# Developing and Validating Solutions and MVPs

## Introduction

Once you get an initial business concept through all the previous screening, and you are certain (to whatever extent it is possible) that it is worthy of further investing your time and efforts, you will reach a point in business design where you'll need to create a **Minimum Viable Product**.

The so-called Minimum Viable Product (MVP) is a strategy used for fast and quantitative market testing of a value proposition, product, service, or feature. It was initially used for web applications and popularized by Eric Ries. An MVP is not always a smaller or cheaper version of your final product. Often a cheaper proxy allows you to test the assumptions underlying your value proposition more quickly and more cost effectively. **Ask yourself what you really want to learn and what the cheapest hack possible is to test that assumption.** Below, you will find useful insights on MVP development and testing, courtesy of Dan Olsen (*"The Lean Product Playbook: How to Innovate with Minimum Viable Products and Rapid Customer Feedback"*). Use the sections to aid your MVP development.

## Specify Your Minimum Viable Product (MVP) Feature Set

If, by now, you have **a clear understanding of your value proposition**, the next step in the Lean Product Process is to **decide on the feature set for your minimum viable product (MVP)** candidate. You are not going to start off by designing a new product that delivers on your full value proposition, since that would take too long and be too risky. For your MVP, you want to identify the minimum functionality required to validate that you are heading in the right direction. We can call this an **MVP *candidate*** instead of an MVP because it is based on your hypotheses. You haven't yet validated with customers that *they agree* that it is, in fact, a viable product.

For each benefit in your product value proposition, you want to brainstorm as a team to come up with as many feature ideas as you can for how your product could deliver that benefit. You have done all this great thinking in the problem space and are now transitioning to solution space. As your team brainstorms, try to build on each other's suggestions and push each other to come up with even more creative and outlandish ideas. When you are done brainstorming, you want to capture all the ideas that your team generated, then organize them by the benefit that they deliver. Then, for each benefit, you want to review and prioritize the list of feature ideas. You can score each idea on expected customer value to determine a first-pass priority. The goal is to

*At this point, brainstorming rules should apply. You should be practicing* divergent thinking, *which means trying to generate as many ideas as possible without any judgment*

identify the top three to five features for each benefit. There is not much value in looking beyond those top features right now because things will change a lot after you show your prototype to customers.

## User Stories: Features with Benefits

*User stories* (used in Agile development) are a great way to write your feature ideas to make sure that the corresponding customer benefit remains clear. A user story is a brief description of the benefit that the particular functionality should provide, including whom the benefit is for (the target customer), and why the customer wants the benefit. Well-written user stories usually follow the template:

> *As a [type of user],*
>
> *I want to [do something],*
>
> *so that I can [desired benefit].*

Here's an example of a user story that follows this template:

> As a professional photographer,
>
> I want to easily upload pictures from my camera to my website,
>
> so that I can quickly show my clients their pictures.

This template is a good start but writing good user stories is an acquired skill. Agile thought leader Bill Wake created a set of guidelines for writing good user stories; to make them easier to remember, he uses the acronym **INVEST**:

- **Independent:** A good story should be independent of other stories. Stories shouldn't overlap in concept and should be implementable in any order.
- **Negotiable:** A good story isn't an explicit contract for features. The details for how a story's benefit will be delivered should be open to discussion.
- **Valuable:** A good story needs to be valuable to the customer.
- **Estimable:** A good story is one whose scope can be reasonably estimated.
- **Small:** Good stories tend to be small in scope. Larger stories will have greater uncertainty, so you should break them down.

- **Testable:** A good story provides enough information to make it clear how to test that the story is "done" (called *acceptance criteria).*

## Breaking Features Down

Once you have written high-level user stories for your top features, the next step is to identify ways to break each of them down into smaller pieces of functionality – a process called "**chunking**". **The goal is to find ways to reduce scope and build only the most valuable pieces of each feature.** When someone comes up with a feature idea, there are often creative ways to trim off less important pieces. We shall deliberately use the term "feature chunk" instead of feature to remind you that you should not be working with items that are large in scope, but rather breaking such items down into smaller, atomic components.

Let's illustrate the idea of breaking a high-level user story down. Say you are working on a photo sharing application and start out with the user story: "As a user, I want to be able to easily share photos with my friends so that they can enjoy them". One way to break this story down is by the various channels a customer can use to share photos: Facebook, Twitter, Pinterest, email, text message, and so forth. Each of those would be a distinct feature chunk or smaller scope user story. You may not need to build out all of these sharing channels for your MVP. Even if you decided that you did, it helps to break the story down to be more specific in your product definition, to enable more accurate scoping from development, and to allow you to explicitly prioritize the order in which you build the chunks. You might also limit scope by enabling the user to share only the photo and nothing else for your MVP. You may have ideas for additional functionality down the road such as adding an optional message to each photo or the ability to tag users in photos. Each of those would be a distinct feature chunk.

## Smaller Batch Sizes Are Better

The tactic of breaking features down is consistent with the Lean manufacturing best practice of working in small batch sizes. When a product is being manufactured in a factory line, the batch size is the number of products being worked on together at the same time (at each step of the manufacturing process). The parallel for software development is the size of the features or user stories to be coded. **Working in smaller batch sizes increases velocity because they enable faster feedback, which reduces risk and waste.** If a developer spends a month at her computer developing a feature and then shows it to the product manager and designer, there is a greater chance that there will be a disconnect and that their feedback will require significant changes.

If, instead, the developer shows her work to the product manager and designer every day or *two,* that prevents a large disconnect from occurring. The magnitude of feedback and course corrections will be much smaller and more manageable, resulting in less wasted work and higher productivity.

This advice also applies to product managers and designers showing their work product (e.g., user stories and wireframes) to their teammates, too. The benefit of working in small batch sizes applies to customer feedback as well. The longer you work on a product without getting customer feedback, the more you risk a major disconnect that subsequently requires significant rework.

## Deciding on your MVP Candidate

Once you are done chunking, scoping, and prioritizing, you can create a simple grind that lists the benefits from your value proposition, and that list for each benefit, the top feature ideas broken into chunks.

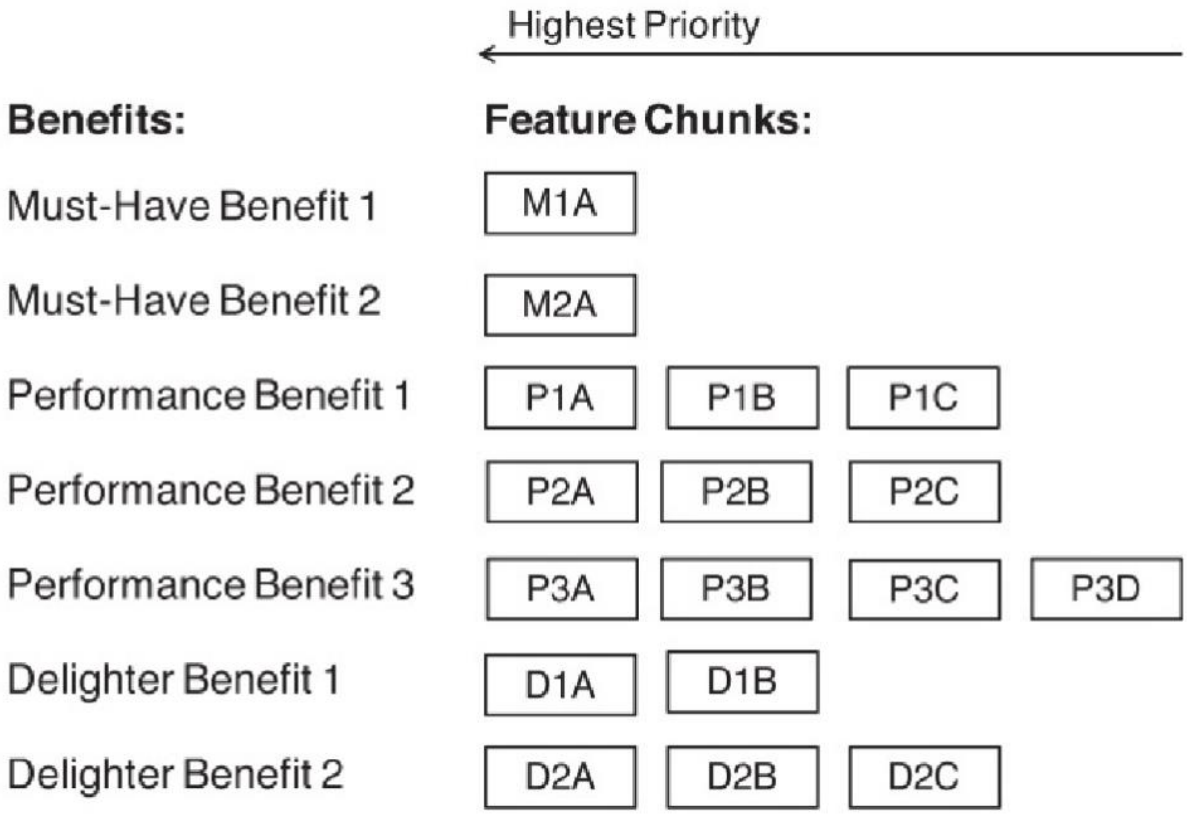


FIGURE 1. LIST OF PRIORITIZED FEATURE CHUNKS FOR EACH BENEFIT

In Figure 1 are listed the top feature chunks for each benefit in priority order, with higher priority on the left. Rather than naming specific benefits or feature chunks, we have intentionally given them generic names so that you can more easily envision replacing them with what would be relevant for your product. "M1A" means feature chunk A for must-have 1. "P2B" means feature chunk B for performance benefit 2, and "D2C" means feature chunk C for delighter benefit 2. In filling out a similar grid for your product, you would instead use the specific labels for your benefits and feature chunks.

Once you have organized your list of feature chunks by benefit and prioritized them, it's time to start making some tough decisions. **You must decide on the minimum set of functionality that will resonate with your target customers**. You are going to look down the leftmost column of feature chunks and determine which ones you think need to be in your MVP candidate. While doing so, you should refer to your product value proposition. To start with, your MVP candidate needs to have all the must-haves you've identified.

After that, you should focus on the main performance benefit you're planning to use to beat the competition. You should select the set of feature chunks for this benefit that you believe will provide enough for customers to see the difference in your product.

Delighters are part of your differentiation, too. You should include your top delighter in your MVP candidate. That may not be necessary if you have a very large advantage on a performance benefit. The goal is to make sure that your MVP candidate includes *something* that customers find superior to others products and, ideally, unique. The feature chunks that you believe need to be in your MVP candidate will stay in the leftmost column, which you can label **"v1**," as you see in Figure 2, while the others are pushed out to the right. You can create a preliminary product roadmap by continuing this process and creating columns for each future version with each column containing the feature chunks that you plan to add.

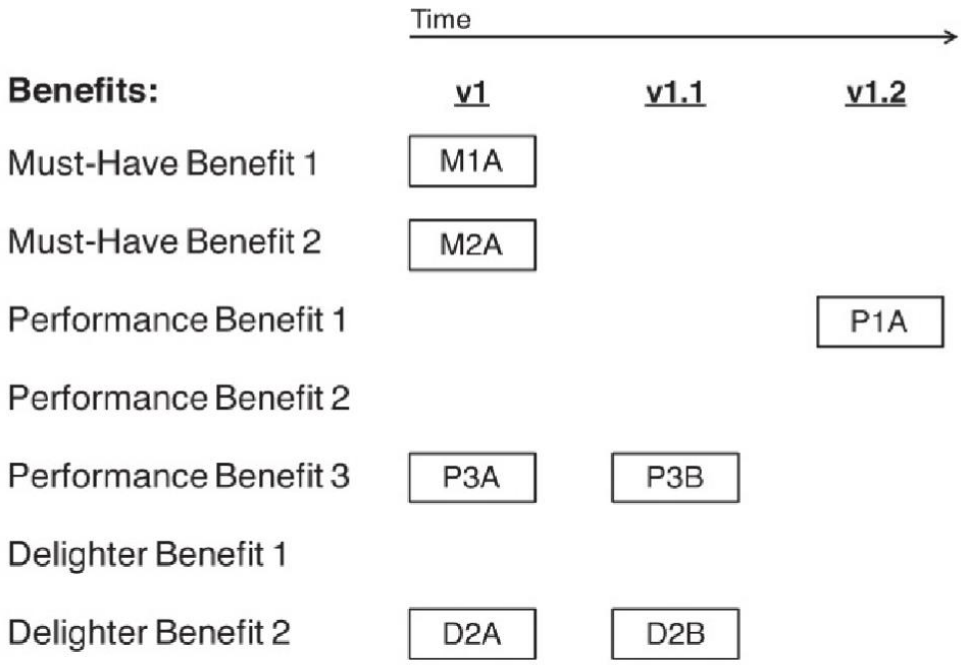


FIGURE 2. DECIDING WHICH FEATURE CHUNKS ARE IN YOUR MVP CANDIDATE

Since you plan to be best at performance benefit 3, you are including the highest priority feature chunk, P3A, in your MVP candidate. You also plan to differentiate with differentiator 2, so you are including feature chunk D2A in your MVP candidate. Your MVP candidate also has the two must-haves.

Looking out to your next version, v1.1, you plan to invest further in performance benefit 3 and delighter 2 with feature chunks P3B and D2B, respectively. In the version after that, v1.2, you plan to start addressing performance benefit 1 with the highest priority feature chunk P1A.

It is not recommended that you plan more than one or two minor versions ahead at the outset, since a lot of things are apt to change when you show your MVP candidate to customers for the first time. You'll learn that some of your hypotheses weren't quite right and will come up with new ones. You may end up changing your mind on which benefit is most important or come up with ideas for new features to address the same benefits. So, if you've made tentative plans beyond your MVP, you must be prepared to throw them out the window and come up with new plans based on what you learn from customers. However, it may be the case that you need two or three feature chunks for a given benefit, depending on your situation and how small your chunks are. The idea is still the same: to pick which feature chunks need to be in that leftmost column, which corresponds to your MVP candidate.

Let's take a step back and reflect. At this point in the Lean Product Process, you have done a fair bit of work You have:

- Formed hypotheses about your target customers
- Formed hypotheses about their underserved needs
- Articulated the value proposition you plan to pursue so that your product is better and different
- Identified the top feature ideas you believe will address those needs and broken them down into smaller chunks
- Prioritized those feature chunks based on ROI
- Selected a set of those feature chunks for your MVP candidate, which you hypothesize customers will find valuable

You have done a lot of rigorous thinking to get this point, but your MVP is still just a candidate, a bundle of interrelated hypotheses. You need to get customer feedback on your MVP candidate to test those hypotheses. But before you can test, you need to create a solution space representation of your MVP candidate that you can show to customers, which is the next step in the Lean Product Process.

## Creating Your MVP – What Is (and Isn't) an MVP?

There has been spirited debate over what qualifies as an MVP. Some people argue vehemently that a landing page is a valid MVP. Others say it isn't, insisting that an MVP must be a real, working product or at least an interactive prototype. The way to resolve this dichotomy is to realize that these are all methods to *test* the hypotheses behind your MVP. By using the term "MVP tests" instead of MVP, the debate goes

away. This allows more precise terminology by reserving the use of MVP for actual products.

Many people misinterpret the term MVP by placing too much emphasis on the word *minimum.* They use this as an excuse to build a partial MVP that has too little functionality to be considered viable by a customer. Others use "minimum" to rationalize a shoddy user experience or a buggy product.

*While it's true that an MVP is deliberately limited in scope relative to your entire value proposition, what you release to customers has to be above a certain bar in order to create value for them.*

The diagram in Figure 3 illustrates the difference between this incorrect way of interpreting MVP and the correct interpretation.

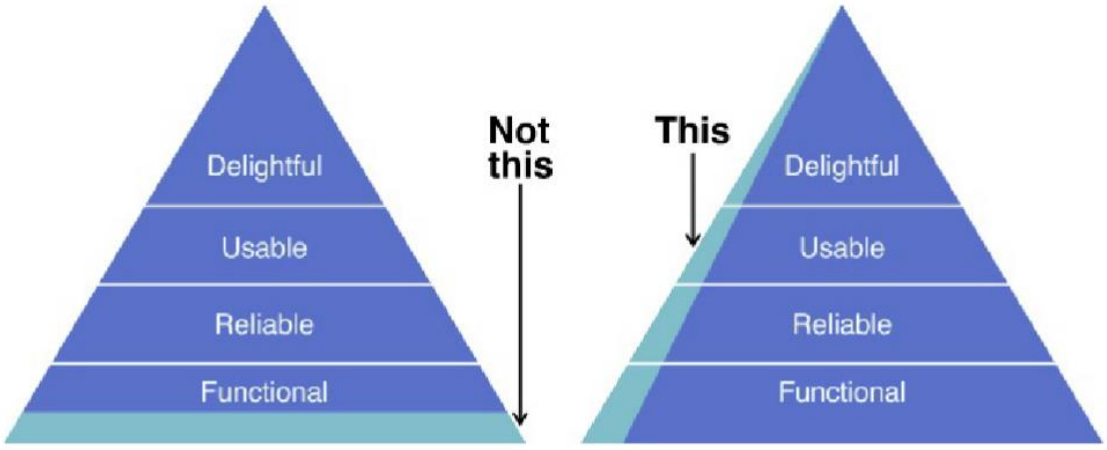


FIGURE 3. BUILDING AN MVP

Similar to the hierarchy of web user needs, this figure separates the distinct aspects of a product In this case, **a pyramid of four hierarchical layers is used to describe a product's attributes: functional, reliable, usable, and delightful**. The pyramid on the left illustrates the misconception that an MVP is just a product with limited functionality, and that reliability, usability, and delight can be ignored. Instead, the pyramid on the right shows that while an MVP has limited functionality, it should be "complete" by addressing those three higher-level attributes.

## Wrapping-Up - MVP Tests

Finally, after you did all the work, now it is time to test your MVP. For a strong conclusion of our session dedicated to developing and testing MVPs, please check the appendix test framework (courtesy of Strategyzer) and try to find the most appropriate testing techniques that you can apply during the search and design phase of your business idea.

Discuss them in your team and choose where / how to start accordingly.

Further reading & information:

- [(PDF) Minimum Viable Product or Multiple Facet Product? The Role of MVP in Software Startups (researchgate.net)]
- [Develop the Solution (iitoolkit.com)]
- [Validation of the solution and construction of the MVP - Scaleapse]
- [What is the MVP stage in a startup? | London Business Advice (hwca.com)]

## Sources & references:


AG, S. (no date) Testing your business model, Resource Library. Available at: https://www.strategyzer.com/assets/resources (Accessed: November 7, 2022).

Improving improvement (no date) Develop the Solution. Available at: https://www.iitoolkit.com/process/solution.html#:~:text=Developing%20the%20solution%20has%20direct,evaluate%20potential%20ideas%20for%20improvement (Accessed: November 7, 2022).

Pansini, S. (2022) Validation of the solution and construction of the MVP, Scaleapse. Available at: https://www.scaleapse.com/validation-of-the-solution-and-construction-of-the-mvp/ (Accessed: November 7, 2022).

Umbraco.Web.PublishedModels.TeamMember (2018) What is the MVP stage in a startup?: London Business Advice, Haines Watts Group. Haines Watts Group. Available at: https://www.hwca.com/accountants-london/opinion/mvp-stage-tech%20startup/#:~:text=MVP%20is%20an%20abbreviation%20for,basic%20principle%20is%20relatively%20simple (Accessed: November 7, 2022).

Burns, P. (2018). New Venture Creation. Macmillan Education UK. https://doi.org/10.1057/978-1-352-00051-1

Lenarduzzi, V. (2017) MVP explained: A systematic mapping study on the definitions of minimal viable product, 2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). Available at: https://www.academia.edu/33157207/MVP_Explained_A_Systematic_Mapping_Study_on_the_Definitions_of_Minimal_Viable_Product (Accessed: November 7, 2022)

Nguyen Duc, Anh & Abrahamsson, Pekka. (2016). Minimum Viable Product or Multiple Facet Product? The Role of MVP in Software Startups. 118-130. 10.1007/978-3-319-33515-5_10.