The UI Audit

BY JANE PORTMAN
Introduction
Why only you, as the founder, are responsible for design decisions

As the founder, only you are responsible for making important design decisions. Why?

Most UI designers will never question the key functionality of your web app. First, designers are trained to work with existing material: just make things beautiful and don’t reinvent the wheel.

Second, they just don’t have the guts to question your major UI decisions. Why would they jeopardize the new fragile relationship with you? They just want to be nice and get paid in the first place.

As a UI/UX consultant, I’ve seen my fair share of crazy startup ideas. A great number of them never saw the light of day. But these founders were paying me money to design their super-opinionated software. They had the right to experiment with their own resources — it’s their unquestionable founder privilege.

I always did my best to make any product attractive and usable, but I definitely compromised more than I argued. Sure, it got better with time: a hefty price tag empowers a consultant to be more vocal. But the tone remains cooperative. That’s how the business works.

Moreover, designers sincerely trust your opinion — and for a good reason!
When invited into a project, they usually know nothing about the product and the target audience. You, in return, are the one who knows the customers inside out. You talked to them, researched their pains to death, and have a clear understanding of things.

**Whatever design decisions you sign off, it’s always your own time/money/success at stake.** So you’d better come prepared with some UI/UX knowledge. You’d better know how to build a simple, focused product. Design is an important tool to have in your belt.

And this book will help you!
“Good enough” will absolutely do the job

In the ideal world, you’d love your product to have brilliant UI design. You’d love to receive compliments, oomphs, and wows every day. You’d love to see your web app nominated for the “Best in Design” award.

In reality, a user interface is rarely perfect — and this fact is rarely as critical as we think. Not ugly is enough for you to gain initial traction and grow to a decent MRR, until you’re able to afford a professional designer.

Here’s what Ankur Nagpal, the founder of Teachable (previously Fedora), says about software design:

“My key advice is, early on focus on making things not ugly. And then I think it's two stages — first tier of design is 'not ugly,' second stage is making things beautiful. The good thing is, in today’s day and age there's so many tools you have at your disposal to make things look not terrible. And I think that's something you should totally leverage.”

Whenever the subject of UI design drives you crazy, please refer to these two lists. They will help you stay sane.
Here's what your UI design should do:

- do what you promised on your marketing website;
- get out of the user’s way while they’re accomplishing their goals with your software;
- satisfy your own taste: you’ll enjoy selling software that you consider good, which is necessary for your well-being as a founder;
- be decent-looking, in order not to scare customers off.

Here's what your UI design shouldn’t do:

- please everybody (it’s impossible);
- be a piece of art (utility is way more important);
- be unique (unique interfaces have a steep learning curve).

Are you familiar with the concept of good enough parents? This concept exists because it’s impossible to be perfect parents. Nor is it possible to achieve perfect design in a living product, especially if you’re bootstrapping your business. It took me years of consulting practice to understand that. It took me years of sweat and frustration to understand that perfection is virtually impossible. That true polish costs thousands of dollars and is only affordable for the top of the top.

For your SaaS, good enough UI design will absolutely do the job.
Design is merely an educated guess until the product is live

Design is a very subjective field. First, we try to make the product visually appealing to the customers. That’s hard enough by itself, as tastes differ dramatically.

Besides their taste, we also try to predict their behaviour, goals, restrictions, and capabilities. This all takes us to the land of unlimited assumptions — until the product goes live and gets into the hands of real customers.

Plus, design never works in isolation. There are always other variables involved, like positioning, pricing, marketing, copywriting, and whatever not.

So everything we talk about in this book is a highly educated guess. It’s based on best practices, common sense, design theory, and my ten-year experience — yet it remains a guess until the product is live. Nevertheless, I assure you that these intangible, ephemeral improvements can make your SaaS business some real money.
We’ll be talking about web applications

This audit method was developed specifically for web applications. I conducted a number of client audits and reviewed my friends’ software, all to find and analyze common design issues.

Everything here relates to a full version of your web app — not mobile or responsive. Mobile is definitely something to take into account, but not here, not this time.
My pledge to you, the founder

A great book is created on the intersection of what the readers want to learn, and what the author thinks they really need to learn. I know you came here for quick design recipes, lots of images, and real-life examples. In return, I came here to explain that the functionality matters much more than pretty looks, and that you need to think about product strategy before even touching the UI.

This book has no illustrations: it’s an opinionated decision that I made after completing the draft and getting beta-readers’ feedback. “Don’t worry about the illustrations, the content speaks for itself,” — one of my friends said.

So this book is a set of exercises you can do with your own product. When you’re reading about a certain issue, bring up your own UI (on the screen or in your mind), and things will instantly fall in the right place.

Each chapter has a brief practicum in the end, which says how exactly you can apply this knowledge. Worksheets from the premium package do the same thing, just in a more extensive and attractive manner. Fully transcribed founder interviews let you peek into founders’ minds. They make excellent listening material, but I did my best to include the most interesting quotes into the book itself.
I pledge allegiance to you, the founder. I have the highest respect for your role. I have the highest respect for your time. I’m doing my best to help you, the founder, build that good enough UI that will make your customers happy, and bring you profit and peace of mind.

Let’s dive in! Let’s see what exactly you can improve in your product design, and how. The best start is defining your product strategy: what audience you’re serving and what exactly you help them to do.
Chapter 1.
Your Product Strategy
Why product-market fit is important

In this chapter we’ll break down your product strategy into edible questions. But before we get started, let’s look at a different term – *product-market fit*.

This term was first coined by Marc Andreessen in 2007: “Product-market fit means being in a good market with a product that can satisfy that market.”

Product-market fit is the key thing you should care about as a SaaS founder. Here’s what you get when you nail it:

- you stop worrying whether people actually need your product;
- your sales are great;
- you have enough confidence to niche down more on a certain type of audience (which makes your message more powerful and further accelerates customer acquisition);
- customers are highly satisfied with a product that fits their needs so well, which results in the most effective “word-of-mouth” marketing.

In his story of building Drip, Rob Walling describes the period of searching for the product-market fit as the most painful and frustrating step. Their big hit was nailing the precise formula “email marketing automation” (both words and functionality). Then they added features to match this new value proposition, for example their famous rule engine. This reduced churn dramatically and caused nearly hockey-stick growth.
Here’s what Rob said in his interview for this book:

“We didn’t just want to build features — that doesn’t help. You have to have some type of vision for where you’re headed. And when we made the decision we were going to become like a lower-cost, high-value marketing automation platform, then we instantly knew what to build and what not to build. You’re not going to build shopping cart software onto it, which some people are requesting. You’re not going to build an affiliate management program; you’re not going to build landing pages probably. You’re not going to build that CRM upfront. There’s a bunch of things that you don’t need, and then we can really focus on exactly what we needed to build.

As we started deploying the automation engine over the course of several months, it was obvious that our churn was dropping and our trial-to-paid conversion rate was going up, and that was what was getting us towards product-market fit.”

Peldi from Balsamiq was lucky enough to hit the product-market fit in the first few months. He was able to do so because he was hyper-aware of the audience and their exact needs: simple, usable wireframing software. As a result, the story of Balsamiq became a textbook example of startup success.
Product-market fit vs. product strategy

In UI/UX terms, product-market fit means implementing all the promised functionality well, and streamlining the UI to meet customer needs and expectations. That is, getting out of their way while they accomplish their goals using your software.

Imagine, you make an irresistible value proposition. Your customers respond well to your marketing page and sign up for the product. What they see inside your product does or does not match their needs and expectations.

From there, we have two possible scenarios: if the users fail to accomplish their goal with your software, they churn, which means your marketing money is wasted. If they succeed at their goal, they become your customers, make you money, and bring in new customers by word of mouth.

UI/UX absolutely depends on the variables in this formula: who your customers are, and what they came here to do. By tweaking these variables, you define your product strategy.

A certain product strategy may (or may not) lead to the product-market fit. To test your strategy in real life, you need to fix these variables — at least for a given period of time — and build a solution based on them.
In this chapter, you’ll determine your product strategy by answering the following questions:

1. Who are your ideal users?
2. What big goal do they have in mind when they sign up?
3. What tasks do they perform daily when they log into your web app?
4. What objects do the users handle while performing these tasks?
Question 1. Who is your ideal user (paying customer)?

In the ideal world, you first define an audience, then research their pains, and finally build a product that solves that pain. But I wouldn’t be surprised if you don’t have a solid understanding of your audience yet. This is a very common mistake. There’s a plethora of products out there (built and launched) which are now “looking for an audience.”

So it always helps to repeat the basic drill. Take a sheet of paper and write down what audience you’re targeting!
Here are the criteria you can use:

- Basic social criteria: age/gender
- Professional skill set (designer, developer, copywriter)
- A certain stage of professional/personal development (student, employee, freelancer, consultant, product owner, business owner)
- A certain stage of business development (getting ready to launch, gaining initial traction, scaling)
- A certain function they perform within their business (email marketing, sales, social media marketing, user onboarding)
- Certain software products they’re already using
- Certain places they hang out online
- Certain books/blogs/websites/forums they read or participate in
- Certain events/conferences/meet-ups they attend

Samuel Hulick has selected a spectacular niche for his training materials and consulting services: user onboarding. However, he doesn’t qualify his audience by skill set or occupation. Instead, he uses a purely functional qualification: people who are responsible for user onboarding.

Anchoring to a software product is also super-effective. You gain a whole set of proven data (much better than vague customer interviews): what work users perform, how much they pay, what language they use. You can explore their support forums, listen to customers, and identify existing problems. Your own value proposition might be solving these exact problems!
Here’s how Ankur Nagpal, the founder of Teachable (previously Fedora), talks about approaching their huge audience of people who make money with online courses:

“We don’t think of a photography teacher different from a programming teacher, or a dance teacher. We’re thinking what these people were doing before. Ultimately, most people we focus on are people with an audience. So we think where do they have their audience from: do they have a large podcast, a large blog, large social media? And that’s how we group them and build marketing funnels towards them.

We can build marketing materials, training materials specifically to people that write books. In the next phase, we can build marketing specifically to people that make YouTube videos. And that’s how we think about the market — not by what people are teaching, but where they built their audience from and how they spend their online time.”
Here are a few more qualifying questions to make sure you’re making a wise choice with your audience.

- Do you know them well? Knowing a certain ecosystem is a splendid competitive advantage, so you’d better capitalize on your previous experience. While *not knowing* your customer ecosystem can ruin your business!

- Do you like them? As Amy Hoy says, “*If you don’t like drunk frat boys, don’t open an Irish pub.*” Serving people you don’t like/respect can be rather excruciating.

- Can they pay you? Are they used to pulling out their credit card online? Do they have a habit of paying for SaaS products?

- Do you know how to reach them? Do you know what forums/sites they go to? Some customer categories are extremely “physical” and hard to find online. In such cases, plan for real-life events or locations instead.
Question 2. What is the big goal that the user is trying to achieve with your web app?

Your product solves a pain. What does the user try to do when they encounter that pain? What is their big goal?

Always keep this goal in mind and cultivate the success related to it. Your stats and metrics should display the progress towards that goal.

Good examples:

- write a book;
- publish a podcast;
- acquire new SaaS customers.

There’s more to this question. In fact, this goal isn’t really “big” in the user’s own world. It’s just prominent enough for you to build a product around it. Beneath this “big” goal always lies a sequence of other more important goals.

This whole system of goals doesn’t directly dictate UI decisions, but it definitely helps when you evaluate potential new features or write sales copy for your product.
Here’s a very basic example to give you an idea of scale:

- (I want to) *write a book*;
- (because then I can) *build authority*;
- (because then I can) *have better clients*;
- (because then I can) *make more money*;
- (because then I can) *travel more*.

That’s a linear sequence, but in real life it’s a huge network of interrelated goals and aspirations. There can easily be another branch for the same person:

- (I want to) *write a book*;
- (because then I can) *have passive income*;
- (because then I can) *work less*;
- (because then I can) *spend more time with my kids*.

Look how these goals range from simple facts towards big lifestyle changes! The goal related to your product (“write a book”) is merely a small step in a huge life picture of your customers. But if your product does its job well, then they’ll be able to focus on their bigger goals and achieve more down the road.

That’s why they’re paying you, not because your app has pretty design. Not because it’s cheap/expensive. Not because you’re awesome. Realizing that will make you a humble founder with a small ego, able to build a useful product without falling in love with it. That’s your recipe for printing money.
Question 3. What primary tasks does the user perform on a daily basis when they log into your web app?

What is the user's daily routine on the way to their goal? What do the users do when they log into your app? List the actual creative processes, procedures, etc.

A polished, streamlined execution of a few key tasks often makes a difference between good and bad software. You will need to move heaven and earth to make these tasks obvious and easy to handle.

Here’s what Ankug Nagpal of Teachable says about it:

“When it comes to building a product, there’s two things you can do. One is you can take the things people don’t like and make them better, so they like your product overall. But what I personally like to do is find out what people like about the product, and make your strengths stronger. Because that’s what makes people love a product — rather than a product that’s just okay in every regard.”

Tasks can be classified in three groups: analytical, proactive, and reactive. Each group is handled in a different way.
Analytical tasks mean monitoring the current state of things or analyzing performance. To do so, the user looks at the dashboard or digs into reports.

Proactive tasks mean creating new objects and editing existing ones. That’s what we usually call “work.” Some proactive tasks might require a comfortable workspace environment, some can be handled with simple forms. In any case, proactive tasks revolve around object lists. Where do all my objects live, so that I can review them and add more if necessary?

Reactive tasks mean handling incoming items: replying to messages, handling bugs/issues, approving reports, etc. Reactive tasks require a separate system:

- notification area (a message/bell icon somewhere in the top navigation bar);
- instant notification mechanism (pop-up notifications and such);
- a certain location in the app where all incoming and outgoing items are permanently stored (inbox or similar).

Email by itself *should never be a single way* for the user to hear about any updates, even if it seems to work. The user should be able to comfortably work without leaving the comfort of your web application. Switching between the email client and the working environment can be detrimental to productivity.
Usually all three task types (analytical, proactive, reactive) co-exist within a single web app. If a certain type of task clearly prevails, this can affect the way customers use your software. Your web app can be forgotten or abandoned in the following cases:

- there are no reactive tasks for the user to act upon (no new messages, issues, etc.);
- there’s no external reason for the user to do proactive work with the help of your application (this often happens to freelance tools when the client work runs out);
- your web application is purely analytical (then you’d better send out some positive email reports).

That’s the worst situation to be in with your SaaS product: these users only remember your app once a month when their card is charged. That doesn’t evoke any positive emotions, just the urge to cancel the irritating monthly charge.
Question 4. What objects does the user work with?

What objects does the user work with? What entities do they create and manage while performing their tasks?

To answer this question, try this: crawl your list of tasks and list all nouns from there. Most likely, that will be the core of your list.

*Good example:* “subscribers, email campaigns, automation rules.”

Why is this list of objects so important to be a part of the product strategy?

- Every UI in the world facilitates tasks, but it doesn’t display them — it displays content (a set of objects).
- Objects help to limit the scope of your functionality and responsibility — what your application does manage and what it doesn’t.
- Objects dictate the data structure in the development process.
- The primary purpose of navigation is to clearly classify and locate all objects.
- The primary purpose of onboarding is to tell the user what objects he’ll be managing and how they’re named (while their location should be naturally obvious from the navigation system).
Important: use your customer’s vocabulary

The exact words you use to describe your product strategy are equally important as the strategy itself. The customers know what they want to do and have their own words for that — your job as a founder is to decode this language and build a web app that’s easy to understand.

There’s a common term for small, instructional bits of copy within a UI — microcopy. It’s a very important subject, but we’re not talking about microcopy here. We’re talking about the words from the product knowledge domain that are used inside your web app.

The exact words — your product vocabulary — should possess the following qualities:

- be familiar to the users and belong to their knowledge domain;
- be consistent across the web app and marketing materials;
- make general common sense.

Names for the same things can vary greatly from one product to another. Let’s look at the vocabulary of three popular email marketing tools:

- One-off email: campaign (MailChimp), broadcast (Drip, ConvertKit).
- Email sequence: automation workflow (MailChimp), campaign (Drip), course (ConvertKit).
Vocabulary depends on your positioning, too. ConvertKit, for example, was originally created for content marketing, so email sequences are historically called *courses*. Drip is a more versatile piece of software that targets many customer groups, so they use a neutral word *campaign*.

It helps to borrow terms from established software products that your users know well. Drip and ConvertKit successfully use the term *broadcast* for one-off emails, so this term is already well-coined for the audience. If you were building a similar tool, it would make sense to use the same language.

It’s a great practice to write down your vocabulary (with explanations) in a single document, and use it for reference. You can later use this exact document for your support knowledge base.

**How do you build a vocabulary that makes sense to your users?** Don’t write it from the top of your head. Instead, research and re-use the existing language of your audience!
Here’s what you can do for your research:

- actively organize opportunities for your audience to speak up: interviews, surveys, feedback rounds;

- scour existing resources and materials: blogs, forums, product reviews, and support threads (Amy Hoy calls this *safari* because you study customers “in their natural habitat”).

Usually these activities take place during the product research stage, but you can jump in again while defining the vocabulary. This time you already know what you’re building exactly, so the purpose will be a bit different: figure out *what exact words* customers use while speaking about their goals, tasks, and objects.
Practicum

Take a sheet of paper and answer the four questions above.

1. *Who are your ideal users?*
2. *What big goal do they have in mind when they sign up?*
3. *What tasks do they perform daily when they log into your web app?*
4. *What objects do the users handle while performing these tasks?*

Congratulations, you have now defined your strategy! It should be a strong, focused vertical: *audience — goal — tasks — objects.*

Whenever you're working on your product — building the prototype, doing the UI audit, adding new features — make sure you take all decisions with this strategy in mind.

Keep this worksheet for future reference: we'll use it for auditing your navigation and key screens.
Thanks for reading through this sample chapter!

Product strategy is important, and I hope the first chapter gave you a fresh perspective of your own product. But it’s merely a beginning of the whole UI/UX journey. Here’s what’s waiting ahead:

- Chapter 2. Navigation
- Chapter 3. Dashboard & Homescreeen
- Chapter 4. Audit Your Screens
- Chapter 5. The Problem of Style
- Chapter 6. Get a Theme
- Chapter 7. Plan for Improvements
- Chapter 8. Deal with New Features

If you enjoyed this sample, you’re welcome to get your copy of the book here: uibreakfast.com/audit

Feel free to share this free chapter with your friends who might be interested in improving their SaaS products. Good luck in your design journey!

Regards,
Jane.
The UI Audit
by Jane Portman

Author          Jane Portman
Editor          Ine De Baerdemaeker
Founder Interviews  Tope Awotona, Giacomo (Peldi) Guilizzoni, Ankur Nagpal, Josh Pigford, Derrick Reimer, Rob Walling

Copyright © 2016 Jane Portman. All rights reserved.

This book is a result of long-term effort. There are no DRM restrictions embedded in this file, but hopefully you are a grateful reader. If you enjoyed this sample, you're welcome to get your copy of the book here.

You are always welcome to quote a few paragraphs from this edition online or in print, but make sure to provide a link back to www.uibreakfast.com/audit. If you want to reproduce something more substantial than a small passage, contact me at jane@uibreakfast.com and we will figure it out. I always do my best to be helpful and respond to all inquiries.