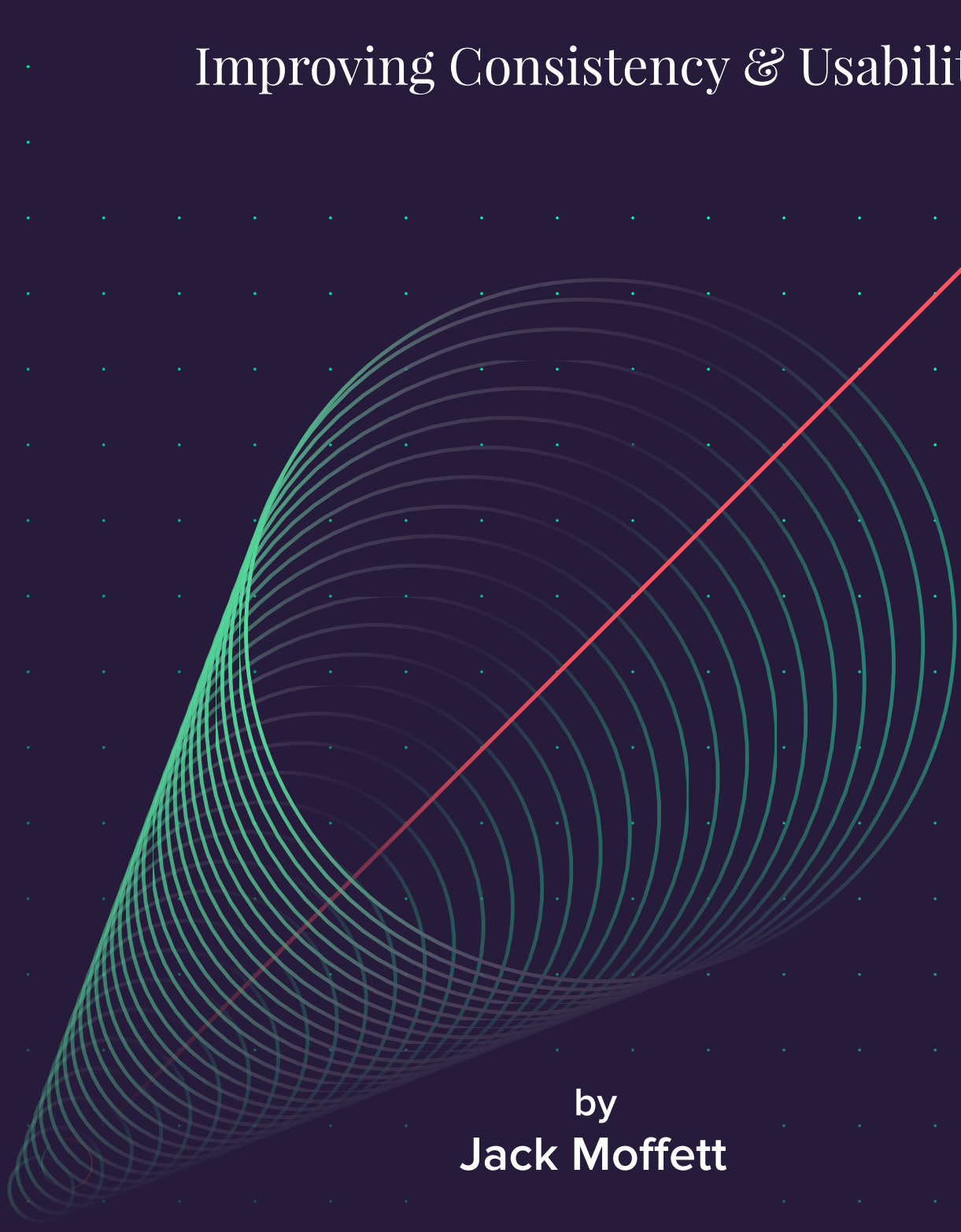
UXPin

Eliminate UX Debt

Improving Consistency & Usability



by

Jack Moffett

UXPin

Eliminate UX Debt

Improving Consistency & Usability

Index

Authors

Jack manages the UX group at Inmedius, a Boeing Company. With a Masters in Interaction Design from Carnegie Mellon, he has been designing web, desktop, and mobile applications for over 15 years in both research and industry environments. Jack has designed software tools for Lockheed Martin, Shell, DaimlerChrysler, Eaton, and many organizations within the U.S. military. He teaches design part-time at WVU, authored *Bridging UX and Web Development*, co-founded and leads IxDA Pittsburgh, chaired Midwest UX 2015, and writes about design on designaday.tumblr.com.

A Few Quick Words

When Adobe wanted to expand beyond photo editing and join the page layout market in 1994, it acquired the company Aldus and its product, Pagemaker. They repeated this strategy in 2005, acquiring Macromedia and its numerous software applications – including Flash, which had become a very successful web platform.

They were following a tried-and-true method for increasing a company's value, but think about the product consequences. All of a sudden, Adobe had several products designed and built by others, all following different visions, different aesthetics, and different behavioral philosophies. When a new product is suddenly dropped into a team's collective lap, it doesn't take long to uncover residual issues with the product.

Whether it's a poorly-worded confirmation message that slips out the door or a strange behavior that fell through the cracks, it's really easy for issues to slide by unnoticed. Before long, a product or three falls short of intended standards and develops an ever-growing list of "We'll get to it eventually" problems.

In other words, **UX debt**.

More than likely, your UX team is already tracking and fixing scattered issues in your own products, as well as those that suddenly became your team's responsibility due to acquisitions or outsourcing.

Fixing UX debt becomes considerably more complicated – and more critical – within a large company developing an array of products. You know this all too well.

Whether you're part of an established team that is seeing renewed interest in your capabilities, a new team that's been recently assembled to catch the UX wave, or a UX team of one, I'm sure you have a pile of debt to deal with.

I'm here to help you identify your debt, classify it for prioritization, and ultimately eliminate it. Even better, once you recognize the sources, you can prevent debt accumulation in the future.

Jack Moffett
Manager Apps Development – GUI
Inmedius, a Boeing Company

Classifying UX Debt: An Overview

First, let's define UX debt.

UX debt is an accumulation of design and development **decisions that negatively impact the users** of a product or service.

All UX debt is either intentional or unintentional. We should seek to minimize the intentional debt, while proactively avoiding the unintentional debt.

Reasons for intentional debt include:

- **Time to market** – A company might release a product with design debt to meet strategic timelines. For example, Series B startup might release an imperfect mobile app to better position themselves for a faster Series C investment.
- **Focus on new features** – A company may decide a "critical mass" release of robust features for a new product is required to gain market share. The team will then schedule design debt cleanup for later sprints.

Reasons for unintentional debt include:

- **Separation of design and development** – When UX is treated as an island, design debt is inevitable.
- **Design by committee** – When design leadership needs to satisfy everyone's requests, the product becomes convoluted. The inconsistency creates UX debt.
- **Lack of access to end-users** – If the product team can't test concepts with end-users, the design is educated guesswork at best.

Regardless of the reason, unaddressed UX debt will eventually lead to products that are so painful to use that customers seek out competitors. UX debt is particularly common in enterprise contexts due to product and organizational complexity.

The below overview of classifications will help you better weigh your options for intentional debt, and better identify and eradicate your unintentional debt. Even if you aren't able to fix a particular issue yourself, understanding the concepts will help you better pick your battles.

Technical Debt

Stop and think about the times developers asked for compromises or shot down your designs citing technical limitations.

Perhaps they said that your requests were too time consuming, wouldn't perform well, or just weren't possible. These cutoffs and bypasses alter the intent of your design, in ways that wouldn't be necessary with a modern, well-maintained technology stack. This is technical debt.

Technical debt comprises two subcategories: Back-End and Front-End Debt.

1. **Back-End Debt**

   Even minor changes in this part of your stack can irreparably undermine your product's usability. Back-end debt segments into four key areas:

   - Performance
   - Hardware
   - Database
   - Security

   All four aspects are deeply entangled; an issue with one typically pulls in others.

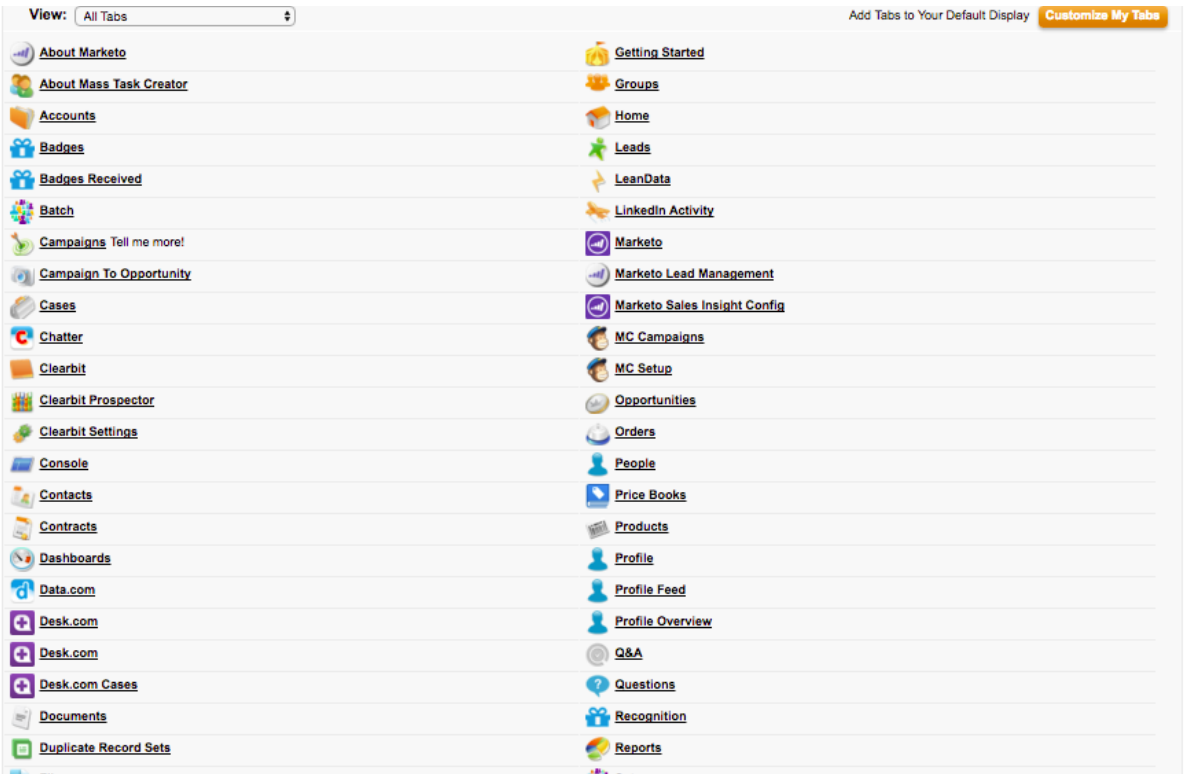
2. **Front-End Debt**

   Your technology stack greatly impacts the types of debt you're going to find here. Since I primarily work on browser-based web applications, I classify front-end technical debt as follows:

   - Browser Version Support
   - Outdated HTML
   - Outdated/ Non-responsive Frameworks
   - Poor Coding Practices

Functional Debt

Functional debt is usually the result of naturally-evolving shifts in requirements (e.g., technology changes, customers needing new capabilities, features added to attract a wider customer base).



You'll primarily encounter four types of functional debt.

1. **Scale**

   If we don't regularly set new performance goals for our products and improve their UI in order to deal with scale, they will eventually choke. Our users also need efficient tools for managing data at increased scale. This means improving search capabilities, providing more sorting and filtering options, and making it easier to perform bulk actions once desired data is found.

2. **Information Architecture**

   As hard as we try to account for future growth, we aren't fortune tellers. Bolt-ons can turn carefully-planned architecture into Frankenstein's monster. We are forced to make compromises to get the job done because we don't have the freedom to rearrange content and navigation each time.

3. **Old Features**

   Enterprise product teams are often loathe to remove features, just in case some customer still uses an obscure function. But these little-used tools may be harming users as a whole – interfering with features the majority are using.

4. **Priority of Functions**

   Your customers' priorities change over time, so don't be afraid of reflecting those needs in the UI. Bring the most popular features to the fore, then move special use cases or anything needed for backwards compatibility to a tucked-away spot that won't clutter

the UI. Regularly evaluate how functions are accessed, even if there aren't features you intend to cut.

Behavioral Debt

Behavioral debt specifically refers to the behavior of the user interface. It's often a symptom of functional debt, but worth categorizing separately to afford more granular prioritization.

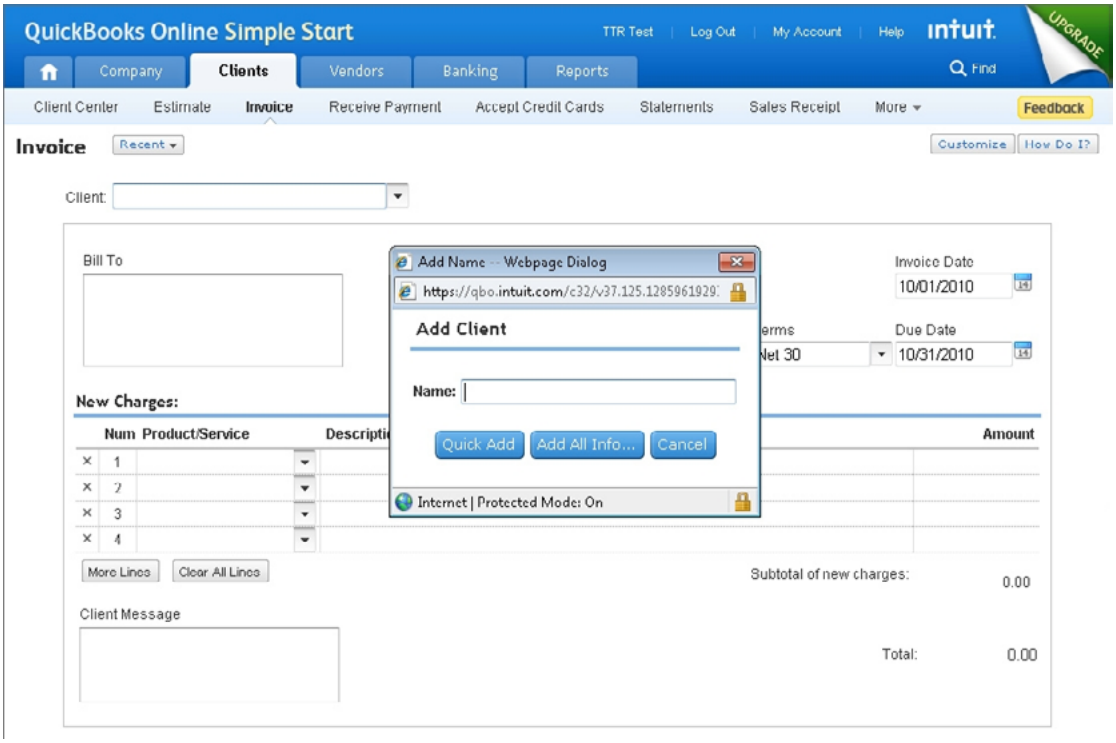


Photo credit: Amanda Linden. Quickbooks before her redesign.
A good example of additional tabs required to support more functions.

1. Tool Time

As the UI gathers more functions, more cruft is created to contain them – more tabs, more toolbars, and more settings. A user may spend so much time fiddling with widgets that she runs short on time to actually accomplish tasks with the tool.

2. **Consistency**

   It's easy for inconsistencies to sneak into the UI as a tool matures, and the effect can be much larger than you would anticipate. Inconsistencies cause confusion and erode trust.

3. **Conventions**

   A convention is formalized consistency. You create a convention when you recognize intentional consistency (or unintentional consistency that is deemed valuable after the fact) and document it as a rule or guide. Be cognizant of this while classifying and addressing UX debt, keeping your conventions up-to-date alongside your UI.

Visual Debt

Teams with poor communication between developers and designers are especially prone to visual debt.

1. **UI Chrome**

   The amount of screen real estate available for the content and important, interactive parts of the UI is frequently reduced as an application accumulates chrome. These static, non-interactive parts of the UI (the button bars, borders, title areas – essentially, anything that isn't text and can't be clicked) significantly cut away your high-value space for functionality.

2. **Iconography**

   Inconsistent iconography is arguably the worst visual debt infringement. Usually the outcome of pure laziness, someone might choose a pre-existing graphic to represent a function rather than design a custom icon. Unbeknownst to them, someone else has already used the graphic to represent a completely different function, sowing confusion across the product..

3. **Consistency**

   Consistency again? Yep. Visual consistency in color, typography, layout, and style is imperative to a quality user experience. Regardless of how well it actually works, if a UI looks broken or unkempt, users will translate that into their perception of the product.

4. **Trends**

   User interfaces are consumer products and likewise subject to trends, whether you like it or not. If your application falls too far behind, users will see it as “old” and “outdated,” regardless of how well it works. If your interface still looks skeuomorphic, expect some backlash in today’s era of Flat Design.

5. **Branding**

   There should be a collaborative process between UX and marketing; when it comes to software, your interface is your brand. When your brand changes for marketing reasons, you should consider changes to the UI in support of the brand. Conversely, when changes are made to the UI for usability reasons, those that impact branding guidelines should be discussed with marketing.

6. **Copywriting**

   This aspect of branding has implications for usability, particularly in terms of labeling and notifications. Copywriting debt covers everything from typos and grammatical errors to poorly-worded instructions and unhelpful error messages.

Documentation Debt

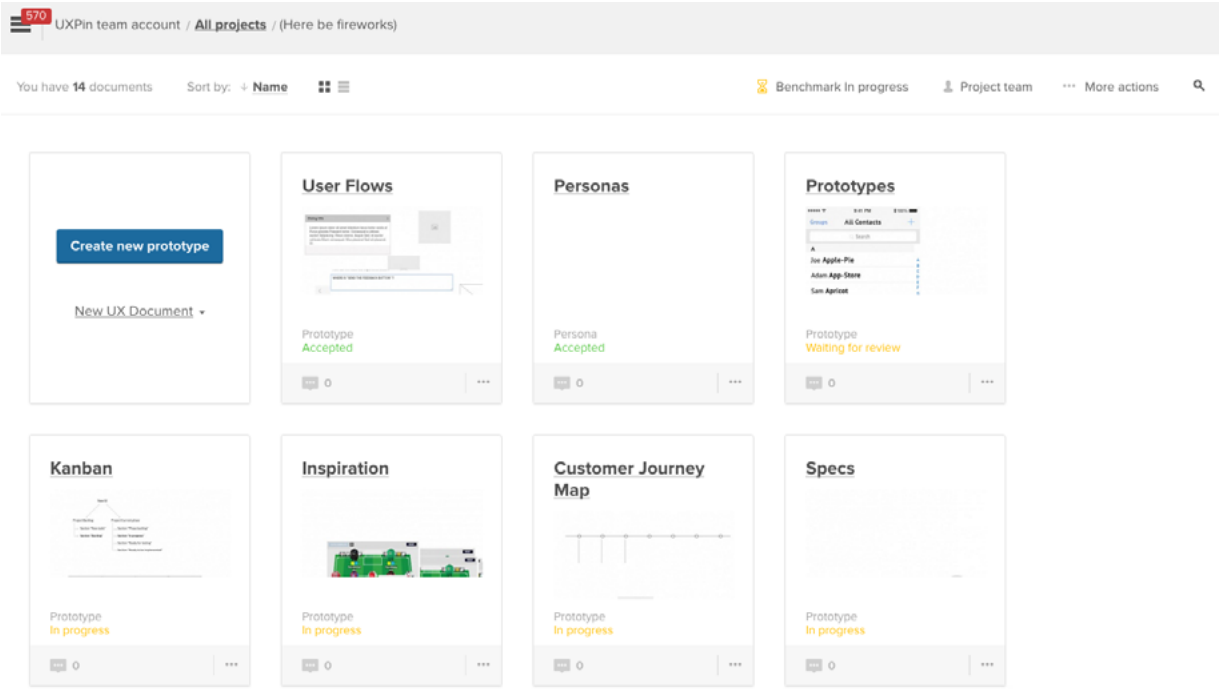
It is imperative that a team leaves behind a visual and written history of its thought processes. Otherwise, the only record of decisions is the final product.

Unfortunately, good documentation is a real struggle for many teams to create on their own; growing piles of poorly-maintained product specs can become indecipherable puzzles. Even a pure Lean UX approach doesn't always translate well to enterprise environments.

As a middle ground, collaborative design platforms (like UXPin) can alleviate much of the documentation burden. There is less risk of design decisions becoming fragmented across different people's desktops or cloud folders when you can create and house artifacts like user flows and prototypes in one central hub.

Never take documentation debt lightly – it can quickly lead to visual and behavioral debt as the project evolves. Unless teams synchronize all work around updated design artifacts, visual standards may dis-

integrate in the name of convenience. Even the best design system could be sabotaged due to behavioral inconsistencies.



Documentation and asset management in UXPin

Conclusion

Like a financial loan, UX debt should be proactively avoided and diligently reduced when possible.

Now that you've seen the common sources of UX debt, we'll dive into tactics for spotting it in your products.

Strategies for Spotting UX Debt

Knowing the source of UX debt is an important first step – and you don't need to schedule a special meeting or phase of the project to begin identifying issues. It's something you can accomplish in parallel with your normal activities.

To vigilantly detect, isolate, and track UX debt from so many sources, here are a few practices and tips that I highly encourage.

Create and Validate a UX Debt Inventory

Whether you're at the receiving end of an incoming product acquisition or coming into the team as a new hire, corralling UX debt starts with discovering what you're up against. And that means conducting an inventory.

Let me walk you through the process.

First, sit down and use the product. You want to do this yourself to highlight anything you find unintuitive or confusing. Keep notes as you go, or ask someone to write down your comments as you use the product – then switch places.

Description	Type	Impact	Severity	Priority	ROM	Responsibility	Status	Logged
Missing counts in search facets	Technical: performance	Users don't know how many results a filter will result in until they try it.	Minor	Medium	2 months	Back End Dev	Scheduled for Build 5.15.2	3/24/16
Delete icon inconsistency	Visual: iconography	Some screens use the "x" icon and others use a trash can. This may confuse users.	Minor	Low	1 week	Design	Backlog	6/7/14
Flexbox layout broken in Edge	Technical: browser support	The product doesn't display correctly in the Edge browser.	Major	High	3 weeks	Design	Scheduled for Build 5.15.1	4/1/16
Inbox implemented in GWT	Technical: outdated framework	One of three screens left in product still using GWT. Inconsistent visual design. Maintainability suffers.	Major	High	1 month	Front End Dev	Scheduled for Build 5.15.1	11/17/15
User list can't be filtered	Functional: scale	Difficult and slow to find a specific user among thousands.	Minor	Medium	2 weeks	Design, Front End Dev	Scheduled for Build 5.15.2	9/2/15

Another collaboration option entails using a spreadsheet (like this one) in your team's cloud folder while evaluating the heuristics together. As the creators Susan Rector and Kim Dunwoody suggest, review the system based on criteria in the following categories:

- Findability
- Accessibility
- Clarity of Communication
- Usefulness
- Credibility
- Learnability
- Overall Aesthetics
- Persuasive Design

You can create a solid snapshot of UX gaps by involving the whole team. Be sure to block out a manageable timespan (e.g., 1-2 months); completing the evaluation will definitely take more than a week.

Remember that while this exercise is highly informative, at the end of the day, you are not the intended user.

Now that you've conducted a UX debt inventory, it's time to validate your findings by observing and talking with actual users and subject matter experts (read more on that in Rian van der Merwe's free guide *Practical Enterprise User Research*).

This will help you better prioritize the work with product managers for the payback sprints or the backlog.

Practice Peripheral Awareness

Anytime you interact with the product, continually scan for issues that may have escaped notice. Do this in your peripheral awareness, meaning focus on the task at hand, but keep your sensors on in the background, just in case something catches your attention.

Also, be ready to jot down notes whenever you observe someone interacting with the product. Then, enter them into your issue tracker as soon as possible (before you forget the details left out of your notes).

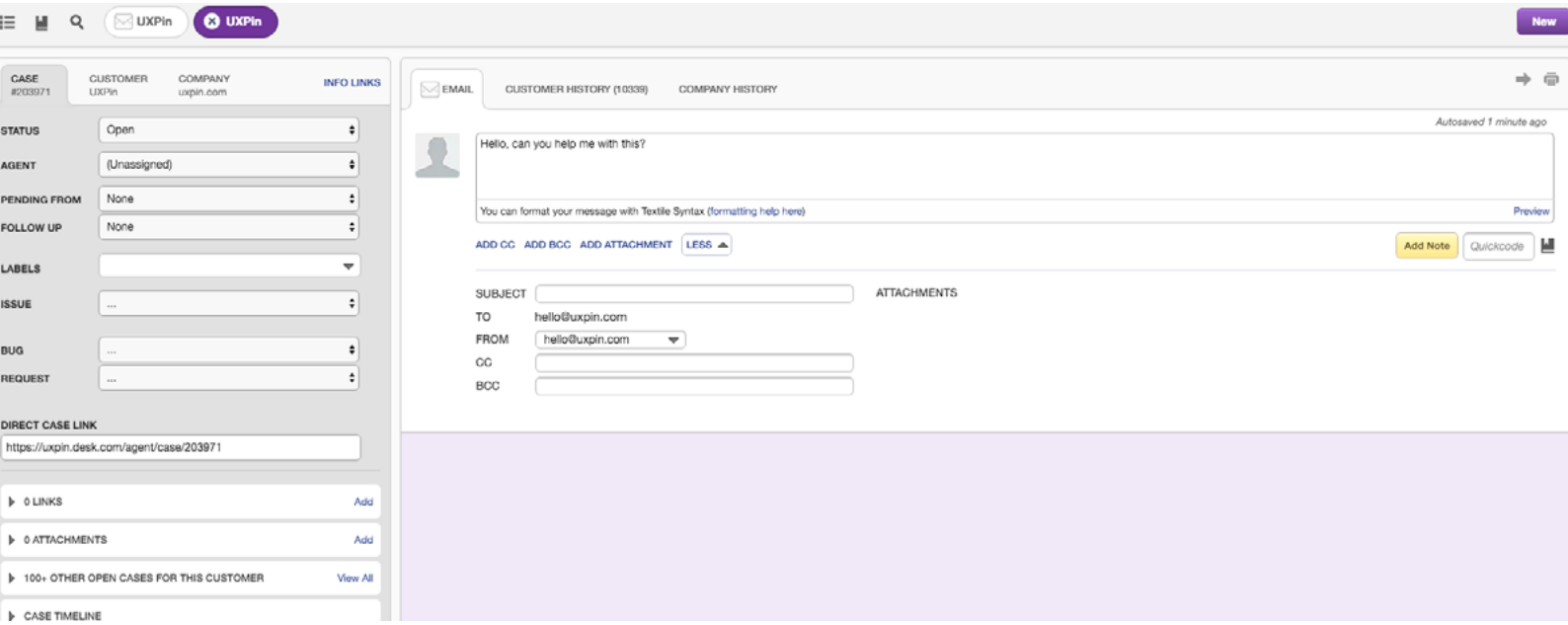


Photo credit: Issue tracking in Desk.

You should also note any bugs – both functional ones and any little details that seem out of whack. Be sure to enter them in your tracking system (like JIRA or Desk) and classify them as UX issues. Classification is important, as it enables you to find them again later, prioritize them, and get them into the schedule to be fixed.

Be Aware of Your Blind Spots

Outside of the product, think about what design practices could *create* debt.

My team faces a situation that could easily create UX debt. We're distributed, so we facilitate our critiques online.

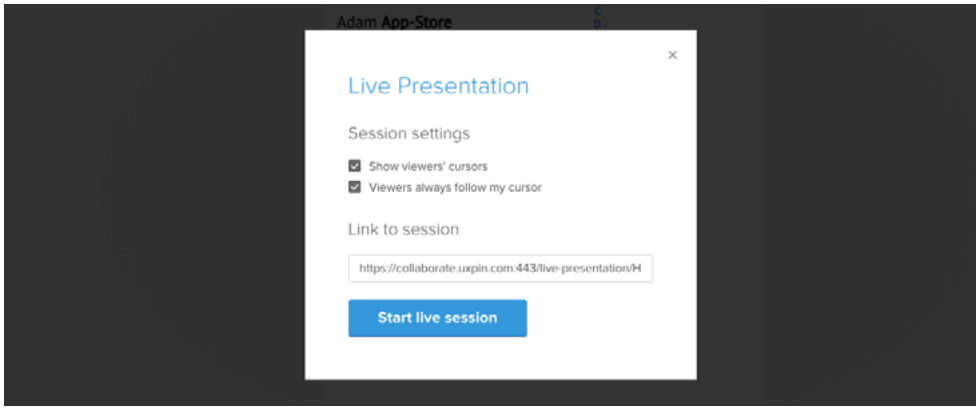
In this format, we're generally only looking at one screen at a time. I happened to look at two screens side-by-side and noticed slightly different formatting being used to represent the same type of information. It was easily noticed when comparing the two, but quite invisible when viewing them in sequence.

If we were colocated, we would all sit in the same room for critiques, and we would have screens printed and stuck on the walls. We would regularly make cross-comparisons. Being aware of this, I can now address the situation and hopefully avoid some unintentional UX debt.

Designer Pro Tip

At UXPin, we follow the 10-line rule: if a problem requires more than 10 lines of chat to resolve, we'll hold a quick Google Hang-out to talk through it live. If it relates to a design problem, we'll share a link of our UXPin project and talk inside the platform.

Ben Gremillion, Content Designer at UXPin



Schedule Regular Time With Users

In his article Fast Path to a Great UX - Increased Exposure Hours, Jared Spool explains that the more time your team spends with users, the better your end product. He also says that companies that intentionally include everyone – not just the UX team, but developers, managers, etc. – have better results.

When everyone sees users struggling with the product first-hand, people argue less about what needs to be done. UX debt is the responsibility and concern of everyone working on the product. This even extends to your customer.

Observing your users is also the most effective way to identify UX debt. Instead of only conducting user research in a frontloaded Sprint 0, consider spreading out some of those user exposure hours across all your sprints.

For example, even just 2-3 hours of user interviews and/or usability testing every 3 weeks can help the entire team identify and prioritize critical gaps in the product.

Conclusion

Once you've inventoried all of your UX debt, you may feel a bit overwhelmed. "How in the world are we going to get rid of all this debt?!" This is a perfectly normal reaction. It won't seem nearly as bad once you've planned an approach to deal with it.

Don't panic. Prioritize.

Addressing UX Debt

You know you have debt. You've identified it. You've classified it. Now what are you going to do about it?

If you are able to walk through your debt inventory and begin pencilling in solutions and rough order of magnitude estimates, chances are your debt is manageable.

That doesn't mean you immediately know all the answers or will have it all fixed by the next release. If you can create a plan that doesn't give product management a heart attack, you'll eventually be able to eliminate your debt.

Roll up your sleeves. We can do this.

Step 1: Prioritization

Your debt must be prioritized so that it can be addressed in phases.

1. **Severity & Impact**
   How big is the issue? Is it keeping users from doing their work? Is it creating a safety or security risk? Is it causing a potential customer to turn around and look to a competitor's product?

   These are all relatively severe effects that would suggest the issue is a high priority.

   But don't just consider the negatives. How big an improvement will the user see? Will the improvement save hours of time in the course of a month? Will it reduce errors? If so, it may be worthy of high prioritization, even if it isn't currently considered to be a big problem.

   If you have a lot of products, you may consider employing a UX Maturity Model as the basis for prioritization.

2. **Estimated Time to Address**
   How long is it going to take to fix?

   If all you have to do is tweak the CSS, you might slip it into the next build. On the other hand, if it's going to require a significant amount of development or will have to be thoroughly regression

tested, it may make sense to hold off until it can be resolved with other issues requiring similar treatment.

3. Responsible Party

Who will be tasked with addressing the issue?

If it falls primarily on the UX team, and they currently have a light workload, it may be given a high priority. If it requires the attention of a specific developer who is already assigned to other high-priority work, then it will have to wait.

Designer Pro Tip

Prioritization is a nightmare if requests aren't consolidated in a single platform. For us, all feedback on features or bugs flows into ProdPad. From there, product managers can assign priorities to fixes or features, breaking them down further into user stories as part of a larger roadmap.

Marcin Treder, CEO and Co-founder at UXPin

Step 2: Schedule

After prioritizing your debt, the next step is to work with product management to get it into your release schedule.

Agile is so popular these days that it seems like any process that isn't Agile is labeled "waterfall." I find that to be a bit dismissive. There are degrees of being Agile, and you can have an effective, iterative process that doesn't involve stories, scrums, and sprints.

For our purposes, however, I'll address all non-Agile processes at once. Then I'll make suggestions for Agile teams.

1. **Not Agile**

   Your work is likely planned based on a release cycle. Your organization decides what will go into the next release based on criteria such as how long the development effort will be; how badly a feature is needed by customers; what will sell; what bugs exist and how bad they are; and so forth.

   I recommend handling UX debt issues as bugs. The real benefit of this approach is that debt items can be entered and tracked using the same tools and business processes as bugs. This will ensure that they get reviewed and treated equally. A representative from the UX team on the issue review board should prioritize items, ensuring that usability issues get the full weight they deserve.

Ideally, a representative from the UX team will also work closely with product management when releases are scheduled.

When a particular part of the application is being scheduled for work, check it for UX debt. Would it add much effort to address the debt at the same time? Often, there will be savings simply because the code is already being updated by developers. Even if it's a low-priority item, take advantage of the opportunity to pay down some debt.

2. **Agile**

   A company that employs a healthy Agile process shouldn't have any problem prioritizing debt with other types of work, assigning it story points, and fitting it into sprints.

   Find the rhythm

   In my own experience, however, Agile has been embraced as a way to get more work done faster, rather than as a method of iterative improvement.

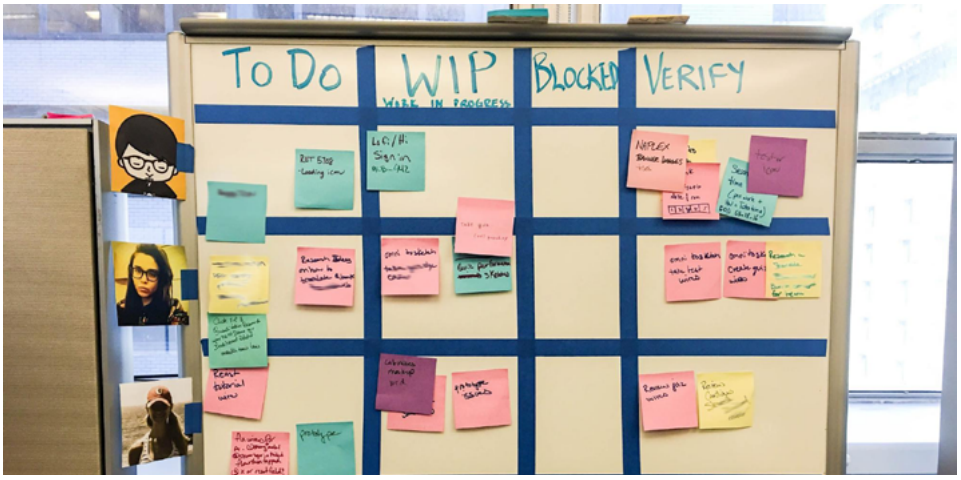


Photo credit: Laura Kershaw. Design process at Kaplan Test Prep.

In such situations, you may have a harder time scheduling UX debt because (as management sees it) there's not enough time to fit in everything they aim to wrap up, so there certainly isn't time for all those trivial corrections you're asking for.

If you find yourself in such an environment, your goal should be to find a rhythm for addressing debt.

Propose a certain number of story points per sprint (or every other sprint). Or, perhaps a sprint could be devoted to addressing debt at some regular interval (payback sprint). This should be done at least until the backlog of historic debt – your debt inventory – has been handled. Then it should become easier to keep up with new debt that crops up without that regular schedule.

Try a "Cheese Day"

For even tighter schedules, consider holding a Cheese Day to knock out as much debt as possible. Management is almost always receptive to a one-day workshop every 60 days where you knock many items off the debt list.

The following procedure suggested by Roy Man is both realistic and effective:

1. About 2-4 weeks before Cheese Day, create the project in your app of choice (Asana, Trello, Basecamp, etc) and encourage everyone from customer support to developers and designers to briefly describe product annoyances.

2. Prioritize the cheese list based on the advice in the below chart. Separate the “Quick Wins” from the “Nice-to-Haves”.

3. Schedule 6-8 hours for the Cheese Day, inform everyone of the date, then dive right into the “Quick Wins”. Everyone will feel productive, and you’ll have progress to show management at the end.

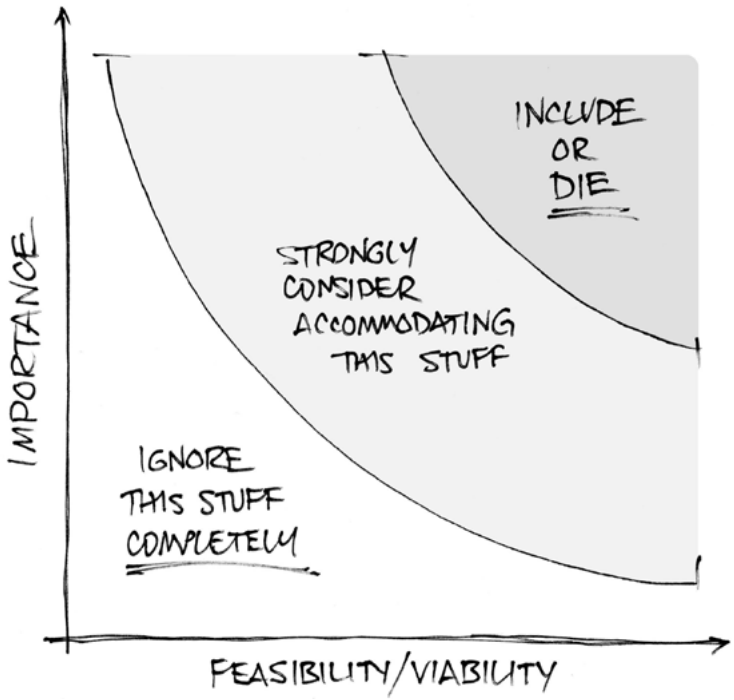


Photo credit: Joe Natoli

Conclusion

Most importantly, we address UX debt through collaboration.

UX debt should be understood as the responsibility of the entire organization – not just the UX group. It takes a good working relationship with your entire team to ensure that UX debt is given the attention it deserves.

Of course, the best way to eliminate UX debt is avoiding it in the first place. In the next chapter, we'll explain some proactive tactics for minimizing UX debt.

Avoiding UX Debt

Wherever you may be in the debt elimination process, don't get so caught up in your present situation that you blind yourself to new debt. It's always out there, trying to work its way into your products.

The first key to avoiding UX debt is to understand where it comes from, which helps you spot trouble and guard against it. Here are some practices that will significantly lower your risk of acquiring unintentional debt.

Usability Research and Testing

To protect against UX debt, it is imperative to understand the intricacies of the work we're supporting. The more you know about users, the less debt you'll likely accrue. Unfortunately, it is common for enterprise UX teams to work without the benefit of user research.

You'll find countless reasons to blame it on, ranging from costs and logistical infeasibilities to billing issues and customer stubbornness. The lack of testing is also a problem; how many times have you heard the phrase, "That's a training issue"? Time to market, short deadlines, and a reliance on "customer acceptance criteria" can all mean zero usability testing. Some company cultures even believe that Quality Assurance is all that is necessary.

If you're struggling to get buy-in for user research, try following Rian van der Merwe's advice:

- Frame any user research discussion around revenue (e.g. $1 in design saves $6 in development and $100 after launch).
- Use case studies to support your point, like the $300M button story.
- Present a concise user research plan that shows a feasible budget and timeline. In the 1-page plan, give a brief background, list the testing methods and schedule, describe the research goals, and desired insights (e.g. "Product Marketing recommendations to overcome barriers to adoption).

When it comes to finding end-users for research and testing, try to form a relationship with the people who need to implement the software. Otherwise, going through the buyer can result in tons of red tape.

Designer Pro Tip

We've found that holding 2 hours of user interviews or usability tests a week helps us maintain sprint velocity without losing quality. For simplicity, we'll run remote moderated tests in UXPin or hold user interviews over Zoom.

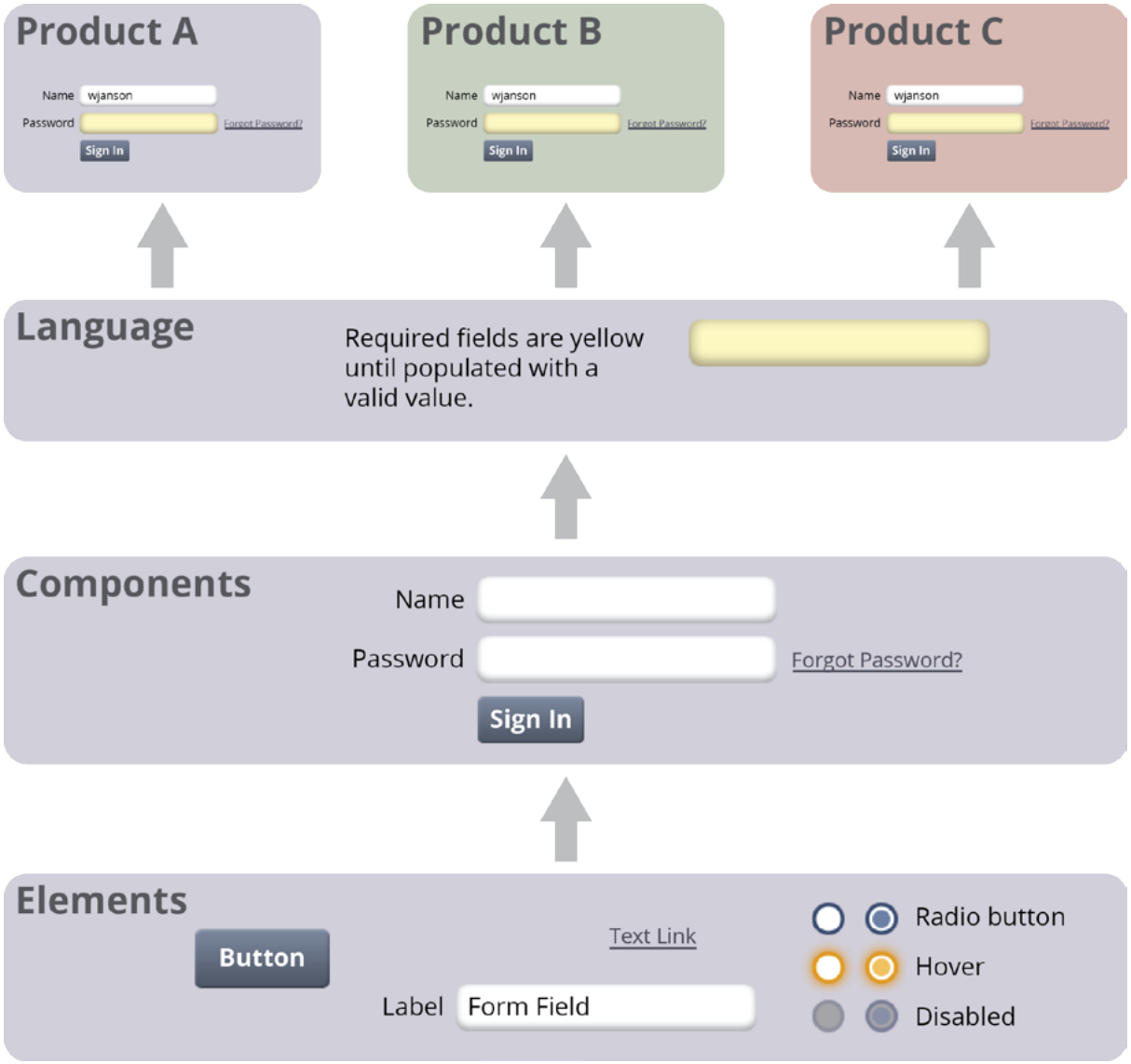
Marcin Treder, CEO and Co-founder at UXPin

Modularity

A well designed user interface, by its very nature, is consistent.

A button is represented in a very specific way; there may be variations, but the user will always recognize it as a button. By its very nature, a user interface should be modular, with reusable components.

Strive for a design system that is both modular in its design language and in code.



In Design

Modular UI design works a lot like modular homes:

- A home builder doesn't design new doors and windows for each house it constructs. They standardize on a set of basic, pre-manufactured parts that they can then arrange in many different layouts. Similarly, we can identify an array of common elements used repeatedly throughout our products in varying contexts. For

example, a sign-in form is composed of labeled name and password fields, a submit button, and typically a “forgot password” link.

- As those basic elements are assembled into components, a design language begins to form. Common UI patterns emerge and can be documented in a pattern library or style guide. You’ll want to explain the use cases and rationale behind the patterns, such as when to employ a card-based layout versus a list.
- Small components may be combined into larger ones, eventually forming screens. At this point, you have achieved an economy of scale in which the effort spent on the detailed design of one microinteraction is magnified many times over as the resulting value is realized repeatedly throughout a suite of applications.

To facilitate a universal understanding of your modular design among your team, start by identifying reusable components together. As you define the principles and patterns, make sure you employ a common vocabulary. As Alla Kholmatova reminds us, the name of your patterns will affect the perception of their function and reuse (e.g. “homepage header” is more restrictive than just “header”).

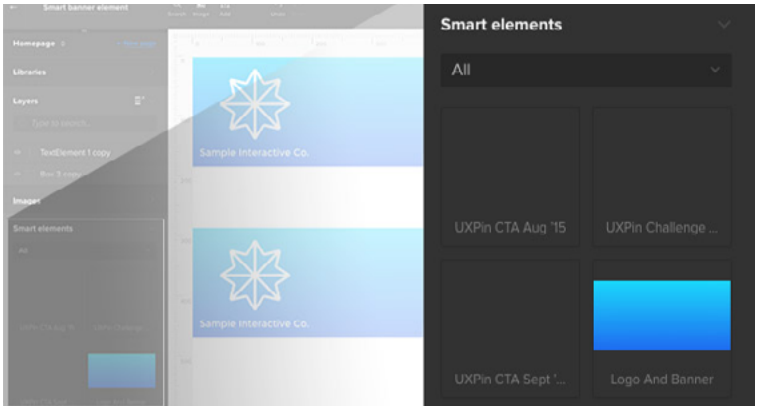
You can even introduce a bit of participatory design as you define your language and patterns. Again, as Kholmatova recommends, consider showing the patterns to users to get their feedback on perceived functionality.

Designer Pro Tip

Consistent, modular design doesn't need to be hard. If you're using UXPin, see how to create a modular design system below.

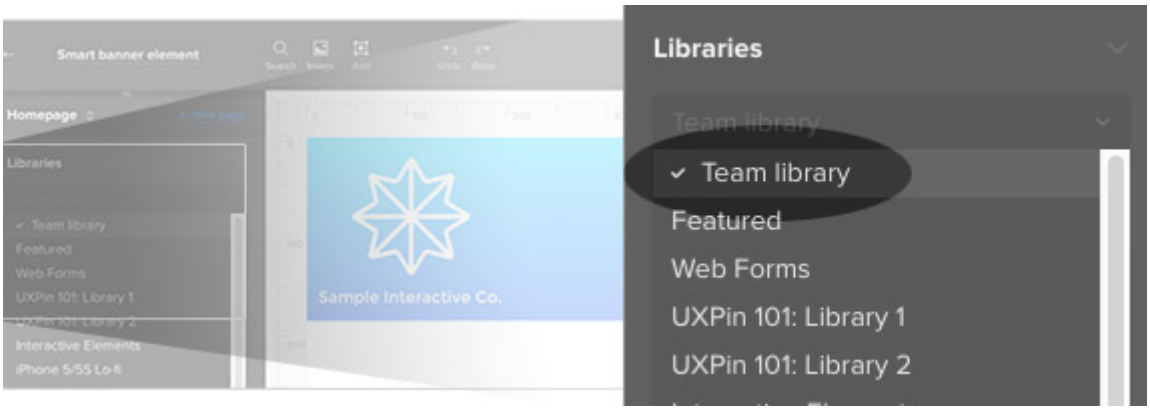
***Smart elements** are sets of elements that, when copied, are always identical. Change one copy and you change them all. These are great for project-wide design components like navigation bars, headers and logos. For design consistency in pattern libraries, smart elements are your best friend.*

You'll find smart elements in the lower left-hand corner of the UXPin Editor.



If smart elements are too restrictive, **custom libraries** *are sets of elements that you can tweak as needed. A custom library contains*

your own pre-built items, like buttons and banners, that you can add to pages – then change to fit their context. Buttons, for example, may require consistent typography but different messaging.



When you need to base a design component on a template, but tweak it for unique cases, custom libraries are the best choice.

Ben Gremillion, Content Designer at UXPin

In Code

While modular design helps reduce design inconsistency (and therefore UX debt), modular code helps streamline implementation and technical maintenance. With modular code, the team doesn't need to change 100 different instances of a button for a small HTML tweak.

Instead, the change is made once within a single chunk of code, and every button in the application automatically exhibits the change.

Here are a few useful approaches:

- Object-Oriented CSS (OOCSS) is a modular, front-end architecture that keeps the structure of your UI separate from its appearance and separates the container from the content. In so doing, it makes your styles reusable and more maintainable. For more information on this approach, I invite you to check out my book, Bridging UX and Web Development.

- SMACSS stands for Scalable and Modular Architecture for CSS, a creation of Jonathan Snook. It's similar in spirit to OOCSS, organizing styles into five categories: Base, Layout, Module, State, and Theme.

- BEM stands for Block Element Modifier. It's basically a class naming convention that follows the OOCSS approach.

- Atomic Design is a methodology for creating design systems. Created by Brad Frost, the approach has you systematically identify the basic elements of your UI – the atoms – which then combine to form molecules, and finally become organisms. It's a perfect metaphor to understand the modular structure of your product.

Ultimately, you're creating a unified design system in aesthetics, interactions, and code – which is the most effective method for avoiding UX debt. It enforces the design, making it easier to build a new screen following the rules than it would be to do it any other way. And, if any debt does creep in, it is easier to address. Fix it once to fix it everywhere.

Designer Pro Tip

Once you have a common design language that breaks down to patterns and elements, everyone does less redundant work. Designers can focus on solving the large business problems, and developers feel empowered to use vetted components instead of asking designers to verify everything.

Daniel Castro, Design Director at Sumo Logic

Be the champion of modularity in your organization.

Smart Documentation

People complain that design documentation takes too long to produce and maintain, much less read – but documentation is not bad. Bad documentation is bad.

Smart documentation helps drive good decisions and avoid UX debt:

- It helps developers avoid implementing a first-thing-that-comes-to-mind decision.

- When accurate, it is a trustworthy source for user guides, help documentation, and training material.
- The act of documenting causes you to think in more detail, and from a different perspective.
- It increases the probability that mistakes made during implementation will be caught and corrected before the product ships.
- It can become a reference that guides future design.

Prototypes are a great way to demonstrate behavior that is difficult or inefficient to describe in a static document. Employ them liberally as tests and exemplars. That said, they should supplement good documentation – not replace it entirely.

Enterprise software is complex, requiring multiple servers, databases, and integration with external, third-party systems. Strict security measures may be in place. You can't rely purely on the live system as a reference. Smart documentation is much more convenient.

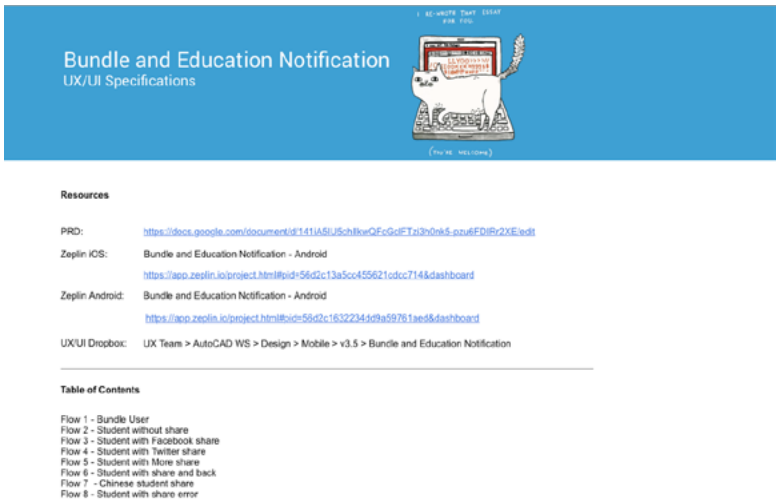


Photo credit: AutoCAD 360 Product Requirements Document.

Autodesk's AutoCAD 360 team, for example, creates their product requirements document as an information hub. They describe technical and UX guidelines and include links to Zeplin and UXPin to illustrate the requirements. As the project progresses, the team continues to update the document to reflect new requirements and constraints.

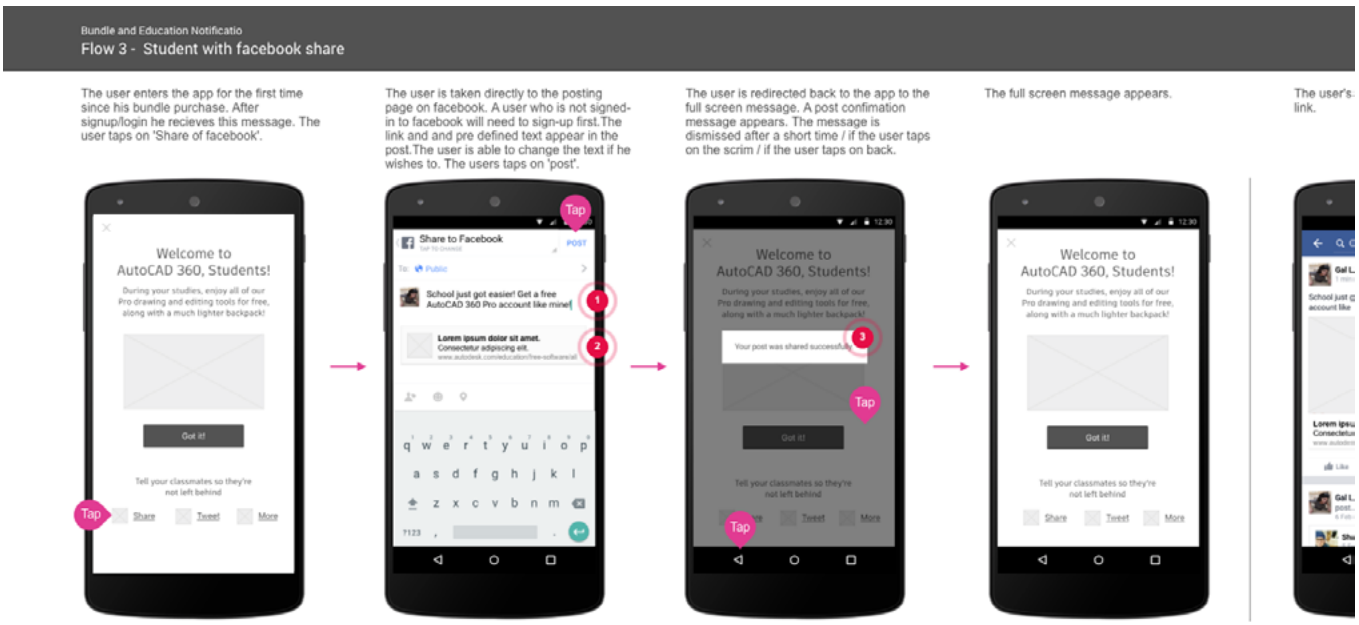


Photo credit: AutoCAD 360 Product Requirements Document created in UXPin.

Too often, documentation is done at the end of the process. Designers wait until the design is "done," and then they document it. This is why such a negative view of documentation persists. It is seen as the chore that must be done before moving on to the real design work.

Documentation isn't a separate task; it should be happening throughout the process, and indistinguishable from designing. Good documentation is a force field against UX debt. It acts as a filter, catching anything that shouldn't make it into the product.

Be the champion of good documentation in your organization.

Conclusion

UX debt is a serious matter, and it often seems insurmountable.

I hope this guide has given you perspective and confidence in the knowledge that you can formulate an approach for identifying, prioritizing, and eventually eliminating the debt your products have accumulated.

Even better, you now have a strategy for protecting your organization against future debt. Lead your team to a debt-free future in which you can spend more time enhancing your products with the latest advancements in technology and addressing the needs of your users, rather than making compromises, beholden to legacy deficiencies.

I'll offer you one last word of advice: Don't be disenchanted by the challenges before you. Take pride in your accomplishments. Celebrate your progress, and promote the vision you are working towards.

UX Efficiency Case Study

Speeding Up Design Reviews By 300%

The Challenge

Based in the Bay Area with 250+ employees and $161 million in venture capital funding, Sumo Logic serves some of the top enterprises in the world. The company's analytics platform visualizes more than 100 petabytes of data per day, helping businesses harness the power of machine data to streamline operations.

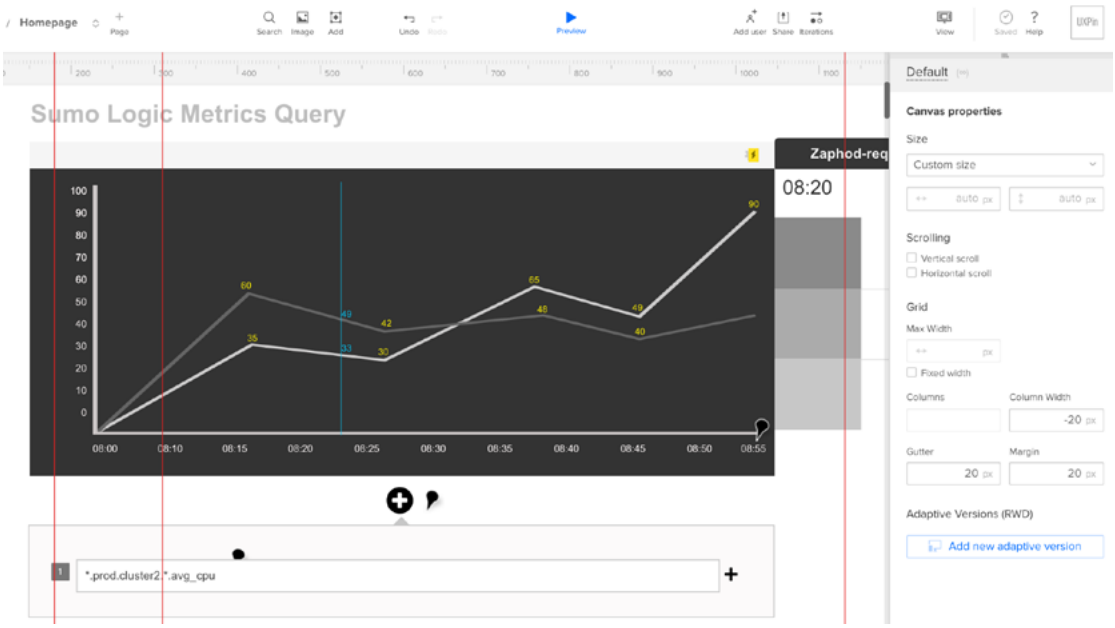
To make the data actionable for customers, Sumo Logic invested heavily in its own internal UX team to bring a "consumer-grade" experience to the enterprise. In 2015, they hired their first UX team comprised of design leaders, interaction designers, visual designers, and UX architects.

The company had been using Axure for wireframing but Design Director Daniel Castro quickly found that the solution did not allow for easy design modification and did not encourage collaboration.

Tired of sharing PDFs back and forth, the company needed a collaborative UX platform that could scale along with their teams and processes.

The Solution

For the first six months, Design Director Daniel Castro and his team spent much of their time holding happy hours and offering show-and-tells of great UX design. The next challenge was to create a collaborative environment where UX designers could work closely with both technical and non-technical staff to revolutionize the Sumo Logic experience.



Sumo Logic prototype created in UXPin for their Unified Logs Metrics product

In a culture already used to collaborative tools such as Slack, this slow process of emailing thoughts on static designs was stifling. Castro had begun using UXPin at his previous job, and knew it would offer his Sumo Logic team the collaboration tools they needed.

"We are constantly collaborating with engineering and product managers and it used to take a significant amount of time to work together going back and forth," Castro said. "UXPin allows us to easily show the flow and main components of our projects. We can share a link and everyone can communicate with our key stakeholders, expanding on each other's comments and allowing us to manage feedback contextually without redundancy. It's like a visual version of our thought process. We can even make comments on a pixel level. This has made our review process three times as fast."

"UXPin has played a vital role in creating a design-oriented culture at Sumo Logic," Castro added. "The team is great to work with, and I'm excited to see what we can do next."

The Results

- Design modification is **quick and simple** with UXPin, instead of the limiting modifications possible with Axure wireframing.
- Design reviews are **three times as fast and now contextual** using UXPin to collaborate instead of emailing static PDFs.
- UXPin is “like gold” when trying to get approval from stakeholders on projects, **halving the effort** needed to communicate with stakeholders.

Want UXPin to help your team? Contact insidesales@uxpin.com to learn more about our enterprise features.

UXPin

ENTERPRISE

Create and Collaborate.
Translate requirements into product features that resonate with customers.

Simplify your Process.
Centralize projects and people into one clear workflow.

Empower your Team.
Guide creativity with a common design language.

Take a look