product manager, you're in charge of it creating and maintaining the whole Epic Spec Sheet and the Epic itself, but you're especially responsible for the first two sections, which again are they introduction and the product section. Let's go through each section of a typical Epic Spec Sheet individually. So first we've got the introduction. This is really just a summary of what the feature or features you're building are for and why you're building it. What metrics are you trying to improve? And you can also put in any links to supporting documentation or other miscellaneous things like marketing plans, legal requirements and that kind of thing. This is also where you can write down what the feature or functionality looks like, some early wireframes and what it looks like in an ideal state down the road versus what you're going to start out building for your MVP, and then maybe what it looks like for a V2, et cetera. So the second section is the product requirements. And here we're talking about the features or functionality that we're developing in detail. We go deeper on what is

required. For instance, we might say that the feature has to be fast, or it has to be available in a certain language, or it must generate a certain type of data and then store it for later use. So the next section is the design requirements. This section is to be filled out by both you and the designer together, but a little bit more emphasis is to come from the designer here since they're the experts. Let's imagine that for a particular feature, the designer wants to display the user profile picture with a really high resolution because they think that's the best design. This is where that would be stated. This is also where you and your designer will attach any early wireframes and sketches or high fidelity prototypes to help the engineers better understand the requirements and what's got to be done here. So now onto the engineering requirement section of the Spec Sheet. This is the portion that is mostly filled out by your engineers after you discuss the product and design requirements with them. Here is where the engineers can write out database and technology requirements or outline the series of things that must be done on the technology side in order to support the new feature. An example here would be developing a certain API end point in order to get the high resolution picture stated in the design requirements, or they could note that they need to work with another team to access their database in order to get the information that's required to make this feature. So let's recap here. Epics always have company documentation or Spec Sheets to inform other teams or guide hours as we build. In most cases, this Epic Spec Sheet will have a few majors sections, including an introduction, product requirements, design requirements and engineering requirements. It's important that your Epic Spec Sheet is available to anyone in the company so that they can easily understand who is building what, like which teams are working on what things and why they're working on them. I also want to mention that the way that companies do Epic Spec Sheets is not the exact same from company to company. Some companies will say, our Epic Specs or requirements docs need to have more information or less. And some companies don't even use them at all, but this is the standard practice. You absolutely need to be familiar with it as a product manager. So we've got the basic foundation of a new feature or functionality or set of functionalities with Epics and Epics Spec Sheets. Let's find out in the next lecture where we go after that.

User stories and acceptance criteria

- Hey. Welcome back. So, now we're going to talk about user stories and acceptance criteria, which are underneath Epics. Now we already know that an Epic has an Epic Spec Sheet associated with it to describe what we're doing here and what sorts of functionalities we need and all of that. But, we're going to talk about now, how we actually get that into the developer's hands and turn it into work. So, here we're going to assume we've made a decision on what we want to build. We've outlined the phases that we're going to build it in. And we've put all that in the Epic Spec Sheet. Now the task is to put this work into our project management software so that the engineers can start working on it. This is where user stories and acceptance criteria come in. So, a user story is just a way to describe a thing we're going to build that delivers some type of functionality to the end user. An example of a user story is, as a user, I want to send pictures in a direct message to my friends so that I can share my favorite photos with them. That's kind of a weird thing, right? It's a weird format. User stories always follow that format. As an X I want to do Y so that I can Z. So why do we do this whole weird formatting thing? Why are we describing work to be done in this weird way? Do you remember how product managers are responsible for the "what" and the "why", and the engineers and designers are responsible for the "how"? That's the reason that user stories follow this weird format. Because the product manager is usually the person that writes them and puts them into the product management or the project management software. By writing them this way we can avoid saying how something should be done on a technical level. In other words, by writing in this format, we're not telling engineers how to do their jobs, we're just telling them functionally, as a user, what we want the user to be able to do. Okay, so where do we write user stories, and how do they get to your engineer? Well, they belong inside your project management tool. And no matter which software you're using for project management, there's probably some kind of place or card or something where you can write the work to be done. And then you can change that card or that place into different statuses like to do, in progress

and then done. These little cards or whatever you want to call them they're commonly called tickets in the product management world, and it's where your user stories belong along with proper designs from the designer. So, engineers take the user stories, they complete them, and then they move onto the next one. However, before we can say that a user story is complete, we have to make sure that the functionality does what we intended it to do. This is where acceptance criteria comes in. Acceptance criteria is just a set of conditions that software must satisfy to be considered complete. Typically, acceptance criteria is written inside of the ticket with the user story and sounds a little bit something like this: Given I am a user, and I click the add picture button in the direct message, I am presented with a popup window to choose the file I can upload, submit it with the upload button and see a preview of the uploaded image. And then here's another one: Given I am a user who has successfully uploaded a photo from my computer, when I click send, the image is sent to my friend through the direct message, and it appears in the chat. This sounds really specific, right? Well, the purpose of acceptance criteria is to be very specific on how a feature should function. It provides a list of things that you can test to make sure it works before you release it to the public. This is also extremely helpful for engineers who like everything to be very specific, so that they can think through the process of engineering this stuff correctly. As a product manager, you are usually responsible for testing completed tickets and stories before approving them to go to the public or to go out live in your product. So, that's it. Tickets inside of project management software have user stories that describe what a user should be able to do, followed by some acceptance criteria that really spell out exactly how it should work. Okay, look if all of this is a little bit confusing to you, don't worry, I'm going to show you a real-life example coming up. I will see you in there.

Real-life examples of epics, specs, user stories, and the backlog

- Hey, everyone welcome back. So if you were left a little bit confused from the last things we talked about with the epics and spec sheets, user stories and acceptance criteria. I'm going to show you a real life example here inside

a tool called Jira. J-I-R-A And this is not the only tool you can use, in fact, I don't recommend it unless you work in a very large company, but it is very common at large companies. You can also use a whole bunch of other tools we'll talk about, but I'll show you in Jira because this is kind of an industry standard. So let's get into it. Let's go down to my screen and I'll show you the real life example. All right, so what we have here is a screen, that is inside of a project I've created. And I'm going to pretend that this project inside of Jira is basically the project that my team owns. I'm a product manager and I lead a product team and this is the project my team owns. And so what you can see here is, this is what is called... I'll expand this so you can see this better. This is called the backlog, and the backlog is merely just a place that we hold, things that we plan to do later, but we're not working on quite yet. If you remember, we do sprints in the scrum model. So the stuff that we're working on right now, it goes into the sprint and we start working on it. And then when the sprint is over, we see what we got done and what we didn't. And we add more stuff from the backlog to the next sprint and that's how we work. The backlog is just a place to hold stuff that you plan to do. So what I've got here is an empty sprint at the top. And then here, I've got my backlog. And over here on the left side, you see the epics. So like I said, most project management softwares like, Pivotal Tracker or Jira, they going to have places for epics and in other ones you might just have to create some sort of structure yourself. But here you can see that I've selected all issues, and that's basically all of the things in my backlog, all the user stories, basically that I want to do, but may or may not already be inside of an epic. And then here we have one epic, which we've called the Direct Message Photos v1. And then I've got another one here that's a technical epic, that's not user-facing and it's Migrate User Database. We can see here in the backlog, we've got four tickets here that, have user stories inside of them that belong to this epic. And I can click on this, and it will only show those. And then below those, I've got the tickets for migrating the user database. So let's go ahead and add a couple of these things to our sprint or pretend like we're going to start a sprint basically. So we know that we're going to work on this epic. We're going to try to implement direct messaging of photos I'm going to do that. And we're going to also be able- We also want users to be able to cancel a photo being sent in mid transfer. And then there's some other things

here where, okay, users can maybe upload the photos from a URL and they can also upload from copying pasting. But that's going to take some more effort, so we're actually considering moving that, to maybe another epic where it's the "Direct Message Photos" version two, or it's our second phase. But for our minimum viable product, we just want to allow people to upload photos from their computer. So that's what we'll start with. I'm also going to add, maybe the first phase or first ticket here from our migrating the user database epic, into our sprints because maybe we have, two engineers that are working on the photo indirect messages, and another engineer is working on migrating this database. So what happens if we've got our sprint loaded up here, and we click Start Sprint. And it says, how long is the sprint? Because not all companies use two week sprints, but it does pre-fill with two weeks. You can choose some other options here too. I'll click Start. And this brings us to the active sprints area. So you can see, we have items here that are To Do, then we have In Progress, then we have Done. And we're going to start, let's pretend like we're an engineer and we're working on stuff. What's going to happen is, they're going to say, all right, I'm working on this as In Progress, and eventually we move it to Done. But what are they actually working on? Let's move this back here. And let's click on this first item. And you can see that it's part of an epic here, which is the direct messages with photos. And, I will just go in, juzzle a little complex so bear with me here. This is why, not many people like it. But we'll go into the more details for this actual ticket. So inside of it, we can see that this is a ticket, and the ticket is Send photos through direct message. Inside we've got a user story. So as a user, I want to be able to send pictures in a direct message by clicking the photo sharing button so that I can share my favorite photos with friends. And we have those acceptance criteria talked about. So given I'm a user and I click the add picture button in the direct message, I'm presented with a pop-up window to choose the file I can upload, I can submit it, with the upload button and then I see a preview of the uploaded image. This is also a place you can see, where we've got a wireframe in here, right? So this is where, we would implement or put in the wireframes or mocks where it shows the designer has said, okay, this thing is this many pixels wide and this many pixels tall. So those go into the actual user stories themselves. But I want to show you one more thing, and that is, because this is a ticket, has a user

story in it, it's part of an epic. So what does that epic look like? Well, we can click here and go to the actual epic itself. And this is just the inside of that epic that we saw on the backlog screen earlier. Basically, it has those sections that I was talking about. So we have an introduction, and it's like, okay, this feature is going to allow users to send photos to each other through the direct message. It says, our research shows that 80% of people would use this and we tested it with people in user tests and they really liked our designs. And basically we say, okay, ideally in the future, we want to be able to share animated gifs as photos as well, and maybe paste photos from the clipboard. But for this MVP, we're only going to allow uploading from a device. That's just to kind of get the base level in there without doing too much work yet still providing some functionality for the users. Notice that we're also saying, we expect this functionality to increase the number of messages sent per user by 15% on average, and, you know, this results in a 4%, time in-app for each user, over the course of a week. The increase is by 4%. This is all stuff that you, as a product manager responsible for figuring out ahead of time using data and then writing in here in the epics. So then we've got the product requirements, okay. A user can initiate a photo message from the message window. A user can select a photo from their own device or connected device. A user can preview an image before sending. We want users to be able to send very high resolution images. This is all stuff that you're going to think, okay, as a product manager, according to data, this is what we're going to need in terms of functionality here, for this to be successful, through all the testing, user interviews we've done, and that sort of thing. And then you and the designer work on the design requirements. But a lot of it comes from the designer. They're saying we should store the photos on our servers so that the users can always see these pictures even if they go to like another computer, they want to look at the old pictures they sent. And this is something that UI designers and UX designers, they're thinking about the user experience. So they're going to, say things like this, like it's important for them to see it in multiple locations because they think that is important as a UX designer and that is up to them. So then we have, okay, maximum photo size should be 5000x5000 pixels. The designer thinks that's very important. Previews should be this size. And then also they're thinking about that actual experience. So once the

user successfully uploads, then, we show a success indicator. And then we have the engineering requirements. So the engineers have looked at this with us at the epic spec and we've talked to them about all of this functionality, they've read the design requirements and they come in and they write some additions. They're like, okay, for this, we're going to need a photo database 'cause we don't have one yet. It's going to be able to need to scale to a hundred million images, with a maximum file size of 10MB per image. So they're thinking ahead in terms of how much traffic they can support and if it's scalable. We're going to need to pull user IDs from the user profile table in order to associate that message, and that photo and that user together. And then, they're going to ensure that the content delivery network is optimized for this. So we can get pictures to people, over in Europe really quickly. So, so on and so forth. You understand just by going through this. And then you see, we also have some wires in here. And usually this is, some of the earlier wires, and then in the actual tickets themselves, we've got the very specific wires around, okay. The pixel widths and all that, like I mentioned before. And then JIRA allows you to comment down here, and it shows you the issues, they call them issues or tickets, in the epic. Okay, so really quick, let's recap on this. I'm going to go back to the work board. This is we're in a sprint and this is what's going on. But we'll start off in the backlog. We've added stuff to this sprint. The sprint is right here. The backlog is right here. This is stuff we're doing in the next sprint or the one after or just down the way. And then you can also reprioritize stuff on the backlog based on, you know, things that happen like, is this now more important than this? And this is something that you as a product manager are also responsible for is the prioritization. We're going to talk about that later, but you can do that in most project management softwares. So then once we're in this sprint, we can go to the sprint that is currently active, and we've got to do In Progress and Done. The engineers come in here they click this They're like, okay, what is this? Okay, here's the user story. There's acceptance criteria. And they're like, all right. And they put it In Progress and they start coding and then they go to Done. And in some cases in some companies you'll even have a step before Done that is called QA or testing. QA stands for quality assurance. And this is sometimes where engineers move stuff before it is sort of approved. And that's usually a phase when product

managers and designers will go through the acceptance criteria and literally use a test version on the test device or a test website and say, okay, is all the stuff working like we said it would. And then they go, okay, it is approved. And then they move it to Done. And then after the two weeks, you should be done with all of the things that you have in your sprint. So that is really, it. That's an overview of typically what an epic looks like. It's just kind of a container, a grouping of stuff, but you have an epic, a spec with it or requirements document that tells everybody about why you're doing this, what you're expecting. And then you've got the requirements from the engineers and designers, and then you've got the tickets, that contain those user stories and acceptance criteria, all of this stuff combined together, it makes the actual day to day workflow of engineering and product management together in a product team at a tech company. All right, everyone, I will see you in the next lecture.

Estimations and velocity

- What's up everyone? So I want to talk really quick about how we estimate the time it's going to take to do a particular software project or how to figure out how long it's going to take to have us build something so that we can say, Okay, at this point in time, we going to have this. This is very much related to road mapping. So I want to bring it up before we talk about roadmapping. First, let's talk about a real world situation. And this is going to help you understand software estimation a little bit more. Okay. Imagine that you're taking your car to a mechanic, and you say to the mechanic, I would like you to replace my engine. I just want you to take it out, put another one in. Well, if your car is, let's say, a particular brand, and let's say it's a BMW 'cause you're awesome and BMW and is the best car out there. I think and the mechanic, he's a BMW mechanic. So odds are that it's a BMW car, BMW mechanic. He's probably done this 100 or 1000 times in his career, he's going to be able to say, Alright, well, that will take me six hours or maybe one day, and I'll have it ready by tomorrow. This is something that's probably very true is very easy for someone who's done something 1000 times to predict how long it's going to take. Now imagine that everyone on the entire planet

has a different car, and that all the mechanics are not specialists in any one particular car brand because everyone's building the cars themselves out of stuff they found at home. It's like, everyone has a custom car. Now, if you bring your car into a mechanic, do you think they going to be able to give you an accurate estimate on how long that's going to take? All right, well, the answer there is no, because, okay, they've never seen your car before, let alone worked on it. And it's going to take them a while to get in there and see how you built your car. Is your engine like a steam engine or a gas engine or an electric engine? Or like, what is it? And if they try to give you an estimation, like, Okay, this is going to take six hours, they're probably going to be wrong, the best they could do is give a really conservative estimate and just say like, this is maybe going to be like a week and then I'll look into it. And if it looks like it's going to be easier, then maybe it's going to be a little bit before then and I'll call you and let you know.. So why is this analogy appropriate for software development? It's appropriate because this is the way software actually works. You have the engineers who are pretty much the mechanics in this situation and then you have the things they work on the systems and databases and code bases that they work on, which are almost always completely unique from company to company. The engineers are building things in different ways in different languages and styles, depending on the needs of the company. And to make it even worse, that stuff, those code bases is changing all the time. So one day, I could be an engineer working on a code base and then I come in next week and try to work on the same code. And four engineers have edited it in between that time working on some other feature. So things change and I have to get used to it. This in a nutshell is why software estimation is very hard. A lot of people say oh, engineers are terrible at Estimating Software like deadlines. This is it couldn't be further from the truth. They're actually quite good, given the complexity of software. But the fact is here that you you really can't really say that, okay, we know for sure that from one estimation from one engineer, that this thing is going to be done on this date. So how do we get around that, the way we accurately or at least semi-accurately estimate software is by figuring out what is called Velocity. So this goes directly into what we're talking about with Sprint's. Now, I'm not showing you this inside a project management software, because the way that Velocity and all this stuff works in different

project management software's, it differs from software to software. So instead, I'm going to explain the concept to you so that you more clearly understand the theory behind this. So let's say we have a sprint and we are doing five things. In this sprint, we're doing five user stories that are a part of an epic, what happens is, before we start a sprint, we do something called a sprint planning meeting or a scoping meeting. And we asked the engineers, how long is this going to take for this thing? And this thing, we ask a few of them, not just one, we ask a couple engineers on the team, and they come to some conclusion they say it's going to take this amount of time. The trouble is if we ask them how long it's going to take it still a little bit inaccurate. So the trick is, we say, How hard is it to do this? And the way we measure a way we quantify how hard something is through numbers. And these are called Story Points. So each story has some number assigned to it by an engineer or a couple of engineers. And they represent these story points they represent the difficulty. Now, some teams and some companies use different scales for story points, you could have a scale like 12345, where five is the hardest and one is the easiest. Or you could have like a Fibonacci sequence where it's like, one, two, and three all the way up to like 21. You could even have like one through 100. It doesn't matter which scale you use. What matters is that you're consistent with the scale. So you're always using the same scale and all the engineers understand that okay, five means super, super hard and a one means very easy. So what you do here is you estimate you give all of these things in your sprint, a story, an amount of story points, basically. And then your sprint has what is called like the total story points for that sprint. Okay, so at the end of the sprint, we see how many things we actually did get done, how many story points were the developers able to actually complete. So if we had five items, and we only got three done, but those things had a combined story point value of let's say 15, because they were as only three things we did, but they were very hard. So as five points each and we're using the one to five scale, then what we have at the end of the sprint is what's called a Velocity. Net velocity now is 15 It's the number of story points we were able to accomplish in a one period of sprint like a one, two week sprint period of time. All right, so now you're wondering like, Okay, well, how does that help me? Does that mean that I can do 15 points in the next sprint. Well, kind of, but I'll tell you how we do it even

more accurately, what we do is this process over many many Sprint's so back to the analogy, the mechanic that first mechanic he was able to make an accurate estimation because he had worked on like 100 or 1000 of these exact types of projects before. Well, if we take our sprint velocity from one sprint, and then we go through, let's say five more sprints in the future, we take those velocities, we take all these velocities and we average them out, then you have like a basically a roundabout way of seeing on average, how much work can you get done over a particular period of time. So you have an average sprint velocity of let's say, like 12 or 16. And then what you can do is say, all right, in the future, we want to do this thing and we're talking to the engineers and they think it's this many story points in order to do these things? Well, we can extrapolate out and say that according to our past average velocity, this is going to take us one month or two months. And in the end, this ends up being pretty accurate, not 100% accurate, but it's the most accurate you can get for software development. One more thing to mention here is about that scoping and that estimation from the engineers. Usually this happens earlier before you start a project. And we'd like to do it before we start a sprint, in either a sprint planning session, or even in the middle of the previous sprint will talk about what's coming up in the sprint after that, and we will have the engineers and the engineering manager talk about, Okay. How many story points are each one of these tickets and user stories? Alright, so that's it. That is really engineering software estimation, in a nutshell and now you've learned about velocity. And this again is why Scrum is pretty useful because on Kanban, it's a little bit more difficult to predict this sort of stuff 'cause you don't time-box your work. Like we mentioned before. That's why Scrum basically has these time boxes so that we can say, okay, over a period of time, we're able to get this amount of work done, and it contributes to accuracy in the long term. Alright, we'll see you guys in the next lecture.

Roadmapping

- Hey everyone, what's up? Now, we're going to talk about roadmapping. Okay, I'm sorry in advance, but I'm not actually going to show

you a real life roadmap right now, and there's a good reason behind it. This is because every company you go to, every single company out there does roadmapping differently. Furthermore, roadmaps are always, if not, mostly always, just inaccurate. So, we know about how agile works. So, from what we know there, can we think about why roadmaps are inaccurate? Because if you're doing things in agile fashion, do you think you're going to be able to predict three to four months out what we're going to be doing at that point? The whole purpose of agile is to say that we're going to try some stuff right around this period of time. This is what's coming next for sure because it's a big user problem. We're going to try to solve it, but we may solve it now with this initiative, or maybe we don't and we get rid of that, and we do something else to solve that. So, these things can shift around so if you ever see roadmaps that say, "in quarter one, we're doing this, and quarter two, we're doing this, in quarter three, this is what we're doing," it's okay to have those things as a general guide in terms of what might be coming in what parts of the year, the first half, the second half of the year as a general guide, but it's not going to be accurate down the date that you say it is, and if it is, then you are not being agile. So, why do you ever see roadmaps that are like, "in Q1, we're doin' this, in Q2, we're doin' this, in Q3, we're doin' this"? It's because, well, for a couple reasons. One, executives, and other departments, and investors, they like to see these sorts of things, but the good ones will realize that stuff that's three-quarters out there is not going to be accurate if you're being agile, right? It could be that it takes us a whole quarter to figure out one problem through a bunch of different user tests and solutions, and it moves stuff back, or it could be that something becomes a nonproblem. Maybe we were going to build something to compete with another company and that company goes away or they go bankrupt. Oh, okay now we don't have to do that anymore. The second reason, you sometimes see these roadmaps that are Q1, Q2, Q3, that sort of thing is because sometimes, and especially in business to business companies, you actually are going to be against an actual real world deadline. So, large companies will do this a lot because they have investors that say they have to have a certain thing, certain functionality to compete with somebody done by a certain date or maybe, you're at a Software as a Service company, and your sales people sell a new feature that

you haven't built yet to a customer, and the customer is like, "We need this by February." So, at that point, you can put something like that on a time-based roadmap and you got to stick to it. You got to figure out how you can do it before that time, but I just want to let you know that these types of roadmaps are generally inaccurate, but it's okay to still look at them. Just keep in mind that they're inaccurate. So, what's the alternative? What do we do instead? Well, what a lot of product managers do, a lot of product development teams do out there is they just basically sort things into priorities like here's what we're doing now, we'll have a column, and then, okay, this is now, this is what we're doing immediately in the next few weeks and next month, and then, they have another column that's mid-term. Here's the stuff that we're going to do after that. And then, they'll have another bucket that's called long-term. This is stuff that we want to get to eventually, but it's not huge priority right now. So, those three columns, the near-term, mid-term, long-term, are usually pretty helpful because intentionally doesn't say any particular dates on there, and it keeps everyone in line by not expecting things by a certain date. This three-bucket model is something you're going to find a lot at consumer tech companies, especially ones that have the luxury of building products that, well, they don't really have any deadlines, like Google and Facebook, they, and even Twitter, they're not really having any financial trouble. So, they can really take as long or as little time as they want to do something. They don't really have to put it into a certain quarter, certain date, or certain month. So, you'll se a lot of these things at the really big money high tech companies. So, that's what I wanted to mention about roadmapping. Don't worry, we're going to put plenty of resource guides and articles to read about roadmapping because you do need to understand the in's and out's around it, but those are the two major things about roadmapping. One, the quarterly or monthly roadmaps are okay to make and show as long as you are on the same page as everyone else and you're telling people, "Look, this stuff may change." And then, two, the more appropriate way, or the more product management, or Lean way, or agile way is to do sort of a near-term, mid-term, and long-term bucket. All right everyone, I'll catch you in the next lecture.

Prioritization

- Hey, everybody. What is up? So we know already about epics and epic spec sheets and we know about user stories. So how does all this stuff get prioritized? As a product manager, like we've already talked about prioritizing things is a major part of your role. But here's the thing, you don't just prioritize user stories, like a lot of people would have you believe. You're in charge of prioritizing things all the way from user stories up to bigger things like epics all the way to even you can help marketing prioritize, you know, marketing and branding campaigns based on product launches. There's a whole bunch of stuff you got to consider. And it's more than just the value of that particular thing that helps you prioritize it, because sometimes you're working in companies with other teams, and different technologies and other times the competitors will do something, all of these things have a big effect on how you prioritize. That being said, I want to talk about three quick methods. They're very basic academic methods that a lot of people know about in the product management world to prioritize either user stories, or epics, or just initiatives that you're doing. The first thing I want to mention is there's something called assumption testing, prioritizing by the assumptions. The goal here is to pretty much de-risk what you're doing by saying, "Okay, this epic or this story, this user story, we're assuming a few things here. We've tried to remove the risky assumptions through user studies and tests and data and that sort of thing, but you can't remove them all. So what are the remaining assumptions for any given epic or story? And how big is the assumption that you're making? Or if you're making a few assumptions, let's see what the biggest assumption is. What you do here is you just take that biggest riskiest assumption and you will assign it a value from like one to five or one to 10. And you say that, you know, 10 is really, really assuming something and it's quite risky. Then you take a score called an important score. And you also score it on the importance of doing that thing or the impact you think it's going to make. You just add these two values together, and then you sort based on that. So you do that for everything. And you just sort based on that. This works particularly well, on smaller stories inside of a bigger epic you're doing is what you're doing

basically is saying, "All right, if we are completely wrong about this one thing, "we want to try it first. "We can't get any more data, "we can't figure out how wrong we are until we do it. "So we're going to try it first to know "if we're right or wrong. "And like get us some more information on the next things "we're thinking about doing." All right, the second method is called the BUC method. This is the business benefits, the user benefits, and then the cost. Basically you want to come up with a value from like one to five or one to 10 for each of these, score them appropriately add up these scores. And then you basically take the benefits part, the business benefits and user benefits, subtract the cost score from the sum of those, and you get a final score, and then you prioritize all of your stuff based on those final scores with the highest being up at the top. There is a problem with this though, and that is that it's pretty challenging to accurately like score things based on these sometimes there's sometimes There's just not enough data available to make an accurate number out of it. Alright, so there's one last method I'm going to talk about. This one's quite common called the MOSCOW method. It's the M-O-S-C-O-W. You might have seen it spelled like that. This stands for must, could, should and would. Basically take everything you've got everything you intend to do, all these stories or epics, and you just say, "Okay, we're going to imagine "we're not going to do any of these. "What is the worst thing that happens? "Like, what's going to happen? "We do none of this." So from there you find out things that okay, we absolutely cannot live without building this for a number of reasons. And you can call those things, the musts. Then you go over all the other ones, and you just say, "is this something that we should have? "Or is it a must? "Or is it something we could do or something "that we would do if we got the opportunity?" This is really not science. It's just going through things and then thinking very deeply about each one and saying, "Is this something we absolutely must have or things "are going to get really bad, "either with the business in terms of revenue or survival "and that kind of thing." Last thing to mention about that Moscow method is that, if you have something that is a must have, you can obviously not have that depends doing something else that you haven't done yet. So if there's one feature that depends on another feature being there, and that first feature is a must, well, then it makes sense that the other feature also must be a must have. Alright, so that's it for those three

methods. We've got BUC, we've got Assumption Testing, and we've got the Moscow method. Those are useful, but they're not going to be applicable to everything you're prioritizing as a product manager, because like I said, there's all sorts of crazy things happening out there in the market. There's companies that go bankrupt and companies that get acquired, and then you've got, you know, other teams that are working on things that you didn't know about. Or maybe there's a new technology available all of a sudden that allows you to do something. All of these things affect priorities everywhere in the business, not just on small user stories, or epics, but on the direction and the vision and the direction that the entire product team, all over the company head stores. Keep these methods in mind. But just remember, they're not the only thing you can use to prioritize things. You can also think for yourself and go by your instinct with a data you got. I'll make one last note, by the way, and say that some of the biggest successes out there in terms of products were not decided on some sort of fancy, logical, quantitative decision, but they come from companies that engender a risk taking attitude or have structures in place to let the employees take some risk. If you work at a company that is not going to go bankrupt if you screw up a little bit. Work with your manager and your bosses and the CEO and the executives. And just think about ways that you could be a little bit more creative with your time or even with some of the other teams to come out with ideas for products that have ever been created before, go a step further. It's okay to do that. But the main goal is not to bankrupt your company by screwing up on a really important feature. Alright everybody see in the next lecture.

General communication skills

- Hey, what's up? This section is an important one. This is all about working with people and stakeholders. It's communication skills and that sort of thing, so let's get started. By far, one of the most important skills to master as a product manager is communication and you already know why, because being a communication hub is one of the biggest parts of our job. Communication happens in a lot of different channels so let's talk about a few here. So first, we've got meetings. This is among the most important

areas a product manager needs to be a great communicator in, just these company and one-to-one meetings. So much of what affects our product occurs in meetings and knowing just how to take advantage of the face-to-face and internet meetings is a key portion of the product management job. We also have to learn how to make sure that activities that come from meetings are followed up on. If you send a recap email or note after a meeting, and you ensure everything gets followed up on from all the people in the meeting, you're going to score major points as a product manager. Number two, conference calls. And this includes video meetings as well. If you've got offices all around the world, these are going to be very frequent. Communicating here very well is dependent on the intonation and the clarity of your verbal message. Ensure that people are following along with you and if you have any resources, or you're sharing some slide decks or stats, make sure there's links to all of that in the meeting invite or in an email that's associated with the meeting. Alright, number three, emails. This is going to be the bulk of communication as a product manager, your primary method. Ensure that what you're writing is clear, that it's concise, and you have links to all the data and resources that you refer to in the email. Number five, informal meetings, like lunch meetings. These are casual environments that can be great for connecting, getting feedback, ideas, and working to build support. Don't underestimate the power of having lunch with a coworker and to tell them about what you're working on and get their input. So in general, there's a lot of ways we communicate as a product manager but to become a truly stellar product manager, you must learn the art of communicating differently with different types of people. There's an old saying that goes, Six different audiences, six different slide decks, we're going to talk more about that next. I'll see you in the next lecture.

Working with engineers

- Hey there, good to see ya again. Let's get into the details on how to communicate with different types of people, and we'll start first with engineers. So engineers are lovers of detail. When talking to them, emailing them, or anything else, be very detailed about the thing you're

describing. This is because engineers are always thinking about software and how what you're saying may or may not work in a technical sense. The clearer you are here, the better. I can't tell you how many times early in my career I was not detailed enough in my talks with engineers or while sending them feature specifications. In order to develop efficiently, they need to know things that other people don't, things like what happens when the screen is displayed in another language. What is the error state read when something goes wrong on this page? What happens when there's no data to see on this page? This is often referred to as an empty state. As time goes on, you'll get better at predicting what types of information the engineers need from you. So now I'm going to list off some miscellaneous tips for working with engineers as a product manager. Number one, if something goes wrong, it's your fault. This is not just being nice; this is the truth. You're responsible for the success of the product. If something goes wrong, or an engineer develops something incorrectly because you gave them the wrong specs, that is your fault, not theirs. Number two, when you're pitching a feature, be prepared and have really good idea of where the feature's going to go in the future. When we do agile development, we think of a feature, and we think about how it's going to look down the road in an ideal state, but also what the minimum viable product will be. Telling developers where you think something's going to go down the road is going to help them plan out how we're going to do this technically in the future, and that matters a lot. All right, number three, when possible, you should do the work up front on things like checking commit logs or looking up data before you ask engineers to do it. Trying yourself before you ask for someone else's time is a big sign of respect, and developers like to know that they aren't the only ones doing the work. This will also score you major points. Number four, watch out for tech debt. So what is tech debt? It's a term we use to describe something in the code or technology which is going to have to be dealt with later as the result of not doing something correctly the first time. Let me give you an example here. If we're building a new feature that requires a database, maybe we learn that one way to save time is to utilize a database that already exists. However, if the feature becomes very popular and it needs more space, then eventually it needs its own database. This impending work is what we call tech debt. We're going to have to do it at some point. Engineers hate tech

debt because they're the ones that have to deal with it. Be cognizant of this and trust their judgment on the right way to do things first. You'll maybe find that a lot of them want to do something scaled out the first time to make sure we don't have to come back and deal with it later, which will stop us from developing features at that point. Lastly, this is a big one, do not treat engineers like an agency. What I mean by this is don't design and come up with all of the concepts yourself or with your designer, and then just hand the completed requirements to the engineers to code. Always give engineers an opportunity to provide you with feedback and product ideas. Just because they spend their days coding doesn't mean that they don't have any good ideas or opinions of their own on the product. The way I give them the opportunity to provide feedback is actively involving them with day-to-day discussion, but also during our planning meetings, I make it a point to have a part where we tell the engineers what we think is coming up next, and we give them the opportunity to tell us what they think about that from the product perspective, not even the engineering perspective, but is this a good idea? Would users use this? It's very important that you bring them in on these discussions. That is one of the best tips I can give you for working with engineers, very important, and engineers will respect you a lot more. All right, so now you've got a good handle on how to interact and communicate with engineers. Let's go on to the next lecture and talk a little bit about designers. See ya there.

Working with designers

- Everyone, so now we have a really good idea of how to work and communicate with engineers. Let's talk about designers. Designers, just like developers, often have certain ways of working that you should be aware of. Here are some tips on working with them. Number one, designers are very creative, and you want them to take a lot of creative freedom in the work that they do. You want them to feel free to really stretch the limits of their imagination. Because of this, it's really important that you're clear about any limitations. Let's say for instance, that we're designing a new profile page for users and the designer wants to put in some data that we don't have

accessible at the moment from the database, and it would take a lot of work to get. It's important that we tell them about this early because one slight change could change their entire design. Keep this in mind. Number two, like developers, you should not be treating designers as an agency or someone who just makes things pretty. This is a big mistake I see tons of product managers make. 90% of the time designers are going to know much more about user interactions than you. Not only that, but it's their entire job to make sure that what they are making fits well within the other areas of the product and the other designers and the stuff that they're doing as well. always provide designers ample opportunity to give you feedback on the core of the idea or the product that you're working on. And I mean, from the very product and idea perspective, not just the wire framing and design part. Number three, you and your designer should function as a team. Again, this means that you should be giving them the opportunity to tell you their opinion on the product in the future. But it also means that you should be giving them data, showing them user feedback, and not just saying that this is a thing we're developing or building because I said so. That's really bad. You need to basically be showing them the reasons that you came up with this being the thing that we're doing next. Number four, don't ever tell the designer what to do. Remember as a PM, you're responsible for the what and the why, but not the how. If you dislike something in particular about the work, feel free to highlight it in broad terms, but don't tell them what needs to be changed. For instance, if you think that something is confusing about their design, remind them that it's important that the interface is as clear as possible. Maybe user research shows that the least technologically adept people are a big part of our user base for instance. At the end of the day, you have the final say on the product to release, but you should never ever have to exercise that power. Okay, number five and the last one here, this is one of the best tips I can offer for working with designers. And it's to always talk about user problems first, and solutions second. In other words, don't say to them that a user should be able to send photos in a message to another user, because then you're just describing the solution. Instead, talk about the fact that the problem that users can't express themselves to one another with photos or media in a message is the real issue. You would be really surprised at how helpful this technique is in finding the best solutions to actual user

problems. And designers really appreciate this approach. Okay, so now that we've got a good idea of engineers and designers, let's talk in the next lecture about communicating with executives and other stakeholders.

Working with executives and others

- Hey there. So we learned already about how to communicate with engineers and designers. But how about the rest of the groups, like let's say executives, sales, legal, and marketing? Well, let's talk about executives first, and then we'll do a few tips on communicating with the other types of stakeholders. In my early days of being a product manager, I always struggled communicating with executives, it was really tough. I would send them really big emails with tons of detail, and they would almost never get back to me. Later, I found out that the secret to communicating with executives, is being brief. Executives are the busiest people at the company. So sending them compact, but meaningful messages, shows that you value their time. So instead of writing two paragraphs, write two bullet points, but call you say, I won't be able to give them enough information in two bullet points, that's right. And they don't need all the information in one email. If they want more data or information from you, they know who to ask, you. Secondly, when talking to executives and other roles like sales, always speak in terms of business effect. By this, I mean, don't tell them that the user will be able to do a certain thing in your product more frequently, but tell them about what it means. Does it mean more or less revenue? Does it mean a change in a specific company metric? If so, tell them that. Thirdly, communicate in a style that they like best. Some executives don't like email as it consumes so much time for them, and maybe even their assistant reads it for them. In this case, a better approach might be to give them a really quick update in the hallway, or at lunch, or at your next meeting. Okay, so here's another couple tips for not only executives, but all sorts of stakeholders in general. Number one, it pays off to talk to people on each team as much as possible. Whenever you get a chance, hold a meeting, take them to lunch, or just find the excuse to talk to people from different teams, like marketing or sales. Number two, ensure that other teams and people understand that you know the user base, the

technology, and the business very well. This makes them much more confident about the decisions you're making in the product. Number three, update them on the latest developments as frequently as possible, but make it clear whether or not they should be replying, or if you just meant to send them an update. One really good way to do this, is if you have an internal company, blog, or wiki, you can just post things there, and they can read it if they want to. But the important thing here is to be communicating frequently. Number four, whenever you're updating outside stakeholders on progress or the completion of a feature, be sure to tell them about the reason the solution is the way it is. Tell them a story about the other options you considered and why maybe those ones did not turn out to be ideal. So tell them that story about how you got to the product or solution that ended up being the one that you built. All right everyone, with those tips you are really well armed to communicate effectively with a broad range of stakeholders. This puts you way ahead in the game. So be sure to keep this section in mind and frequently come back if you need to. It's going to take a little bit of time to get this like really ingrained in your head, but you'll learn over time. I will see you in the next lecture.

Get relevant experience

- Hey everyone. So this is a section all about preparing for a job in product management in order to maximize your chances at getting a job. There's one big problem that's also kind of a benefit about being a product manager. The problem is that it's pretty hard to get the job at first because it requires prior experience, but on the positive side, once you have that sort of experience or you've done it at least once, you have a lot of career security because everyone else has trouble getting that experience, so it's really easy to switch jobs and get promoted and that sort of thing. There's also just a low number of really qualified product managers out there, and the demand for them is really increasing lately. Unlike programming and computer science, there's no way to major in product management in college and say that you have prior experience from that. So how can you best position yourself to get the first associate product manager job or junior product

manager job? What you really need to do is demonstrate that you understand the concept and role of a product manager thoroughly. But just talking about product management is not enough to get a job. You need to have the experiences to point to and discuss with people that are interviewing you. All right, so where can we get these experiences? Well, we can find them in two major places, either in your current job or with a side project. In this lecture, we're going to talk about finding relevant experience in your current job, and then we'll talk about a side project. Since what you do as a product manager is so broad and we just do so many things, I guarantee you that some sort of stuff you're doing at your current job is related to product management. In other words, you're probably already doing it, you just don't know. Think about all of the stuff we've learned in this course so far, and then think about the things that you do in your current job that involve these concepts. I'll give you some examples here. So write down any experience that you have with the following: Number one, think about any time you've had multiple things to do either by yourself or with a team and you have to prioritize what you do first. You made the priority decision based on information, and that's what we do all the time as PMs. Number two, think about all of the times you've talked to any users of a product or a tool internally or externally. If you work in marketing, for example, then talking to the people you are marketing to in focus groups or that sort of thing is very similar to talking to the users of a product. If you work in customer service, then you're definitely talking to users of a product on a regular basis. Number three, think about any times you've given feedback or reported bugs on the product of your company or even another one. Number four, if the company you're with now has a product team, then recall all of the interactions you've had with them and how you may have helped them acquire data to make decisions or gave them feedback on the product. All right, number five, think about any times you have worked on something at your job with multiple other teams or groups of people. Have you ever worked with the legal team, the engineering team, sales, marketing, or customer service? If you've helped teams communicate something between them, this is very relevant to being a product manager. The last thing I want you to consider is any work you've done to improve the efficiency of a process. Product managers work a lot with different processes, so any efficiency you have made better in your job by

finding more efficient ways to do things or improving the way things are done from either stuff you do every day or stuff you and your teammates do, that's going to be very relevant to talk about. So after you've written all of these things down, think very deeply about each one, and then detail each one of them even more. These experience can be put on your resume, and they can be talked about during product management interviews. One other thing you can do is email product managers that you find online and ask if you can ask them a few questions. Use what you learn from them to inform your job applications and interviews. I recommend browsing for product managers in industries you're interested in, in places like, let's say, Product Hunt, LinkedIn, and Twitter. But if you can't think of anything related in your current job to product management, then doing a side project is going to be the most helpful. We're going to talk about that in the next lecture. All right, I'll see you there.

Build a portfolio with a side project

- Hey, what's up, welcome back. So, as I mentioned, the key to getting a product management job is having some sort of relevant experience to talk about and point to on your resume. If you have no experience at your current job and just can't think of anything you've done related to product management at all, then a side project is the best option. So, if you've been following along with this course, especially to the sections that Evan has taught, then you already have a really good idea of how to do a side project on a small product or feature. What you need to do is document your experience of coming up with ideas to solve user problems through a product or feature and put it together somewhere like a vlog post on your own website, a slide deck, or you can actually make your own prototypes really easily, and put that online. These projects can be very detailed or really just high-level. You can even take a product you use daily and describe in writing or graphics how you would change it in order to solve user needs better. This is a pretty common tactic in the world of design, actually. You've probably seen a lot of designers host their own re-designed versions of a popular websites, or apps, or product. And a lot of designers get their jobs this

way. You can actually do the same, but for product management and with features, and that sort of thing. But if you want to do a deep dive into the topic of doing your own project, I recommend two great resources. Firstly, check out the other courses that Evan has taught on coming up with ideas, creating an MVP, or getting started as an entrepreneur. These will set you up with everything you need. Another option if you want to do a step-by-step project, while also learning about product management, I recommend you check out my friend Ellen Chisa's course called The Fundamentals Of Product Management. You can find a link on her website at www.ellenchisa.com. Ellen also writes some really good blog posts on her experience as a product manager, and she's got a lot. So, also be sure to check those out. So, to recap here, doing a side project is a huge benefit if you don't have any sort of relevant experience at all and can't think of anything in your current role that qualifies as product management style experience, in the least, then you should start thinking about doing a side project. All right, in the next lecture we're going to talk about some other things you can do to really prep and increase your chances of getting a job as a product manager. I'll see you there.

Brand yourself

- Hey, what's up, welcome back. The other thing that's going to really help you out immensely in getting a job as a product manager, is having a good personal brand online. What you want here is basically an online record that demonstrates your knowledge of product management and the skills required. Building credibility before you walk through the door to your interviews is worth its weight in gold. The best way to demonstrate credibility is to have a website or blog, where you talk about things related to product management, and have those projects we talked about, available to the public. Recruiters will often look at your social media profiles and website to understand if you're qualified or not for the role. Since there's not as much information out there on product management as compared to other topics, it's actually pretty easy to stand out. Here's a couple example topics that you can consider writing about to get started. Alright, number one,

you could write about why a particular product you use is poor, and then how it could be improved. This is actually a pretty common interview question that they'll ask you as a PM, they'll say, you know, what is a favorite product of yours, and how could it be improved? Number two, you can say, the top five mistakes you've made in your current role and what you've learned. Number three, you could read some product management books, which we will have in the resources, some recommendations for those, and then you could write some blog posts about your reviews of those books, and what you agree or disagree with. Number four, you could write a post on your thoughts on recent technology trends or industry events, and how you see them developing further or changing the market, right? So besides a blog, it's also very helpful to have a public social media account like Twitter, where you participate in product discussions or talk about relevant things to product management. I also recommend you go on Quora, Q-U-O-R-A .com, and ask or answer questions related to products and product strategy. Alright, so a quick recap. One of the best things you can do to make sure you have a great online brand, is to write or just have a public presence. If you don't have a website or blog, you should make one. You should start one, you should start writing, and you should include some topics related to product management and product strategy in there. Also, make sure you have a publicly available social media account like Twitter, where you talk about things that are also on topic with product management, and UX design, and user experience. Alright, I'll see you in the next section where we'll be talking about, looking for product management jobs, once you feel that you are sufficiently prepared.

Where to look and what to look for

- Hey guys, welcome back. Once you've prepared to look for product management jobs, and you've got that good online brand, and you think you can point to some relevant experience, you have to know where to go to look for the jobs and really what you want to be looking for. In this lecture, we're going to talk about three things. One, looking at your own company. Two, Networking. And three, looking online. Firstly, it's important to understand that if you're really serious about getting into product management, then the best

place, by far, that you can look for a job, is in your own company. Trust me on that. It's so much easier to move internally, than it is to get a new job as a product manager, with no pervious experience. If you already have a product management team at your office, your company, ask to shadow some product managers in meetings to learn more about what they're working on. What you should do here is, demonstrate that you're interested in helping them out, and offer to do some work in your spare time, for their team. The goal here is to learn as much as you can. If you don't have a product team already at your company, then you can still start offering to do tasks and projects that will add to your product manager resume. Things like prioritization of initiatives, technical implementations, user reviews and research. And that kind of thing. Secondly, it's very common for people to get jobs based off of references, rather than just applying to them. This is the case with every type of job, but so much more so for product management. But in order to get references, you need to meet people in the product management industry. For that, I recommend you go to websites like meetup.com, and look for product management or product strategy meetups, in your area. You can go to these events. You can talk to people that are product managers. You can find out who's hiring, and maybe even meet somebody who would be willing to provide you a reference for a job, or tell you about an opening. Outside of finding product managing jobs through networking, there's a few places I recommend you look online. Number one, go to angellist.com, which has a big focus on Silicon Valley tech companies, and look for product management jobs. Number two, go to Hacker News, and they have a tech-- it's a technology message board basically. And you can Google it, or you can search on their site, because they have a monthly thread called, "Ask HN Who is Hiring?". Just Google it and you'll find the threads that I'm talking about. They do this every month and it's just full of uh-- openings for product managing positions. The third place that I would recommend is, do a search on Facebook for any local Facebook groups around your city, that involve any type of job, but especially product or technology jobs. These are really good to browse through and something good to keep your eye on in case some lis--listings come up. I've seen a lot of product jobs go through these types of groups. Number four, is mailing lists. And there is some good ones out there. I recommend a

mailing list by a guy named Ken Norton. You can go to, kennorton.com, it's K-E-N-N-O-R-T-O-N, and he's a really well known product manager. He has a great little news letter, but he always has job postings at the end of each email. All right, so those are some tips on where to look for your product management job, if you're looking to get one. And in the next lecture, I'm going to give you a little bit more inside advice, on how to go about your product manager job hunt. I'll see you there.

Inside advice on your PM job hunt

- Everyone before we move on to the actual resume and interview process I wanted to give you a couple of sort insider tips on looking for a product management job. These are kind of miscellaneous tips that I just thought that I should tell you about and pay attention because they will help you in the long run. The first tip is for any product management job opportunity, you should make sure that the company that is looking to hire has a good understanding of the product management role and can clearly illustrate why they're hiring one. I say this because as an observer of the industry, I've seen a lot of companies that don't see the difference between product and project management. Sometimes these companies will end up hiring somebody merely to keep engineers on task and if you can you want to avoid this to maximize your experience for actual product management. The second tip I have for you is to do a lot of research on the current product team or product management leader at the company. Find them on Twitter, look at their blogs, check out their job history on LinkedIn and see what you think about the things that they say. What you're looking for here is someone that you think you can learn a lot from that has had product management experience before. Sometimes someone will be named Director of product at small growing companies, but they'll have no product management experience before that and this is something that trusts me you really want to avoid. On the other hand, if you're interested in working for a particular industry or company down the road, then working for somebody that has worked in that industry in the past is a great thing to know. It's a great thing to be able to say all I would like them to be my boss so I can learn from them. So pay

attention to that. All right. So recap here, do extensive research on the people that you'll be working with and use it to your advantage. Also make sure that the company that's hiring you knows why they're hiring a product manager so that you get a good roll at a good product team where you can learn a lot. I'll see you all in the next lecture. We're going to move into the actual resume and interview process. Catch you there.

Interview for product management

- All right. So you've got your resume optimized. You've got your online brand perfected, and you're going to interview with a company. Congrats, on getting that first interview. In this lecture, we're going to talk about some tips to help you get through the product management interview. These interviews are tough. And that's because it's not all cut and dry. You got to demonstrate some skills, usually on a whiteboard or talking through problems. So what I am going to do is give you three big tips on how to nail these interviews. The first two we're going to talk about here, and then the most important one has its own lecture coming up next. All right, tip number one. Use every interview to make the next one even better. Each interview you do is going to help you a lot in making the next interview even better. Whether it's another interview at the same company or an interview at a different company. You should pay very close attention to the questions they ask you for that product management role. Then think about whether or not you were able to get a good answer to them or not. For every question, take mental notes or even physical notes, if they let you. When you go home, start researching those things immediately. If you didn't understand something, you didn't have a good answer, go look it up. So if you make it to the next round of interviews for that company, then you have more relevant knowledge to discuss. Even if you don't get that job and you are going to an interview at another company, this is going to be super helpful for prepping you for these types of questions in the future. You got to be optimistic here. These are tough interviews. But each failure is an increased chance at the next opportunities. That's how you should look at it. Tip number two. Ask them good questions. The more research you do before the interview about the

company and the industry and the product, the better. You can't do enough, to be honest. Most interviewers will give you a chance to ask them questions at the end of the interview. So you should have some really good questions in mind. Before you go into your first interview, do that research on those latest initiatives and industry trends and even what their competitors are doing. What rounds of funding other people have raised that are in the same industry. Use this information to ask them thoughtful questions. I'll give you an example here. Let's say I'm interviewing for a product management position at a messaging app, and they compete with WhatsApp. Which is a really popular messaging app owned by Facebook. I probably ask them questions about how they tried to differentiate themselves in the market and then what they think the biggest obstacles for them are in the next year. If WhatsApp releases a feature and you know, a month before you do that interview, or anytime in the last six months. You need to say, "Hey, what are your thoughts on that latest feature? Are you guys going to try to compete with that or what?" And this really just shows that you're interested in their product. You understand what's going on in this industry. And I guarantee that if you ask good questions like this. You're going to stand out from the rest of the candidates. All right, let's recap here. Use every interview as an opportunity to make the next one better. Don't cheap out on this. You got to do it. And then two, ask insightful questions at the end of your interview that are relevant to the product management role, the industry and the product. If you use these tips. Your interviews are going to be really good. And you will be way ahead of the competition. You're going to be even better off, if you watch the next lecture. 'Cause it's a really good tip.

How to answer interview questions the right way

- Welcome back to everybody. Like I said, this lecture is going to be one of the most important tips I can offer you for interviewing as a product manager. As you already know, if you've been watching this course so far, and I hope you have, because what are you doing here, if not? Product management positions are based on making good decisions based on data. If someone asks you what you would do for a product in a particular

situation, you never just give them an answer without mentioning how you came to that conclusion and then, why, and then, what are the alternatives? Okay. So, here's a real life example from my actual career. I was once interviewing a candidate for a product management role at an eCommerce company. Everyone's like, Oh, you got to interview this person. She's going to be, she's great. She's got a great resume. Great product manager. So, I'm like, Oh, okay. I'll interview her. We'll see. And I asked her if she thought we should develop a native mobile app or just continue focusing on just doing a normal mobile website like we had been doing. The problem is, without even thinking, I mean, she, she had like a three second delay. She goes, "Yeah, you should do a mobile app, native mobile app." And she didn't give any other details. So, whether or not she was correct with her answer is not the issue. And in fact, it didn't even matter. The problem is that she just gave an answer without asking for more information to help her make the decision. And she didn't follow it up with any reasoning behind how she arrived at that conclusion or how her answer may change based on data. So you're thinking, well, Cole, what was the right answer? Look, the answer I was looking for in that situation was for her to say, "Ah, okay. I need more information to help make this decision. How many of your customers shop for mobile devices? What volume of traffic do you have? What sorts of engineering knowledge does your team have? And what's your budget?" These things help make better decisions. I would have also loved to see her follow up her answer by telling me the reasoning behind it. And then, most importantly, tell me what other options were on the table besides just a, you know, a mobile website or a native mobile app. Because there are things you can do besides that. There's an in between. You can do like a hybrid app or that yeah. There's other options. So, you need to mention these types of things when you're answering questions. As a product manager, there is never a single correct answer to a question. There are always multiple possibilities and it's your job to synthesize information to come up with a good approach and be clear about why this is the best option. We work on data, not feelings. Just like if someone in your company asks you why you're doing a particular initiative or adding a certain feature. You can't just say, "Because." No. You got to say, "Because the data says this, the user interview say this, the market is

going this way and our competitors are doing this. But here's the other options we considered. This one seems to be the best approach for this reason." That is how you think and act like a product manager. So, in interviews, always ask clarifying questions when possible. It shows that you're being thoughtful and thinking like a product manager. By the way, I don't mean to start asking them questions if they ask you simple things, like, "where did you go to school?" Or "Where did you work before here?" What are you going to ask them? "What do you mean by that question?" No. I mean, if you get to the point where they're asking you about your opinions on their product, or what would you do in this situation? There are a lot of tech companies out there, Google and Facebook and Twitter, and a lot of the big ones out there where the entire product management interview process is just sort of a hypothetical situations. They're like, "Hey, what do you think about this?" "What if this happened? What would you do?" You got to reason through it. It's like showing your work on a math test in high school. Man, I sucked at math in high school. (sighs) I don't know why I like this job though. Anyways, that's a really big tip. If you can internalize that, you can start thinking about these things behind the questions and asking for more information to make the correct, or at least, most correct decision on the answer, then you're going to go far. I'll see you guys in the next lecture for some real insider tips on actually getting that job.

Insider tips for getting the job

- What's up everybody? All right, fair warning. This is a longer lecture, but I promise you it is worth it. I'm going to let you in on a little bit of a secret here. This is one of the best ways for getting any job, but it works especially well for a product management role, and works, especially, especially well for getting your first product management role in a junior position, which is what you got to do if you don't have any experience prior. We've already talked about how it's very important to demonstrate knowledge of product management in order to land the job. This is why interviews make you demonstrate this during the interview. But what about outside of that interview? Can you do anything to demonstrate that you really, really know

what you're talking about? Well, if you followed along so far, you're probably already well underway to demonstrating knowledge by establishing an online brand and doing a side project if you've never had an experience at work before. But there is one more thing that you can do to maximize your chances at getting the job. However, I got to tell you that this is not right easy. This is going to require some more time and effort from you. If you aren't serious about getting your dream job as a product manager, and you don't want to go the extra mile to stand out from the tons of other people that want that job, then you don't need to finish this video, and I am super disappointed in you. So go ahead, if you're one of those people, turn off the video, the rest of us will wait here. Go ahead. No, no, yeah, go. Okay, so now that the lazy people are gone, I'm going to let the rest of you in on this secret, but it's not that complex. It's actually pretty simple. Do a demonstration project for the company that you're applying to and send it to the hiring manager and the recruiter. It's like doing the generic project that we already talked about, but it's much more targeted, and that's the key. You're probably thinking like, "Oh, does this work? Is this, what am I, why am I going to do this?" It does work. In fact, this is how I got my first few product management jobs, and I have also gotten a ton of offers by doing this that allowed me to basically choose the company I wanted to work for instead of having to accept the first offer that came my way. So the basic technique here is to find out what sorts of things the company is working on with their product currently, their strategy and that sort of thing. Show them how you would approach the problem if you were to be hired. You can do this by either doing some generic research on the company and finding out through news articles and that sort of thing, what they're trying to build or the challenges they face. Or you can be even more effective and ask them some questions during your first interview with them. And with all of the work you've done so far on this course with the online branding and your social media profiles, and just knowing what you're talking about, it should be pretty easy to get at least a first round interview or a phone call. So here's how I do it, and you know how I did it back when I was first starting out. Let's say that I got a first round interview for a product management role. During the first interview, I would answer the questions, but then I would also ask them why they're hiring, which features this product position would work on and see if they give me any information I could use on

the things that they're trying to build or the markets they're trying to get into, or the clients that they want, that type of thing. So what I would do after that, I would immediately leave, after the interview, I'd immediately go home and I'd do research on the product, the industry, the technology, and the competition around the stuff that they told me when I asked them those questions. Then I would spend at least 24 to 48 hours developing a comprehensive overview of how I would approach that problem. Usually I would spend an entire weekend doing this. "But Cole," you say, "I want to go out with my friends and drink beer on the weekends." Or not beer. I don't, vodka. (shudders) Maybe you should do that instead. But the other people are going to be doing this extra work on the weekends in order to put themselves above the other candidates and show that they really want that job, and they know what they're talking about. So in the past, these projects I've done have ranged from a small four-page PDF document, all the way to a 60-page slide deck. And I'm not kidding. I actually put together 60 slides once for a company. They freaking loved it. These things usually contain things like descriptions of the research I did, the approaches I would consider, the technical limitations I found, wireframes of potential solutions, A/B tests I would run, competitive overviews, and sometimes I even have gone so far as to find friends of mine that use their product. Then I'd call up those friends and say, "Hey, can I talk to you for 30 minutes?" I interview them about their feedback and their experiences with that product. I record it, or at least I transcribe it. And I include those interviews and the conclusions I drew from those interviews in that project. So once I'm done with a project, I will usually email the completed document in like slide deck or PDF, to the recruiter and the hiring manager, thanking them for the interview, telling them that I put together some suggestions to demonstrate my interest in working with them and what I know about the industry. So the silver lining to all of this, even if it doesn't help you get the job, is that if you do it, you will get an even better chance of getting the next job. This is because first of all, you've done a lot of product thinking while you've been putting this thing together, and trust me, it builds up and it helps. But mostly because you can just take that case study and you can just put it on your website or your blog. And then the next recruiter that talks to you about this position is going to check it out. And they're going to say, "Wow, this person really knows their stuff. Look

at all of these case studies. If they've done in their free time, it's going to be awesome." On top of that, you can even talk about these suggestions and case studies for that company that you didn't get the job for, you can talk about them in the next interviews with other companies. It has so many benefits. All right, so use this information wisely. This is really a great technique, not only for getting a job, but thinking through that product management mindset. Like I said, you're going to have to go through that process and doing it is just going to improve your product mind as you go through it every single time. Look, if you apply to a company and you sit back waiting for them to call you with a job offer or waiting for them to call you for the next interview, you're doing the exact same thing as everybody else. Stand out from the crowd by putting in the effort, showing them that you really want this position and demonstrate to them that you've got the skills to do it. I hope a bunch of you are motivated now with that lecture, but also the stuff before it. If you do all of that, you're going to be in such a good place that you're going to be so far beyond where I was. I mean, when I first started, I had nearly no experience. I did not take this course, obviously, or that would be weird. At that point, there were no schools teaching product management. I had no product management experience. I kind of had to just figure it out, but now you've got someone like me that's gone through all of this telling you exactly how to do it. You are leagues above every single other person out there trying to get this position. All right, I hope you enjoyed that lecture. I know it's a lot of work, but it's totally worth it. I will see you in the next one.

The first things to do

- What's up everybody. So this is all about how to excel at the role once you've already got the product management job. And because this is all about how to excel after you've already got the job, I'm assuming you've got the job. And in that case, congratulations! I'm so happy for you, it brings a tear to my eye. And if only there were a way that I could through the screen, congratulate you or make you happy. How about this bear suit? I will teach the rest of this lecture in this bear suit with these claws because you

make me so happy to be a product management instructor. You got the job. So once you've got your first product management job, it can be a little bit daunting trying to figure out what the first thing is that you should do. But bear with me here, product management is notorious for having one of the longest ramp up times for a lot of jobs out there. By that I mean, it's really usually taking a lot of time to get completely comfortable with the role. This is because the work you do involves so many different teams and you need to understand the history and goals of each one to do the best work. What I'm going to do here is give you some tips on things to do as soon as you get the job and start work. These tips will make you beary-much more effective at your role. All right, so tip number one, schedule one-on-ones with everyone in the team. If you're at a large company, you don't have to meet with literally everybody, but be sure that you at least meet with the leaders of each major team. Ask them what their goals are, and what challenges they have. Take notes so that you can refer back later. All right, step two is arrange a meeting with the lead engineer on your team and talk about the technical stuff. Have them explain to you how the architecture looks on the backend in detail and what sorts of technologies they use, like databases and languages and that sort of thing. Also ask them what the biggest technical challenges are and make sure you understand the basics of how data is stored, and again, what languages they use, and where the sticking points are, what they're having trouble with. This will just help you build a great relationship with the engineers, and it will make sure that you know what you're getting yourself into if you're going to try to change something major on the backend with a feature. All right, so the next tip is, start talking to users. When you join a lot of companies, it's going to appear as if they know exactly what users want and even show you user feedback that they've received in the past or over the last couple of years. However, it's extremely important, or should I say beary important, to talk to your users and get the best idea of what's important to them. You want to see this yourself, because you may draw some different conclusions, after all, you're a brilliant product thinker. Some things you can do here are ask to join in on a sales call, go to sales meetings, spend some time with the customer service team, helping them answer user questions and tickets that people send in for support. Things of this nature is what you're going for. I also like to do a quick

Google search for things that people say about the company on the social websites, like the Twitter and the Reddit. Number four, my bear head is falling off, but read every internal document you can get your hands on. I recommend you ask or find documentation for the epics and mobile releases that have been made in the past, as well as things like sales decks and internal presentations. All of these things are going to help you get a great historical view of the product and what has happened over time. Tip number five, look at the data. As soon as you have the chance, you should take a look at the existing metrics for the product. If there's a data science team or any data analysts, ask to have a meeting with them where they describe the types of data that are available in terms of metrics, and any plans they have around implementing more tracking. Sometimes you might not think there's enough tracking to make the best decisions, and you'll have to make an effort with your team to track more. Tip number six, meet with the boss. You should make it a top priority to have a meeting with either the head of product or the CEO to ask them about expectations for your role, but also to understand their goals. Make sure what you talk about is related to what metrics they're trying to optimize for and what they'd really like for you to do in this role. What are their biggest problems? Like, what are they fearful of over the next year, in terms of stuff that's going to be challenging, or maybe what competitors are doing. You can also ask them about things like process improvements internally that need to be made, or improving marketing. These are all things that you can help out with as a product manager. All right, with those tips, you're going to be in a very beary position to do the best work as soon as possible in your new role. Like my claws are sharp, and I do mean my claws because I am a bear, your product management skillsets will be sharp, your knowledge will be sharp, starting this job. It's super important that you do this, and I've done this in all the companies I've worked for. It's been really huge. All right, everybody hope you enjoyed that one.

Summary of the course

- Hey, everybody. This is the end. But only the end of the course, not the end of your journey into the world of product management. Evan and I have

included a ton of reference material and content and, as you know, we've got the interviews available to you, which are very good. But we'd really like to add a lot more content to this course. The last thing I want to say before we get out of here is that something really fundamental to the role of product management is that it is not so much a set of skills that makes you a product manager. You do have to have these skills, don't get me wrong, and we've already talked about them, but you really need to have a mindset. That mindset is all about proving things through data, making the best decisions based on the information available, getting more information if you need it, and just generally thinking about things in a deeper way than just through your gut feeling. Remember, as a product manager, you're almost always wrong and so is everyone else. That is until you can prove that your assumptions are not wrong through data and through talking to users. Lastly, I want to say that this is the coolest job ever and I'm so glad, along with Evan, that you guys have joined us through this journey and learned so much about what we really, really like doing. Please feel free to reach out, to contact us, email us, write us, whatever. Tweet at us. Twitter's kind of cool. Snapchat us. Whatevs. And yeah, just see how you're doing. If you get a job or if this was helpful for you at all, let us know. And we're there to answer questions for you in the future. All right, everybody, I'll see ya later. Thanks so much for taking this course.

Interview with Daniel Demetri, product lead at Earnest, ex-PM at Google

(gentle music) - (Daniel laughing) Yeah, I like cheap. so awesome. So I'm here with Daniel Demetri, the head of product at Earnest. And could you tell us a little bit about yourself and your background? - So I was a computer science major at Harvard, studied that undergrad and economics. Worked in private equity for a year and a half, and then applied to Google's associate product manager program got in and started in New York, moved out here and worked in ads a bit, sorry maps, was on the maps team for two years at Google. And then I left and joined Ernest and I've been there for a year and a half. I was their first product manager and now I lead a team of eight. - Tell us about the APM program, because this is something that not everyone knows

about the Associate Product Manager program at Google. - Well, the Associate Product Manager program at Google is a amazing two year program. They take 30 people a year globally, typically, well zero to three years out of college, but it's typically zero. They take computer scientists, they train people up into the nebulous craft of product management. And most, I think that they had about 50% retention within five years. So they do a pretty good job retaining talent for quite a while. And they expose you to different parts of the Google ecosystem. So you can see ads, you can see search, you can see Google+ back in the day. - Do you get a choice on like where you want to go? - Yeah. So I remember my first request was like, please don't put me in ads. - Why? - And they were like, well, we can't promise anything, we'll see what happens. And then I showed up on my first day and they have this session where like all the trainees or the recruits of the crop sort of get picked up by their bosses. And I get basically at my first meeting with my first boss and I'm like, so what do you do? I mean, what do we do? And he's like, well, I'm the director of the ad exchange. And I'm like, my face fell, and I was like, Oh, I'm going to be an ads, this is the worst. And they like totally voluntarily disregarded my request to not be on ads, my only requests. - You have only one job. - I actually loved being on ads. Second year, you have a little bit more influences in what you do. And then throughout the rest of your career, Google has quite a bit of mobility, so. - So before you got into product management, before you were in APM program, what was your notion, your preconceived notion on what product management was and why were you interested in doing it? - Great question. I had no idea what it was. So I had a friend who was a product manager at Google, and I asked him, what do you learn as a product manager? Like, what do you become? And he was basically like, well, you learn how to get things done. And I'm like, that's super vague, come on. But he didn't really do a better job in explaining it. I felt that the role was going to be the marry the business and the tech sides and being a joint kind of computer science and economics person that really appealed to me. I knew that I liked being engaged with technology, but I also felt that I didn't want to be coding. And so thinking about technology, but also not coding, but being engaged with the coding and the coders that felt really appealing to me. I wasn't really sure what it was, I just knew that Google is a great place to work and it seemed like a great program, and so I felt like I'd try

it out. - So now that you have done it for some time and you're head of product at Earnest, if you were to summarize it in some way, like a short way, which we've actually, we've had a whole lecture on this in the course as well, like summarizing product management. How would you summarize it? - I mean, I think that a better way to think anything's done might be to fill the gaps between all of the cross-functional teams. So everyone in the company is working to make the product a success in some fashion, but there tends to be a lot of translation that needs to occur between engineering and in our case capital markets, 'cause we're lending money out and marketing and PR and all these functions need to collaborate, creating a shared sense of timeline, creating shared sense of what is being delivered or when or why and is sort of core to the product role and that sort of manifests in product requirements and manifests in roadmaps. And so I view like as a simplified version of defining what product management is, I would say it's something about filling the gaps between the cross-functional roles. - Cool. We actually just got done filming a lecture about the different types of product management roles in different companies, actually referring to your blog post. It's a good blog post. - Thank you. Thank you. I'm excited about it. - Except you have a different, probably a more thorough labeling, but we talked basically about internal product management, where your stakeholders are internal and then business to business and enterprise product managers and then just consumer product managers. - Yes. They're very different. And I think in resume screening, I think that like hiring managers definitely are taking a view as to, oh, that they're that type of product manager. And so one of the great things about Google was that it's such a broad conglomerate at this point, you can seamlessly transition between the teams and allow your functional or your core expertise in computer science or just academic pedigree generally speaking, to allow you to transition at a junior level between teams with actually very different styles. So I worked on DoubleClick, very B2B sales driven environment, which is like features prioritized on what customers were asking for. But then I switched to maps, it was entirely different it's logs analysis. What are clients or, clients I'm stuck in the Ernest, what are users of Google really doing and analyzing that and then prioritizing and bug closings and feature requests. And so then you can go out from Google or from another program to sort of address kind of any

business need. And so that's a great reason to start off a product management career at a big company, because as you were pointing out earlier in the course, there's so many different types, they do very different things, they have different values. So getting to experience both is very valuable for one's career. - I've talked to a number of Facebook PMs that apparently Facebook has a rotation program on. You're an early product manager, I think you don't even have a choice, but to rotate between departments or products, I think. You have to go from like timeline to photos, to whatever and get a good handle on all of these things. And that keeps the ideas sort of fresh. You don't get a tunnel vision on the vision for that one product. So that's interesting. - Google's APM program is not that similar. - Oh really? - Yeah. You can convince the leadership to allow you to stay in a similar area, but they do want to see you switch your perspective. - So speaking of features and you said feature requests, we have a whole section in the course as well on yeah. So you have ideas and you have feature requests, then there's this need to get to the core of a user need rather than just taking feature requests at face value. So there's a, I mean, in the course, I've talked about a number of times where I've had a feature request and I do two or three levels of digging and you find out that there's something that, that was a symptom, and then there's more of like a disease somewhere else. - I can think of a million instances right? - Yeah, could you tell us about any of these instances because of this practical, real world, like storytelling is really useful? - Sure. Yeah. So, I mean, right now this week, so Earnest is a consumer lender. We do student loan, refinancing and personal loans. We have an internal team that does some of the loan processing and they have gotten to the habit of reviewing raw documents that have been provided by the client, think like a tax form. And so they've gotten into the habit of, okay, I'm going to look at these tax forms and then I'm going to make a bunch of calculated, subjective judgments based off of that. And then I'll, using those subjective judgments, put in a decision. And so they ask for features that are like, I want to be able to pull all the documents more quickly when in reality, if you're thinking kind of a little bit ahead, what you want to do is extract the important information off those documents and then have the calculations run that judge the application's quality. And so, you don't want to build features that are going to make it easier to suck all the like

make it easier to review PDFs faster. You want to build the extraction mechanism and then automate kind of a lot of the logic that they're doing. It's a classic problem when you're dealing with an operations team, it's like that balance between automating their job and making their jobs just more efficient. In other circumstances, you know, you think that you can get away with, and a great example is on Google maps, we tried to get away with removing a feature called search nearby. So search nearby as a feature where you would search for a Gary Danko restaurant, maybe you're going on a date there. And then you want to find a bar next door for afterward meet up with friends. So you would click search for Gary Danko and then click search nearby and then search for bars. And we thought, can't people just type in bars near Gary Danko or just search for Gary Danko and the map will move and then they can search for bars. But then in the opposite to people requesting features is the opposite would be removing features people aren't worried about. Fixing what ain't broken, kind of can lead to, we remove this feature searched nearby and got, this became our biggest chunk of customer feedback at Google and the maps team. - By removing that? - By removing that. The consumer operations team is the team of Google that aggregates customer feedback, or user feedback. - There's a whole team. - There's a whole team that clusters the user feedback. I think very few people outside of Google are even aware that this happens because Google has a habit of not really responding to user feedback. In some circumstances they do, but actually the user feedback that's clustered by a team in India and then the product managers review the clusters. It's the only way to digest this information on scale. - Sure, because probably there's so much information. - Exactly, so the top cluster quickly became, where does search nearby go? And we're like, Oh shoot. We have to redo this. - That's awesome. I met a guy that was a PM of bugs at Facebook. So there's a person that they have like one person or I guess many people in charge of bugs, but it's the same thing. They're just have such a volume at like one and a half billion that they, you just can't look at bug reports. You have to do some analysis on the bug reports 'cause there's hundreds and hundreds and hundreds of thousands of them coming in. There's no way you could ever read that. - Which is another classic trade off product managers have to make, is between closing bugs and working on new features, especially when those

features are going to render obsolete. Some of the bugs that have been filed. - No features for that matter. - That's right. Or we could add that check box, but eventually the whole page is going to go away. - And they think you're ignoring them? - Yes, it's the worst when people, I mean, most product managers are probably hated in some respect in their companies 'cause they're a, no people are like, not yet people, or not now or that's not a priority. - That's the classic. We will prioritize it. - Yeah, well, that's P4 or P3. That's the next quarter thing. I think that that's what, if you generalize the view of those sorts of decisions, what you realize is that product management is very path dependent exercise. 'Cause you might agree on the vision with a bunch of people, which is probably the first goal of product management is just to agree, where are we headed with this product? Like with Spotify, it's like, what do we want people to do with this product? Is the vision more like adding videos in, this is a multimedia center? Or is this more just like a music? This is your like lay time. This is the background of your whole life, which is not video. Background of your life is probably more like just silent and responsive and aware of you. So then, okay, we agree on the vision of where this is headed, where of course the vision at Google is just be all things to everybody. And then which made it very challenging in fact to direct priorities. But then, okay, what's the roadmap to getting there? What are the coins along the playing field that we're going to navigate off course to pick up? And so it's like a video game. Like, should we pick up the two cent piece over there, even though it's kind of further because if we do actually it's a traveling salesman problem? Which is why we're not really worried about product management being obsoleted by technology, 'cause they haven't cracked that line - Right. Exactly. (laughing) Computers can't crack the traveling salesman things. - Exactly. So we're safe. - Yeah, exactly. Actually the first thing you mentioned, of the feature ideas where you have to dig deeper with the documents being pulled at a faster rate, that is like the classic example of the, I think the Steve jobs quote, or if you ask people what they want, they'll just say like a faster horse. - I've heard it's tribute to Henry Ford even. - Oh okay. - Oh, but Steve is the one who made it popular. - Unlike everything, didn't do it first. - Certainly not, I think very few people were asking for the iPhone. And I remember I was pretty excited to have a separate iPod from my phone and that they just seemed like different

functions of my life. And then only later I found V3, gen three was really where kind of came together as this is going to offer some pretty compelling new experiences. Some simplicity for one's life. - There's a, as you know Apple is a, I guess they're notorious for not doing like, I think they do user research, but not doing the broader high volume like- - The vision doesn't come from the bottoms up, there's top down vision and then they'll validate the execution, but they're not deriving strategic insight as much. - From people, from the average consumer, because back in, I think you can still look at some pictures like the early or late 1800s and their version of the future was like horses and buggies on like the moon, somehow we're going to get the horses and buggies up there, but it's still got to be a horse and a buggy. - It's not this similar companies like Google, Google takes a very supply side driven approach, which I think it makes them differentiated in the market. They're really focused on, what can we do with all of this technology? So given that we have Bigtable, like how can we apply Bigtable? Given we have search, how can we apply search? - Bigtable is their database of everything. That's what the single table (indistinct)- - No, that would be, they have the knowledge graph, that's the database of everything. Bigtable is the database storage technology. okay, so if you do have the Knowledge Graph and you do know a bunch of things about things, and then so like integrated that into Google maps and other product managers were integrating that to search as like knowledge panels. Other people are integrating that into Android. So you're just geographically located somewhere, it's pulling up relevant facts about your life in Google Now. And so these sort of core technologies is in data assets, the product management team then translates those into cool features and openly products. There was a whole product Google had called, I don't remember, and as you navigated through a foreign city, it would sort of craft a story for you to follow along. It was like guided tours. I think it's called like Sidekick or something like that. - One of these Google side things that just comes out. - There's a million of them. - Sometimes they die quietly. Sometimes they don't get publicized in the first place. - So companies like Google and what we're trying to create an Earnest it's awareness of what technology you have that then translates into products and features, which is reasonably different from how product management is approached at other companies that might focus more on

user research or focus more on customer demands. And it's just thinking, what can we do with this? - Okay. So if I am not mistaken, you actually have, you have job listings out for product management for Earnest. I mean, it probably will not be the case when this course is out, but- - No, we're going to be canceling hiring part. - Okay. - Don't worry. We had it open for a year and a half or year. We're constantly hiring. It's how the team goes from just me to eight people, by that thing being open. - Okay. But you're still the person I'd like to ask about, because this course is about getting into product management for a number of people, how would you, first of all, suggest someone goes about learning about the role? Obviously this course, but ultimately looking to get a job there, techniques and such for finding them and that sort of thing, but then also what do you like to see as a hiring person? - Hiring manager. - Yeah, as a hiring manager. What qualities are you for? - For sure. So product management learning about it is tremendously difficult. It's very hard to craft a product management internship. We did do it at Earnest and we are hiring interns, including interns with no product management experience. - SO that's good then. - Relevant audience hopefully. - Product management- - Did she want this? Tiffany wants. - Product management internships are probably a good exposure. I didn't do one. I didn't know about product management internships when I was in college and afterward I didn't need it. So, but I've heard great things about it from my friends who have done it. Otherwise learning about product management are to work in a product environment. So I've definitely had a number of engineers sort of approach me and they're interested in product management. They don't yet have enough experience working with products engineering. So they might be very infrastructural- - Backend production engineering, that sort of thing. - And so I encourage them to transition closer to products prior to jumping onto the product management team so that they can observe and participate in the practice decision making process before they're in a role that it's exclusively focused on that. So front-end engineering or just product engineering, generally speaking is a great way to get closer to product management. Once one has a sense for what product management is, the hiring managers are looking for folks, at Earnest, we're looking for four things. So we're looking for analytical skills, technical skills, creative skills, and collaborative skills. So that's like at Earnest, I think other companies do based on the type of product

manager they're looking for, they might switch it up, but I can speak to Ernest really well. And so analytical skills, we're looking for things like proficiency and SQL. Do you know your different join types? Do you know how limit works? - Sort of a basic proficiency, but not too basic? - Up to the point of an advanced proficiency would be even better. - SQL is a database technology. And then you wouldn't learn my SQL, which are, you learn SQL queries, which is being able to query the database for information. It's pretty universal. - And disparate data sets. Excel junkies can kind of, are most apt to be able to transfer into SQL experts. - That's not too different. - Exactly. You're just dealing with understanding data at the highest level. Maybe you're or a stata junkie, wherever data is, if you're capable of thinking through basic statistics or thinking about how you can tie data together to derive insights, you're on the path to making data driven product decisions, which are the types of decisions that you'll be able to use to get the respect of your team. - That's a big one. The respect of your team sort of thing. At least even if you're not technical, you should... there's a product manager named Ellen Shiza we actually went to school in Boston here in '09 and she has a really great article about this. If you're not technical, like how technical should you become? And she argues that it should, at minimum, you should be curious all the time about learning more. It doesn't mean you have to go be an expert coder this weekend, but it means that you should be curious about every single new technology that you're using and always asking engineers about things and demonstrating interest in learning these things, rather than just saying, I don't care about these things, we'll have an engineer deal with it or whatever, or I'll ask an engineer every single time. - Which has got to be an expensive conversation to keep up. - They're very expensive people. - So that describes kind of the second bucket of skills that we look for, is technical skills. So we're looking for folks who have done engineering or have done a computer science class or have learned programming through general assembly thing and then applied that in some respect on side projects. So ultimately your ability to understand how the technical architecture has implications for product extensibility is what is core to the technical knowledge, as well as not getting shown up on the technical side by folks like your designers. So that just kind of impairs trust in a manner of speaking. So I challenge all of my product managers to not get shown up by our product

design team, which is, relatively advanced design team. They're getting into gear, they're making some copy changes, style changes, they're publishing those. They're dealing with unit tests failing and understanding the build failed. What does that even mean, the build failed? If you don't know what it means for the build to fail, you still need to ramp up your technical skills before you can really get into a technical product management role. Maybe you'd be product manager like a toothpaste company, but of a tech company, you need to understand the process of building the system. - And even if you're not pushing code yourself, to production, you need understand how flow works. You can do this integration and like what gear is and what branches are, and what- - You need the vocabulary. - You need the vocabulary. - On the creative side, I usually challenge people to imagine what I mean by that. And then on the collaboration skills- - That's a good one. - (laughs) On the collaboration skills, that's like, I think of communication skills. I think of getting things done without stepping on too many feet. Communication skills, I've also further broken down for folks. It's the ability to speak with accuracy, so what you were saying is true, which is actually sometimes challenging, especially when you don't know something, it can be easy to speak about truth with things that are actually inferences or you're uncertain about them. So speaking with truth is accuracy, speaking with precision or communicating with precision. So that would mean saying things in an unambiguous manner. They can't be misinterpreted. And then speaking with brevity. So even if you were to speak with accuracy and precision, you might still have that coded in tons of other information and really great product managers are able to communicate with all the certain teams in those three respects briefly, accurately, precisely. And then additionally, they're able to translate. They use words that sort of communicate with each of the teams. Legal is going to have a different perception on what a policy is than engineering, because engineering has a rounding policy and a data persistence policy, legal doesn't really care about the data persistence policy. They care more about the products implications for users and how their users being charged to things. - The communication thing is huge as you need, there's a saying, which I don't know if it applies anymore, but for every single group of stakeholders, so you executives, your legal and marketing and engineers and design, there's a different deck. So

there's different words for different groups. Like you said, they should all still be concise and brief and true to the point, factual, but then if you're talking to an executive, it's going to be ultra brief. - Ultra brief. At Google, you were known to have to, especially in the earlier days to chase around executives. And so that you could catch them in an elevator moment and then give them your 30 second pitch and get the verbal approval and you've run off and keep executing. That was how brief you had to be to win as a product manager back in like 2007. - That's awesome. - And people would stalk there'd be like a chat group where you're like, Oh, Marissa is in the elevator, grab her. Fortunately we're not like that yet at Ernest. - So you mentioned the four things and let's say that someone has, they've learned about what product management is and they have learned some technical skills and so on and so forth. And they've applied some technical skills maybe to the side project and that sort of thing. And they're looking not even for a job with you, but just anywhere. Is it vital to have some sort of portfolio thing project to point at or? - That's a great question. I think that what I've been looking for has been a little bit less portfolio based and a little bit more experience based. We want to see that people have gone through the trials of interacting with a really top class product designer and knows kind of how to communicate as we were just talking about. One of the core competencies is being able to translate with the different functions. That's sort of the ultimate purpose of the role. And so if you have created an independent portfolio that you've product managed this product on your own, it doesn't demonstrate that you have the skills to communicate well with the product design team and understand what their concerns are going to be or their considerations or what you can delegate to them. Similarly with a great product engineer, similarly with a great, maybe even a legal team or marketing team. And so we're looking a little bit less for independent work and a little bit more for experience working with these teams. - So product internship role and (mumbles) - Or even like in coursework, like cross-functionally collaborative products sort of- - It could be a side project as long as you're actually doing this communication between the different groups and (mumbles). - Exactly. It just can't be a tiny group of you and one other person, you're wearing too many hats in that role. And you're not practicing the scaling of those hats being worn by different people than you coordinating

all of them. So, okay, for entry level product managers, we're really just looking for basic proficiency in those skill areas, more so than experience. So we're looking for, do you know computer science fundamentals already and do you know SQL and do have a track record of the creative side of being able to problem solve interview questions with us in the whiteboard, or an academic thought, do you have a cool thesis. On the collaboration side, did you participate in any group projects extracurricularly? So, at Harvard there's the like robotics competition, you can jump into that and help build a better robot. Or do you just show leadership by becoming president of the school newspaper? These are sort of like the core collaboration skills or indications of those core collaboration skills that will give us confidence to bring you in as a product manager at the entry level, but beyond the entry level, going into mid level and senior level, we really need to see actual product management experience with the types of stakeholders that you're going to be engaging with at our environment. - Cool. All right. Well, so I think that about does it. - Thanks for having me. - Daniel Demetri, head of product at Earnest and we'll see you on the next lecture. So, there's one lecture on metrics, and one lecture on program management overview, one lecture on like- - [Man] Direction on city course, insider course and MVPs - Oh yeah, yeah, yeah, yeah. - And understanding like customer development and product market share- - So which segment is this...

Interview with David Lifson, VP of product at Homepolish

- Hey everyone, I'm with Dave Lifson out of New York, VP of Product for Homepolish. And we're going to do a little bit of a interview here on some product management topics. And Dave has been, Dave you were at what? You were at Etsy before, long ago and what Amazon as well. - Yeah, so my first job out of school was at Amazon. I was a software developer and then realized within probably 15 months that I was made to be a product manager. And so, I just switched to the recommendations team at Amazon, which is because you bought these items, we recommend these others. And, I was at Amazon for a total of two years, learned a ton, loved it there. But I just had an opportunity I couldn't say no to, to move to Brooklyn from

Seattle and be the first head of product at Etsy. So, that was 2008 when Etsy was 50 people and all the three founders were there. And, I then did my own startup for three years with the technical founders of Etsy. We sold that company, I joined General Assembly as employee 25, and grew that from 400 when I left, over two and a half years. I started running curriculum and then moved to running product design and engineering. I spent a year at Poppin, doing the same thing, product design and engineering. And then now I just started at Homepolish about a month ago with the same. - Cool. So yeah, this course has about of, right now has over 4,000 students that are enrolled and it's only been out for about a month. And it's focused on the people that are new to product management or people that are in the role and they're trying to get better. And yeah, one of the recurring themes as I know that you know, from teaching at General Assembly and that sort of thing is that product management, like learning about it is quite popular right now because it's becoming a more popular role. People are starting to learn kind of, what product managers are doing, but they want to find out more, but there's no good way to learn about it, like in a formal way, or you can't get a degree, you kind of have that. Like you and I both just happened upon it. I think most people end up doing that. And I think right now, due to things like General Assembly or like this course, you can learn about it, like formally from an actual product manager. So like the first question I wanted to ask was, it's an interesting one, no matter who I ask is like if you had to sum up the role of product management in a sentence or two, what would it be? What does a product management mean to you? - Oh, wow. Yeah, that's really great. So, what does product management mean to me? I think the role of the product manager is to have a deep understanding of the customer, a deep understanding of the business and marrying those two together to identify opportunities and problems prior advertise those opportunities and then devise the right solution for the right time. And so, what's so challenging I think about product management is that, we don't fit the mold of the individual contributor. I think organizations historically have been structured similar to, how they were in the industrial revolution. With, in manufacturing where you just, you have departments that, have workers and workers create output and coders write code and marketers create marketing and designers create designs and Product managers don't create anything, (mumbles) for

everyone else that we work within the organization. So that the quality that is produced is higher and the solution is sort of more efficient, and more effective. - Cool. Okay, and so what did you think about the role before you were actually a product manager? Did you have any frequency of notions? or did you know anything about it? - Oh, well, my first team at Amazon, I was in the community group, which did user-generated content? And so my little corner of it was this thing called LiS mania where, Amazon customers could create lists of their favorite products. Like your top 10 things to take with you on a camping trip or my favorite scary movies. And, we had no product manager, it was just a bunch of engineers and we just built whatever the heck we wanted to build. And so I ended up, Jeff Bezos had this idea. This was back in 2006, where Facebook was becoming a thing. People were talking a lot about networks. And so he's like, "Well what, if we created a network "of product SKU?" If, products SKU have relationships to each other, what would that look like? And somehow that ended up on my desk, as a 23-year-old. And so we decided to build this hot or not for products. So essentially customers could vote on which of these two shoes is more stylish? Or which of these two electronic to use. And as you vote. (mumbles) We would use your votes to generate you a recommendation. Very expensive thing, we built the whole thing. We launched it and it was a total disaster conversion rate was somewhere around like half a percent. And it was like such a clear example of how product management could have saved the day, and we just didn't know it. - Mm-hmm. Yeah, if we're working on the right thing rather than just something. - Yeah, and we were testing any assumptions or doing any sort of user research or prototyping. And so, when I moved into the recommendations team, which did have, a track record of having strong product managers, I was surrounded by people who could really show me the routes, which is sort of how I learned. - Cool. So, something that we talk, because you mentioned this, like talking to users or, validating something and validating your assumptions. Another thing that happens quite often, to us as product managers, is we talk to users or we get feedback and they send some requests or maybe if you're an internal product manager, you get basically like what amounts to a feature-request or product, spec request. And oftentimes, that thing that's being requested, is not what, it's actually the solution right there. "Cause it happens to everyone, if you work

two levels deeper, there's actually another problem here that maybe that solution doesn't solve. And so, when you're receiving feedback from users, you got to dig in deeper. Do you have any war stories of user requests that are just absolutely, were just not the solution at first that you can share and maybe what the real solution was? - Well, I don't know if I can tell the real solution, 'cause I don't know, but I'll tell you a story of this exact thing. Except it wasn't a thing. (mumbles) It was a founder. So this was the end of 2012, and the founders were really worried that the business model was looking a lot like University of Phoenix, in the sense that our revenue came from adults coming to a classroom and learning. And so, revenue was really constrained by, one, our ability to purchase or rent real estate. And then secondly, our sales ability to fill that real estate. And so, they came to me and they said, "We really need something that is a subscription "and can target customers who don't live, "where we have classrooms. "And you have three months, go build it." And so this was like, this was a big mistake for me because as a product manager, I should have recognized immediately that there was no mention of the customer here or any sort of customer problem. This was something that came out of a spreadsheet. And an analysis of sort of like market valuations and it would be really nice if you know, this sell in our Excel spreadsheet was twice as big. And in some cases, it is up to the product manager to translate that into, "Okay, well then I'm going to go do some user research "and discover what a customer problem is, "and then work towards product-market fit." But given the three-month timeline on top of that, we felt really pressured to just look around to see what assets we had available and what we could packaged together. And so we did happen to have, a library of live streams that we had recorded and stored away. And so we said, "Okay, how about for 25 bucks a month? "You can have access to our library of videos." And we called that front row. And so we built it, we launched it. It did okay. But no matter how much we optimized the signup flow, the UX, optimized the acquisition marketing channels, the LTV just really wasn't there. I still today feel like at its core, subscription is just a very odd business model for education. So that yeah, that was always a trick one. - Yeah, yeah, I've got tons of stories of myself, but that's the one thing all PMs have in common. So, there's always these weird, yeah, user requests. And yeah, I think in your

early years, you'd probably just couldn't do those things, and then you realize that's big part of your job is taking all of those and figuring out what the real underlying need here is. And in another interview we did, we were talking about that quote that, I think the Henry Ford quote which is like, "People would have asked for faster horses," instead of the car back in the day, 'cause they just can't see past. Not everyone's sitting back and thinking big all the time, and that Apple sort of takes advantage of this with their products, by not asking the users what they want, but rather seeing what people want through, yeah, usability testing and all of that. And then kind of giving them stuff that they didn't know they wanted or needed. - Yeah, this is really the hardest thing I think for a project manager, because when you're starting out in your career, I think it's, you want to get some traction, get some projects under your belt. Just like practice the process and so it's much easier to take things sort of, after product-market fit, you're just in optimization mode, right? How can we take our signup flow, and increase conversion by 10%? And in that case, you do your user testing. You like, source out what those steps in your funnel that are giving you the most trouble, and then you just tweak them. And so, it's very like here's my problem, here's my solution and solve it. But when it comes to like inventing the iPhone it's such a big open-ended problem that it requires, I think a very different approach where you really want to take more of that IDEO style, user-centered design approach of let's just like it a very open mind, just talk to people and learn their context and there situation, and then take these leaps of faith and do a whole bunch of prototyping around it. Yeah, it's very different. (chuckles) - Okay, so just a couple more questions, one, because not only are you a teacher, for that you've taught product management in the past, but you also are a hiring manager and now VP, and before that, head of product, if you were to, work from the inside out and say, if you're doing it all over again and you didn't fall into the role like you or I did, but you were actively trying to get into the role. Let's say in modern, like current day, how would you recommend that someone with very little or no experience tries to work their way towards getting a product role and what skills maybe should they learn or how should they look at approaching it? - Mm-hmm. Yeah, so, (clears throat) Yeah, so I think the way to do this is to really show that you understand the product management process and that you're able to apply that process to a lot of

different situations. So the process in my mind is, (clears throat) identifying what you think the problem is, what you think the customer is, what you think the solution is, going through this iterative approach to getting the product-market fit. Being able to draw upon a lot of different tools in order to validate those assumptions, right? All those different MVPs that you want to use. And I think that there's a lot of different ways in which you can demonstrate your understanding of that. You can go to a hackathon and work with people and in the span of two days, apply the process or perhaps you work in marketing or you work as an analyst, like you can still get involved in the development process, at least perhaps in the user research part or the opportunity analysis part. And so, in that way you can then actually structure your resume, so that it highlights all of those different steps of the process. So as a hiring manager, I can look at it and be like, "Okay, they don't have the product manager title, "but they completely get sort of what the five steps are. "And they've shown that they've done those steps "in the past." - Yeah, okay, that's really good. So, this is some of the advice that I give in that in terms of prepping for the roles, first of all, look internally is one of the easiest ways obviously. But if you're not able to do that and you're trying to apply somewhere, then you should look at things we've done in the past that have to do with product management. And since it's such a vague and wide-ranging role, most everyone has done something that a product manager will do at some point, like cross-team collaboration or project management or, coming up with a few different ideas and finding the best one with data. Even people in the community operations, customer service, marketing, people do some things on these roles that are product management style things, and then shape your resume to explain those things. That's what I'm hearing from you? - Yeah, I mean, let's say you're an account manager, you probably talk to customers all day, so put together what you think the product roadmap should look like and then support your proposal. Like, what is the data behind your prioritization? What is the user research behind your prioritization? What are the risks and assumptions that you would want to validate for each of the ideas in your roadmap so that you would feel confident that executing those roadmap ideas would in fact achieve the desired goals. - Yeah, absolutely. Okay, so just for the last question, if someone is, let's say very familiar with the stuff and they're looking at just, they're looking at learning

and real hard skills. Like either technology or UX or something like that, let's say they're all going to take a PM course somewhere, and then they were also going to go learn a skill. what skills would you recommend that people be like either really familiar with, like a technology right or something that really stands out to you as being a big benefit on someone's resume? - Well, certainly they're going to need to understand just basic agile project management principles. And so being able to break things down into user stories and sprint planning and sprint retros and, just the mechanics of that, I think wire framing is something that they should be able to do. They don't need to learn a tool like Sketch or OmniGraffle. I think it's fine, If you just grab a marker and paper and can draw, but you need to be able to communicate what you see in your head to other people through pictures. And then on the technology side, I think people have the most anxiety about the technology piece. And what I would tell you is, spend some time understanding how the internet works. So if you type a URL into your browser and click go, what happens? And you should be able to go really, really deep on what happens. 10, 15 different steps between when you hit go and when the page shows up on your site or on your browser. And I would not, feel like you need to learn a programming language. I don't think you need to learn HTML and CSS. And I don't think you need to learn Ruby on Rails. As much as it is more important, just learn internet architecture and how do web servers work. - Yeah, and things like, in the course, I've talked a lot about you just need to understand what okay front end is versus back end and where things are being calculated, like where the logic is living on the server on the Front end or how did things get to your browser or your device? - Yeah, and I think if you want extra credit, you could probably spend some time learning SQL because a lot of times product managers need data. And so sometimes they have to go get it themselves because there's no data department to do it for. - Right, awesome. Okay, so I think we're just about out of time, 'cause we're in different time zones. You got to get to work, but thanks. Thanks for turning for this. And, do you have any, I don't know, websites or Twitter handles or anything blogs that you want to pitch as a last thing for the students that want to follow you on there? - Follow me? Oh, no. (chuckles) I'm not much of a social media person. I mean I, no, sorry I don't. I mean, like my, (mumbles) you can find me on

tumbler@caterpillarcowboy.com, but, it's just my personal blog. There's nothing on there that you're going to learn much from. - Yeah, that's good though, that's what I like hearing to be honest. 'Cause it means you're not on medium posting those, life health, life advice, articles - Yeah. - that a lot of people--- - And like, I don't know Twitter just stopped being exciting for me like five years ago. But here I am, I'm happy to teach, and give back. - Cool, alright. Well thanks, thank you much thanks Dave, we really appreciate it. - Yeah, you bet. - Alrighty, catch you later.