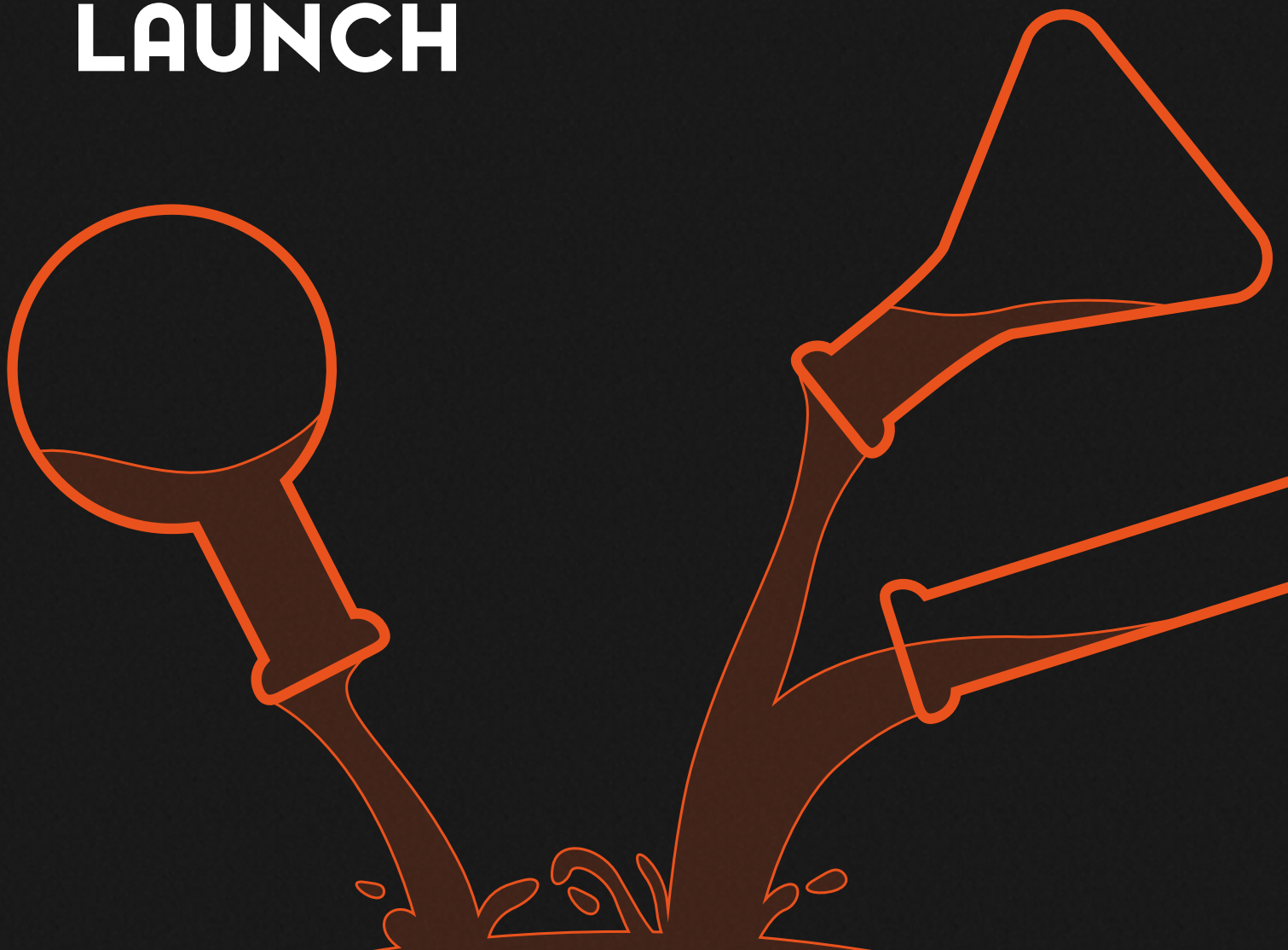
merixstudio

ULTIMATE GUIDE TO COMPOSING

THE RIGHT SOLUTION FOR YOUR PRODUCT LAUNCH



How to mix PoC, prototype, and MVP in the right proportions

www.merixstudio.com

Table of Contents

The start in every startup 03

Proof of Concept 04

Prototype 10

Minimum Viable Product 16

Key takeaways 23

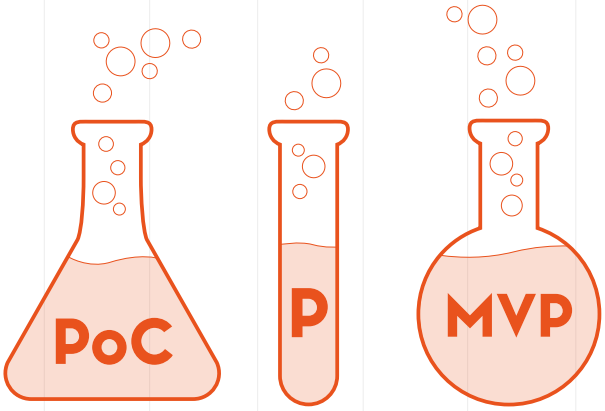
What's the first thing that comes to your mind when you hear the word "startup"? A grungy basement in the suburbs? Silicon Valley high life? Or maybe simply a new beginning? Although there are no wrong answers to this question, the latter association is particularly relevant. After all, **there's a start in every startup** and the beginning is said to be the most important part of the work.

And yet, beginnings are never easy, especially if you consider the multitude of ways to launch a product. To save you dozens of hours of research and prevent you from getting stuck in decision paralysis, we've prepared a thorough comparison of three most prevalent formulas for launching product development: **Proof of Concept, prototype**, and **Minimum Viable Product**.

The shortest way do decide on either of them, is to define the question that the first version of your digital product is supposed to answer:

POC	Is the revolutionary product idea technologically feasible?
PROTOTYPE	How will the product function and look? Will it be usable to the target audience?
MVP	Do the core functionalities of the product fit the market and answer its prospective users' needs?

This, however, is only the beginning. **Oftentimes, it's not about choosing one option but rather mixing some (or all) of them in the right proportions**. Are you ready to find out what solution will work best for you?



PROOF OF CONCEPT

is my product idea feasible?

If you follow the Silicon Valley startups news, you've probably heard about the rise and fall of Theranos, whose founder, Elizabeth Holmes, came up with an innovative idea. To cut a long story short, she dreamt up a medical device capable of running a wide range of laboratory tests with the use of no more than a single drop of blood – and that's within one day only.

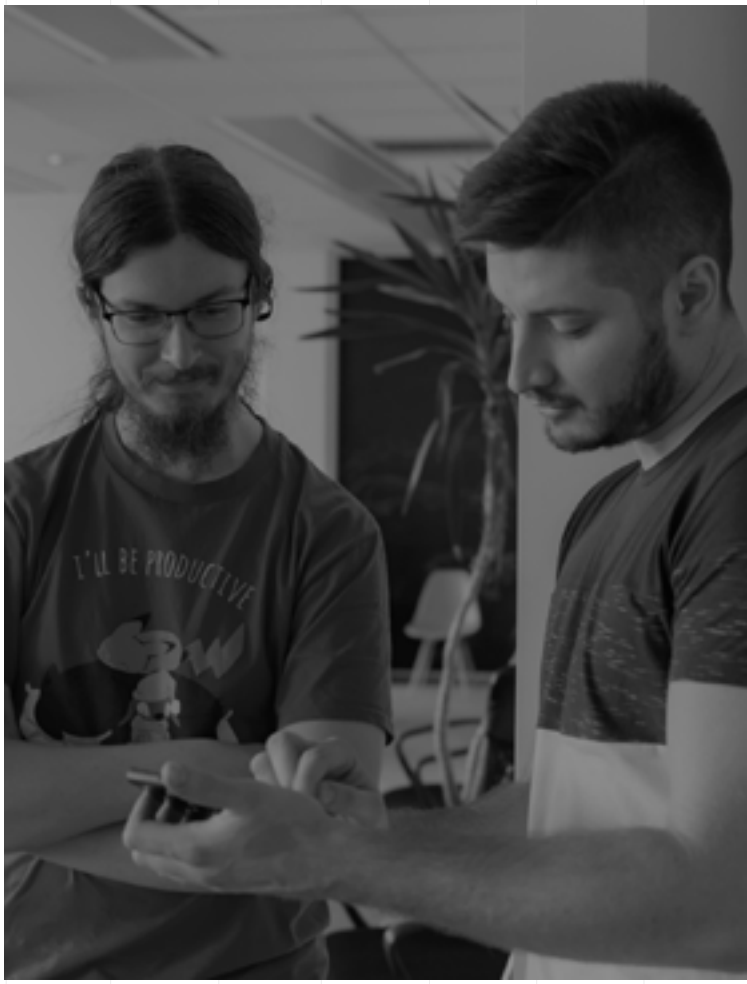
The idea was disrupting indeed and initially met with the approval of the BioTech giants. The problem was, it lacked validation. If carried out beforehand, a series of clinical trials would reveal that Holmes' vision was simply impracticable. In the end, the infamous financial collapse of Theranos was a perfect example of how **the Proof of Concept (or lack thereof) may make or break a business**.

What is a Proof of Concept?

A Proof of Concept can be considered as a very first step on the way to launching a functional product that will simultaneously cater to the needs of its target audience and benefit those involved in its creation. It's not yet a product because it doesn't call for exact solutions concerning scalability or performance. It's more of an exercise in understanding what your project is all about.

In other words, a PoC is supposed to test whether the product you dreamt up is feasible with a minimum resource involvement.

As such, a proper Proof of Concept should primarily provide you with an answer to the following question: **is it technologically executable?**
A PoC is all about subjecting your ideas to a reality check – and in some cases, bringing you back to earth.



When does your product idea call for a PoC?

Try to recall all the apps or services that gained ground in the last five or even ten years. You'll surely notice that while a couple offered unmatched experiences, others hit the jackpot due to providing an improvement to what was already available on the market. Now think about your business idea – which group does it fall into?

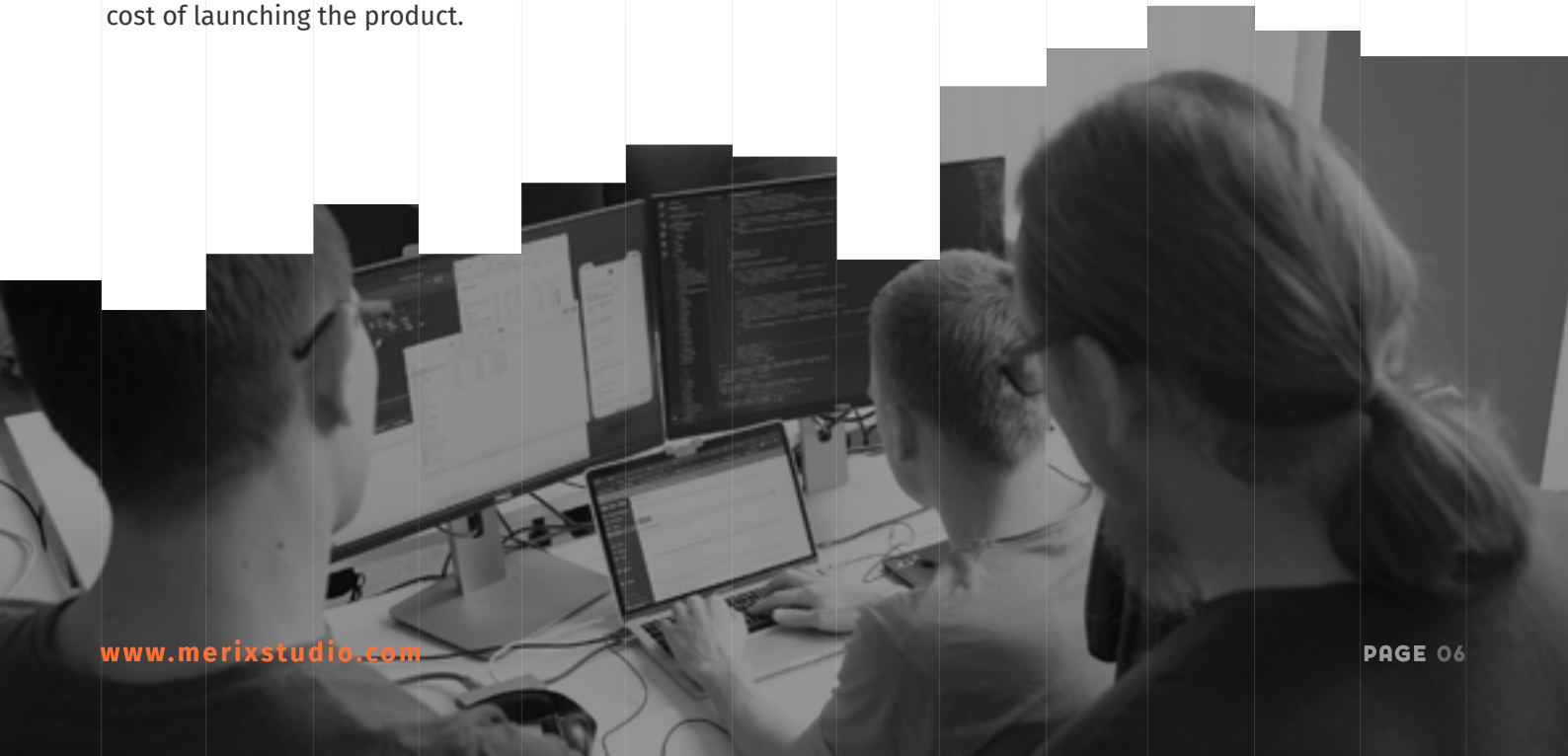
If you have in mind a greenfield project, meaning there are no similar solutions available on the market and there is no competition to look up to, a Proof of Concept is the answer.

While a staggering number of 6.5 million startups are born every year, only 10% survive to enjoy the long term success. The reasons for these failures differ: for some, it's because of not fitting with the market need; for others, it's all about underestimating the cost of launching the product.

But **there are also entrepreneurs who simply tilt at windmills because the products they envisioned are not technologically feasible at all**. Starting the software development process with a Proof of Concept is the first thing you can do so as not to follow in their footsteps.

This first scenario, however, is not as common in the IT world as the one that involves taking an already existing business idea to the next level. In this case, a PoC is not a must. After all, your competitors have already proven the feasibility of a given application and right now, you can simply learn from their mistakes.

To outdo them, however, you'll need an additional unparalleled feature, whose uniqueness may call for a Proof of Concept.



Whether you make it a separate project or a spike in the SCRUM sprint is up to you. What matters, however, is that it will give you a data-based insight into the time and resources needed to break into a competitive software pool.

Benefits of building a PoC

Hands-on experience

When theorizing, it's easy to make a business idea resemble a Frankenstein monster. You continue adding new functionalities in the hope of addressing all pain points that your customers may have until the product becomes unbearable. **A Proof of Concept makes you think about real-life limitations and viable solutions**, even if it's conducted within a limited time and with the help of no more than two subject matter experts.

Risk mitigation

Even a primitive **PoC delivers bits of precious information regarding business logic, usability, and functionality**. It's also helpful for service architects before they launch the MVP development as it becomes the first draft of the project's future roadmap. Moreover, a PoC as a tool in the hand of the entrepreneur informs us about the possible costs.

Gaining allies

Nowadays, investors need some solid evidence of the idea's feasibility before they decide to fund it. **With a Proof of Concept at hand, you assure the business angel of your commitment and down to earth attitude**. And a PoC is significantly better than a PowerPoint presentation.

What makes a perfect PoC?

Once you define the hypothesis you wish to test with a Proof of Concept, look for a software development company with proper resources, experience, and know-how. The process of building a PoC differs on a project basis so it's difficult to indicate how much it should cost or how long it should take. Most often, however, it will revolve around the following steps:

1

Uncovering your needs

A trusted tech partner begins by discussing your product vision with you and doesn't get distracted on the way. They define a primary goal of the Proof of Concept and stick to it so as to make this phase as effective as possible.

2

Forming project team

Don't be surprised if it's a small team of, let's say, a mathematician in charge of an algorithm and a developer whose task is to code it. Although the composition depends on the complexity of PoC, we generally aim at obeying "the fewer, the better" rule.

3

Validating assumptions

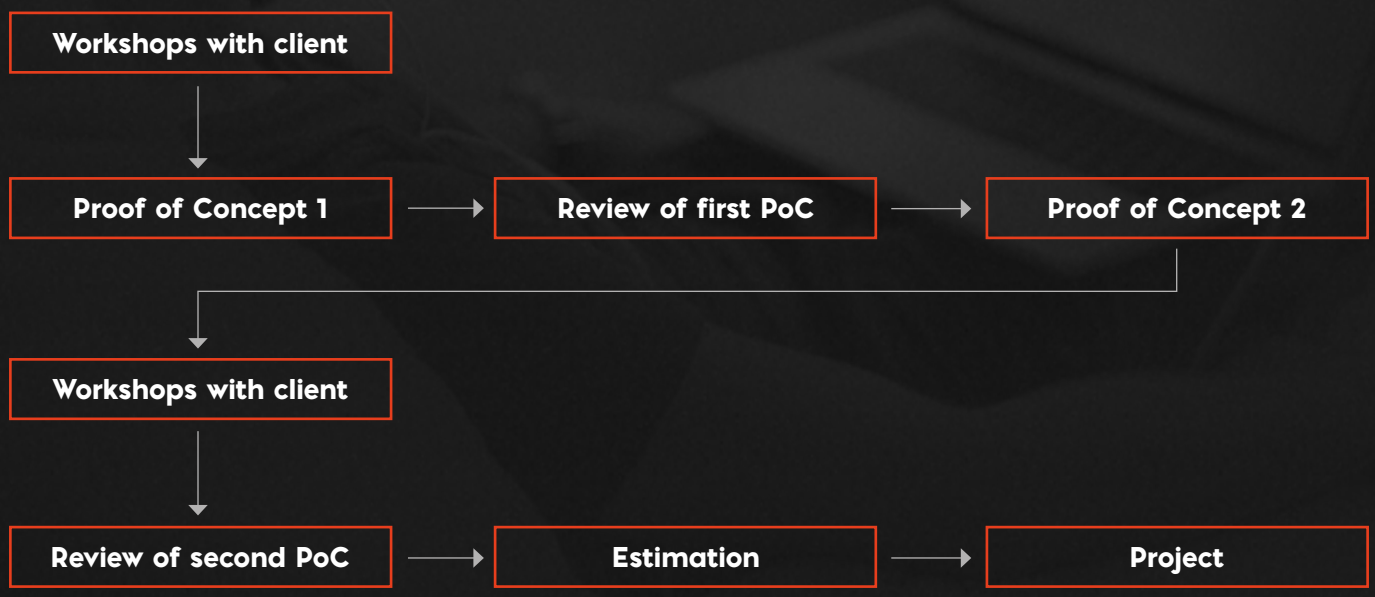
Then, the team develops certain features with the goal of proving their feasibility in mind. At the end of the day, the effects of their work are supposed to lay the solid foundations for the first iteration of your app.

Mexistudio's experience: bookingkit

Founded in 2014, bookingkit offers a SaaS platform that brings the international tours and activities together in the form of a digital inventory. When the client approached us, their goal was to scale an existing product by going from a monolith to a more flexible architecture.

It was a time consuming and costly vision so first, we needed to determine whether it's feasible at all. The challenge for us was also to propose the most efficient way of achieving this aim – and **once we held a technical workshop, we knew that a PoC is essential to validate our assumptions**.

At first, we went for a Python/Flask combination and built a simple version of an API gateway service. This PoC showed that the client's idea is viable. However, it also revealed that when processing loads of data, the platform might not perform as smoothly as we'd want to.



The discussion about our concerns led to the creation of another PoC. This time, we ran a series of tests comparing the performance of two technologies, Python and Go. We didn't need a large team to do that; in fact, the analysis was conducted by one backend developer.

In the end, **the Proof of Concept allowed us to confirm that the client's idea was executable and made it possible to choose the optimal solution** – in this case, Golang as an efficient programming language.

PROTOTYPE

how does my product look and is it usable?

Have you ever heard that "a picture is worth a thousand words"? Believe us or not, this rule applies to business and software development as well. Think of building a digital product. With your concept defined, you could take a risk and proceed straight to the development phase.

But you could also turn the idea into an interactive prototype before actually getting down to code. This way, you'd **test some hypotheses, get early feedback, and make changes without putting a significant amount of resources at risk**. As you can see, Tom and David Kelley, the founders of IDEO, didn't claim that "a prototype is worth a thousand meetings" for nothing.

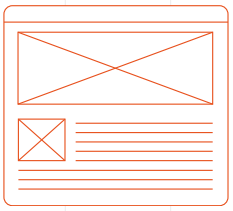
What is a prototype?

In software development, prototyping is a common method of testing the waters before committing to building a full-blown digital product. Since it processes no real data and involves no code, a prototype is not yet the functional version of your application. It's more of a clickable or an interactive model of the product that focuses on the look and feel.

A prototype serves validation of the assumptions underpinning your general business idea as well as initiating the feedback loop early on.

When it comes to the form and the complexity of a prototype, the sky (and a budget) is the limit. The two most frequently discussed types are low and high-fidelity prototypes.

The former **convey the key information about the product**. Rather than colorful designs and meaningful content, they display shapes, hierarchy, and basic interactions. The latter, on the other hand, **not only feel but also look almost like the finished product**. Their design is realistic and the content within is meaningful – sometimes to the point when a user doesn't notice it's just a prototype until they try to interact with it in the more complex ways.



	low-fi prototype	hi-fi prototype
looks like	a simple black and white sketch of the app's interface	a nearly finished version of the product
include	dummy content **few details**	meaningful content **detailed design** branding
good for	brainstorming **early validation** collaboration	testing usability **winning stakeholders' hearts**
pros	unleashes creativity **encourages collaboration** cheap and fast to create	gives a realistic look and feel **provides meaningful feedback**
cons	requires further clarification **puts a limit on what can be tested**	time and budget-consuming
tools	pen and paper **post-it notes** Balsamiq **Mockplus**	Adobe XD **InVision** Axure **Sketch** Figma **UXPin**

When should you build a prototype?

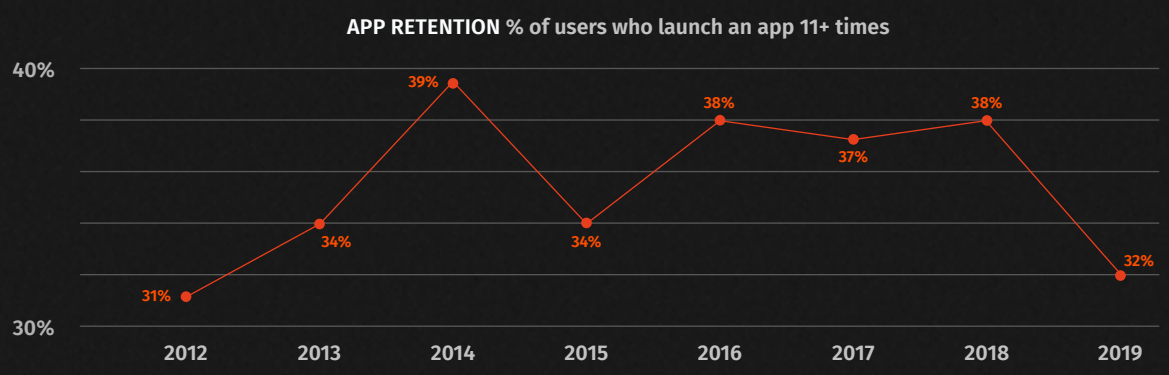
Is prototyping an essential part of the software development process? Let's be honest here: not always. If your goal is to build an application that's just like a dozen others on the market, let's say another simple e-commerce, there's no point in reinventing the wheel.

But there are a number of scenarios in which prototyping makes sense. **Think of a brand new idea for an application or a disruptive feature that can be added to an already existing solution**. If that's the case, building an early visualization of the product will give you a more or less real-life experience of your application and provide a means to verify whether certain features and the designed user flow make sense.

Benefits of building a prototype

Focus on users

In 2019, 25% of users abandoned an application after one use and only 32% liked the product enough to launch it over 11 times. Wouldn't it be splendid if your application lasted longer on the market? A prototype lets you find out how the prospective users interact with your product, what they demand from it, and how big its potential for longevity is.



Source: Localytics, 2019

Ace up your sleeve

Humans are visual creatures – so if you're about to pitch an angel investor, it's usually better to do this using not only words but also pictures. Sometimes a mockup will be enough, sometimes it'll make more sense to prototype a single functionality. Either way, an interactive trailer of your product will show your knowledge and dedication.

Time and cost-efficiency

Everybody wants to be successful but that's not always possible – and usually, failure is the best lesson one can get on the road to success. Starting with a prototype gives you a chance to validate some assumptions and steer the project in the right direction. It can also save you from failing miserably and increase the chances of long-term success.

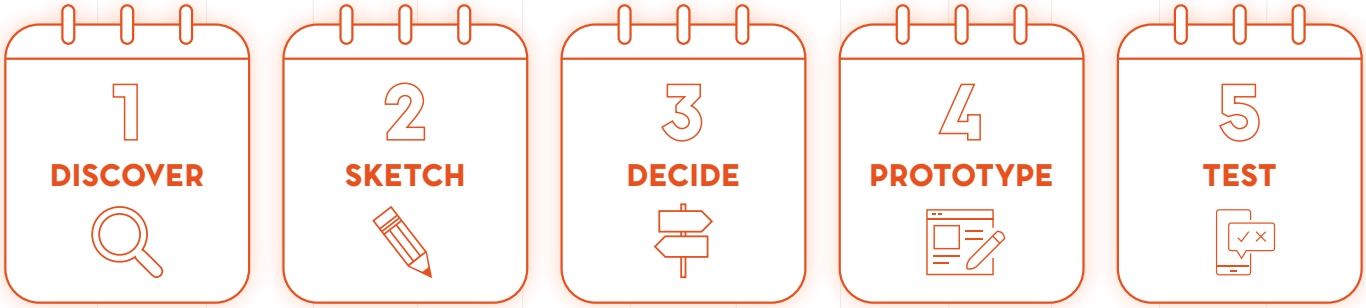
How to build an effective prototype?

Before we delve into particular steps of the well-thought-out prototyping process, let us briefly refer to a design sprint. The framework originated at Google Ventures as a means to align teams in terms of product vision and project goals. In short, a design sprint is

a design thinking method used to solve complex problems throughout co-creation, rapid prototyping, and qualitative testing with targeted users.

It comprises five stages during which the team is bound to progress from a problem to a solution by means of asking questions, testing hypotheses, and finding answers.

Design Sprint



The reason why we're mentioning this framework is that a proven prototyping process is usually comprised of analogous steps:

1. ## *Drafting scenario*

At this point, the goal is to get to know the product inside out. A trusted tech partner ought to ask questions about the needs of both the client and the app's target audience. These insights help brainstorm solutions and suggest optimal user journeys, processes, and information flows.

2. ## *Wireframing*

Now's the time to visualize the ideas and transform them into static views. The key thing is to ensure close collaboration between designers and developers already at this stage. Doing so, we'll prevent the discrepancies between the design and the tech solutions (often limited by the budget) from appearing later on.

3. ## *Prototyping*

In other words, getting down to business. Close cooperation between design and development teams is still advisable if you don't want to risk creating a prototype that's out of touch with reality.

4. ## *Testing*

Here comes the moment of truth. Depending on the project and client's resources, usability testing may take a wide range of forms: from calls involving screen sharing to in-depth interviews. The more data is gathered at this point, the easier it gets to plan the next steps.

Best practices for software prototyping

Start by gathering requirements and brainstorming solutions
Focus on prospective users and their needs
Match your needs with the right type of prototype
Choose suitable tools
Ensure cooperation between designers and developers
Don't go overboard with details
Test the prototype and implement feedback

MVP

how does my product fit the market?

What do Facebook, Airbnb, and Uber have in common? Yes, they all are exceptionally successful companies. But there's more to it: they all started as minimum viable products and it was for a reason.

Building an MVP first comes with a number of benefits: it costs less, takes less time, and helps focus on the most valuable features. It's no surprise that the masterminds behind these world-renowned brands decided to put the "Rome wasn't built in a day" rule in practice and begin the software development process with an MVP. Why don't you try this approach as well?

What does the Minimum Viable Product stand for these days?

The term Minimum Viable Product was coined by Frank Robinson in 2001. Back then, it meant a product big enough to cause adoption, satisfaction, and sales, but not so big as to be bloated and risky. A decade later, Eric Ries came to play with the concept of an MVP as a tool for validated learning and

an imperfect (although not sloppy) version of a product that allows a startup to get something tangible in front of their potential users and learn what they really want.

Over the years, various authors have used the term Minimum Viable Product differently. What's more, the discussion about what an MVP actually is has led to coining a couple of new, closely related concepts, which makes it easy to get lost in the definitions. To prevent that from happening, let's take a look at the terms you can run into when working on your product.

Minimal marketable product (MMP)

a version of a product with a minimal set of features yet still ready for launch

Minimal marketable release (MMR)

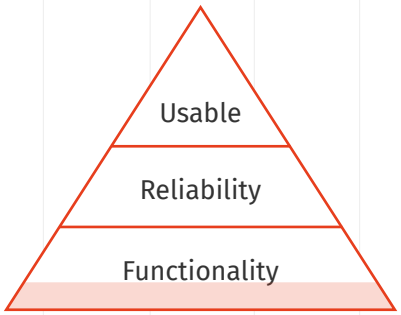
an iteration of an MMP containing the smallest possible set of features that address the users' needs

Minimal marketable feature (MMF)

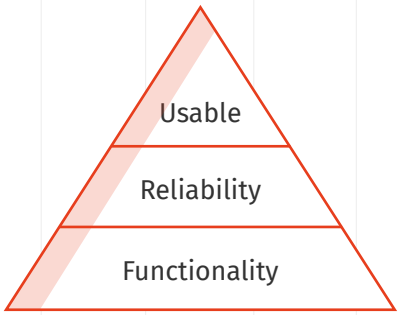
the smallest piece of functionality that gives value to the end-users and to the organization

With all those concepts, it might be easier to say what an MVP is not.
In short, we're not dealing with an MVP in four cases:

1. When the scope is too big and we're trying to build a full-bodied product with fancy features.
2. When we don't know what to measure and how to assess the product's success.
3. When there's no hypothesis, meaning there's no goal for the MVP.
4. When there isn't enough value for the end-user to pay or even sign up for our product.



not an MVP



proper MVP

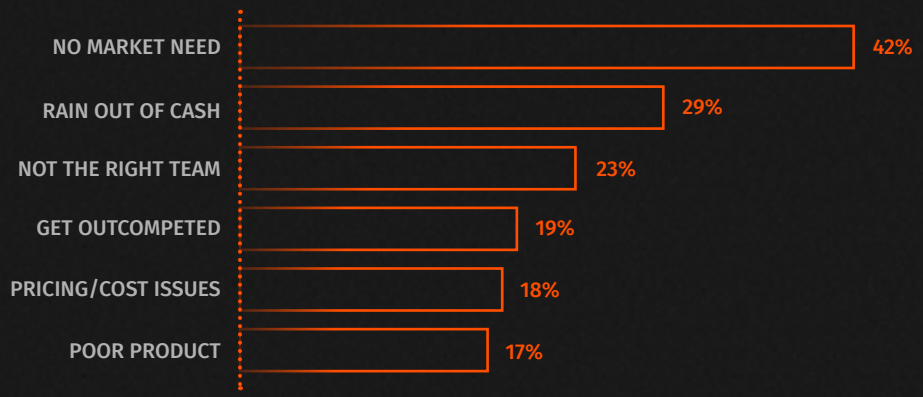
How is MVP building important for startups?

In theory, building an MVP is not a prerequisite for success. In practice, however, starting with a Minimum Viable Product is often considered to be a safe and reasonable strategy – especially if you have a limited budget or a relatively innovative solution in mind.

Benefits of building an MVP

Risk management

The more you delay entering the market, the more time and money you risk. Building an MVP minimizes the risk of investing in a product no one would use. Enough to mention that no product-market fit is the top reason why startups fail according to CB Insights.



Cost savings

The information collected via an MVP makes it easier to assess the potential return on investment. Shorter release loops also mean lower development costs.

Staying ahead of the curve

By focusing on early adopters, you're able to deliver value to the end-users much more quickly than with a full product. Moreover, an MVP gives you the advantage to plan ahead based on the feedback from the market.

How not to go overboard when building an MVP?

When developing a Minimum Viable Product, we're not on the lookout for perfection.

The key is to choose a minimal scope that would address the needs of our potential buyers.

To truly benefit from this approach and save yourself from throwing money down the drain, you need to prepare. And the more thorough the preparation, the better (and less painful) the development.

1 *Research*

Before you hire or outsource developers to build your solution, there's a lot of homework waiting for you. First, you have to clarify your goals – they will act as a lighthouse for all business decisions you'll make along the way. Then, you'll need to know your customers inside out. It's the only way to guarantee that your users would find your features indispensable. Finally, don't forget to research your competition.

2 *Scoping session*

Next stop: a scoping session. Your MVP scope should comprise the backbone features and the first release. Everything else builds up a roadmap showing how the product could develop beyond the MVP level.

How to pick up the core features only?

1. *Prepare user flow and name all possible actions your user can make in your application. These will be your features.*
2. *Assign business value to each feature.*
3. *Estimate the time and budget you need to build every functionality.*
4. *Compare business value with resources you need.*
5. *List the key features for the walking skeleton.*

A scoping session is also an excellent tool to check whether your business knowledge is enough to build a product. An experienced product design team would ask you tens of difficult questions to find potential loopholes and flaws in your product and help you fill them with validated information.

3 *Iterative product development*

After the scoping session, start turning your plans into reality. To stay focused on the business goals, divide your MVP scope into smaller chunks, called sprints. Agile approach will make it easier to plan, prioritize, and then test particular product releases.

Mexistudio's experience: Autium

Autium is the brainchild of a UK-based tech startup whose founders had the first-hand experience of fleet management. Knowing how cumbersome insurance claim processing can get, they came up with an idea for a platform enabling automated incident management.

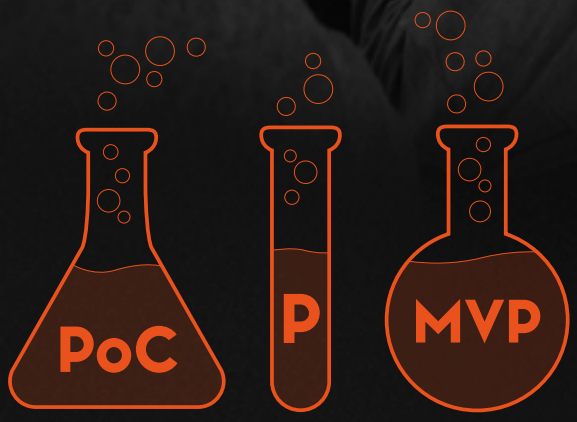
Presented with this vision, we drafted the first version of estimation and pointed to the need for clarifying it further during a product design workshop. **At that point, our goal was to define the client's objectives, confront them with the end-users' expectations, and select core functionalities accordingly**. We drafted the project roadmap for the next iterations and refined the budget estimate.

Autium decided on starting with an MVP for a reason. **Their goal was to build a functional product that would truly benefit its users – and they knew this can happen only with the help of their target audience**. With an MVP at hand, they approached the prospective customers and started gathering feedback which we used to improve the following versions of the platform.

Version 1.0	Version 2.0	Version 3.0 (in progress)
accident reporting	daily inspections	fixes
admin panel	defect workflow	planning new functionalities
vehicle and driver database	approaching technical inspection date notifications	
real-time data and analytics		
document generation		

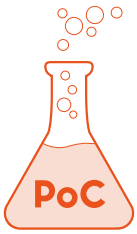
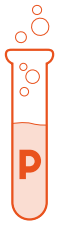
It wasn't only the iterative approach to development that made the MVP successful. **Based on our previous experience, we also went for the solutions which we consider reliable and relatively fast to implement – especially that both the budget and deadline were tight in this case**. As the mobile app was to cover both Android and iOS, we decided on a cross-platform solution that ensures faster and cheaper deployment: React Native. For backend, we chose Django – and thanks to Django admin, we didn't need to reinvent the wheel. In the end, the MVP development spanned around 1900 manhours.

The client is still collecting user feedback and together, we're working on version 3.0 of their product. As you can see, **validating your business idea against user expectations and using feedback to improve the product is much more than just a theoretical best practice**. Matched with the software development expertise, it can truly work to your advantage.

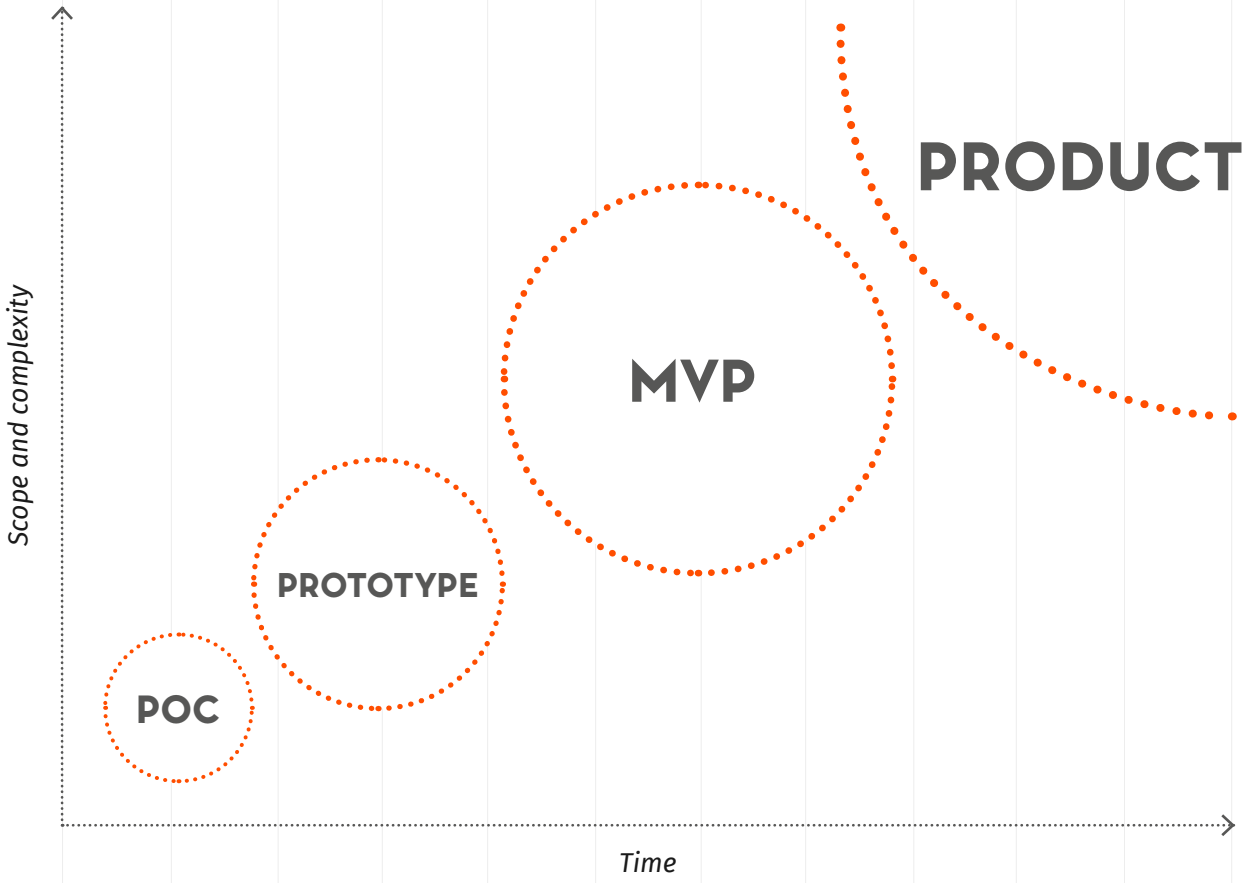


POC vs PROTOTYPE vs MVP

Although similar at first glance, the three concepts differ considerably – especially in terms of purpose each of them serves. The choice which one to start with depends on many factors, including your current state of knowledge or the budget you can spend on a given project.

	 PoC	 prototype	MVP
Key objective	To check the solution's feasibility	To see how the product will look and work	To provide value for the users and enable validated learning
Cost	$$	$-$$	$$$
Audience	Stakeholders	Stakeholders, representatives of the target group	Users
Revenue	No revenue but can help obtain seed money	No revenue but can help obtain seed money	Can be sold but also can help you secure funding for next iterations
Technology	Calls for tech expertise to check different scenarios but minimal or no UX/UI	Focused on design rather than code	Requires the involvement of the entire team, including designers and developers
User experience	Doesn't take users into consideration	Focused primarily on user interaction	Helps gather user feedback and implement changes accordingly
Reusability	Rarely reusable or scalable	Effects of UX and UI work can be used later on	The first iteration of the finished product

You can think of a Proof of Concept and a prototype as tools that will give you a broader perspective on the project and allow you to test certain assumptions empirically. Matched with market research and a comprehensive business plan, they can make a thorough analysis of the project. An MVP, on the other hand, will show you whether there is a demand for your product or not as well as indicate the direction in which you should be heading.



At the end of the day, it's up to you to decide whether you wish to go the full PoC → prototype → MVP circle or structure the process of building a digital product your way. To help you make an educated decision, we've prepared a cheat sheet that matches the three concepts with some common scenarios in which entrepreneurs find themselves.

Which model will fit me best: a cheat sheet

	PoC	prototype	MVP
Is your product idea one of a kind?	✓		
Not sure if your idea is technologically feasible?	✓		
Don't want to jump straight into development?	✓	✓	
Are you looking for seed money?	✓	✓	
Do you want to visualize your product first?		✓	
Are you enriching an existing product with new functionality that needs testing?		✓	
Are you focused on user experience?		✓	
Is your vision defined and detailed?			✓
Do you want the product to turn a profit quickly?			✓
Do you wish to gather feedback from actual users?		✓	✓
Do you want to get stakeholders on your side?	✓	✓	✓

Stuck in the decision paralysis?

Contact us and let's mix the product launch solution that meets your needs together!

contact@merixstudio.com

+48 570 001 928

ul. Małachowskiego 10, 61-129 Poznań

See our work Bēhance dribbble merixstudio

Check what our clients think about us Clutch

Follow our stories

merixstudio

Design & development team of creative technologists