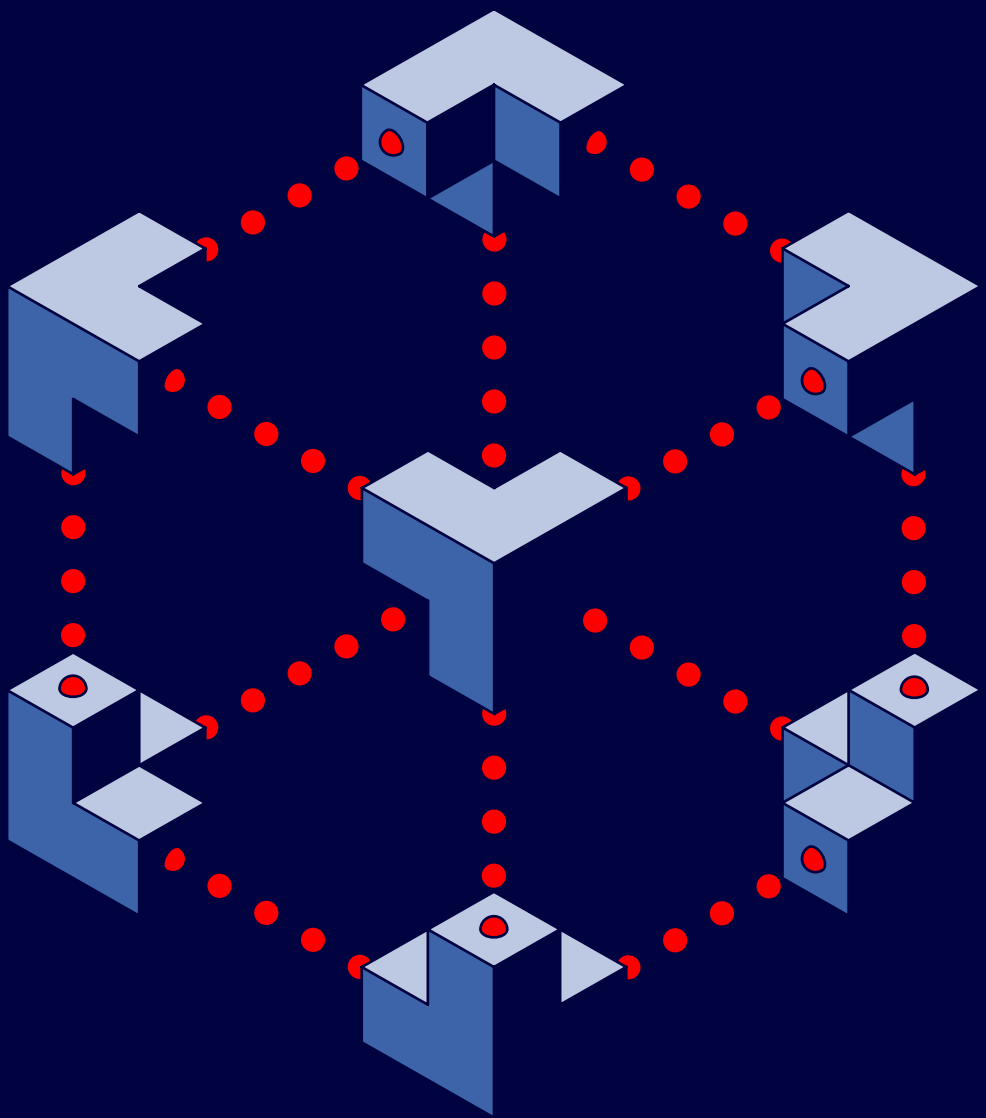
UXPin

Product Development for Distributed Teams

Processes for UX & Product Leaders



UXPin

Product Development for Distributed Teams

Processes for UX & Product Leaders

Index

Authors

Jerry Cao

Jerry Cao is a UX Content Strategist at UXPin. In his spare time he enjoys playing electric guitar, watching foreign horror films, and expanding his knowledge of random facts. Follow him on Twitter.

Sunita Reddy

Sunita is the VP of Product at UXPin, where she leads the UX and product management teams. Her nearly 20 years of product experience includes VP Product roles at Accellion, TBD Networks, and engineering roles at Oracle. She also holds 4 U.S. patents for mobile enterprise products. She holds an M.S. in Electrical Engineering from the University of Maryland.

A Few Quick Words

Distributed collaboration isn't the easiest endeavor. We've known since the 1970s that communication decreases as distance increases.

Research also shows that distributed teams are very common: of the 275 designers who responded to a Mural poll, 2/3 said they deal with distributed situations on 80% or more of their projects.

The challenge, of course, is that product teams operate in the grey zone between ideation, creation, and validation. Our work becomes naturally collaborative as we form feedback into insights. So, it's not surprising that the same poll also shows that about 50% of designers feel their quality of work suffers when working distributedly.

As Jim Kalbach of Mural explains in his excellent article, we need to adapt our processes for three common product team structures:

- **Split Teams** – Two teams, each co-located, but separated physically. While communication is balanced between both teams, an "us" and "them" dynamic can arise.

- **Mixed Teams** – The main group is co-located while several teammates are distributed offsite. Mixed teams may run into a “here” and “there” dynamic: the on-site team has the advantage of in-person communication. As a result, distributed participants may feel like “employees in exile”.
- **All Distributed Team** – No central location exists. While you don’t risk a split culture, issues with asynchronous communication are magnified. Companies like Buffer and Upworthy are among the 125 most popular fully distributed companies.

Whether your team is fully distributed or somewhere in between, you can close the distance with smarter processes, lightweight documentation, and overcommunication.

In this guide, we’ll walk through some best practices we’ve learned as a company operating in Poland and the US, as well as techniques learned from our customers and friends in the industry.

Sunita Reddy, VP of Product at UXPin

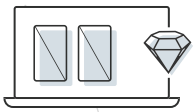
Design Systems in UXPin

UXPin

One platform for consistent design and development.

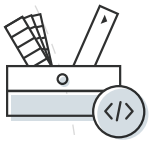
PayPal Sapient adidas HBO Microsoft

DESIGN SYSTEMS



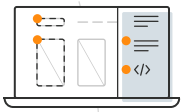
Design Language

Sync Sketch with UXPin for a consistent design language: fonts, colors, icons, assets, and more.



UI Patterns

Scale designs consistently with Symbols and interactive components.



Automated Documentation

Documentation syncs everywhere and travels with library elements.

Modular design and development

Scale quickly with design system libraries.

One source of truth for everyone

Close your knowledge gaps. Formalize your design and code conventions.

Painless documentation and developer handoff

Eliminate busywork. Generate style guides, specs, and documentation.

Tracy Dendy
HBO

> My productivity and developer productivity have both increased. They love that they can collaborate and move quickly to a powerful experience.

To book a demo, call +1 (855) 223-9114 or email us at sales@uxpin.com

Use a Scalable Team Model

The right team structure creates a strong foundation for the rest of your process. It should be one of your first orders of business.

Popularized by Spotify across 30+ teams and multiple locations, the squads and chapter structure seeks that perfect balance between autonomy and collaboration. While we started as a hub-and-spoke model described by Jeff Gothelf, we've since evolved into more of a Spotify model in the past year. In our experience, it is one of the most scalable autonomous team models.

Here's how the model works.

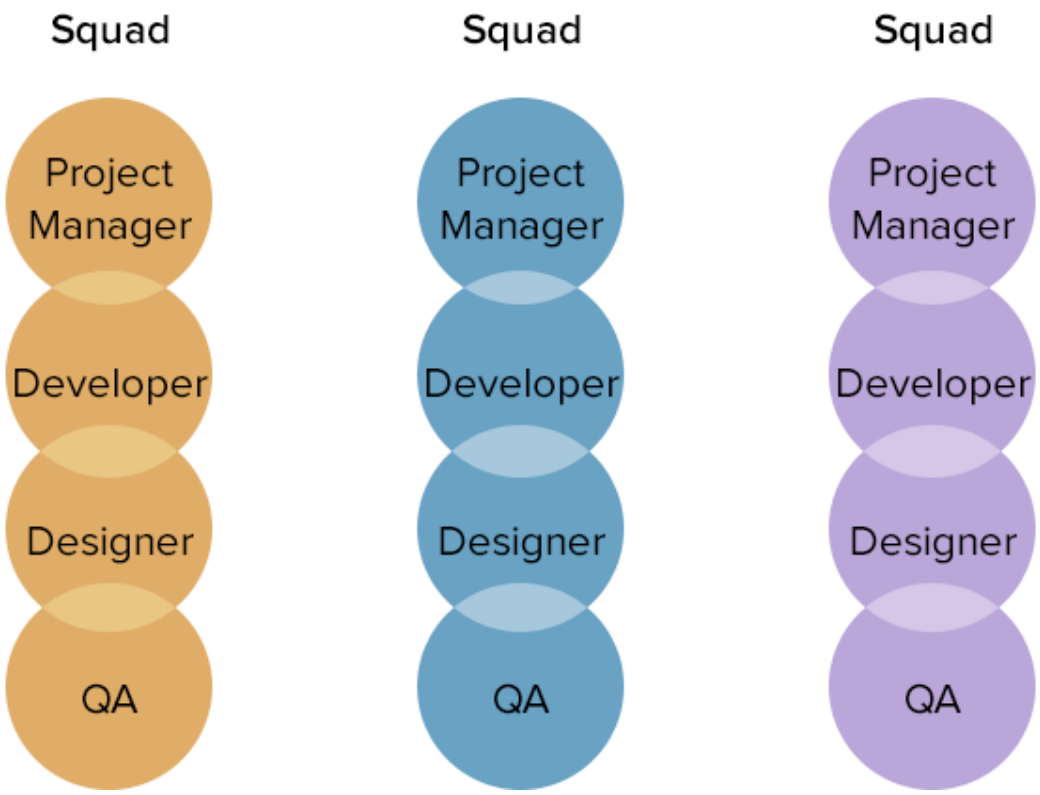
Squads

Squads are cross-disciplinary teams of 5-8 members focused on one product area. Their small size ensures the greatest autonomy, which improves efficiency for daily decision-making.

Squads include only the core product disciplines:

- Product management
- Development
- UX, visual, UI design
- QA

For example, one squad focuses on the dashboard while another focuses on onboarding. Members of each squad are located in both Polish and U.S. offices.

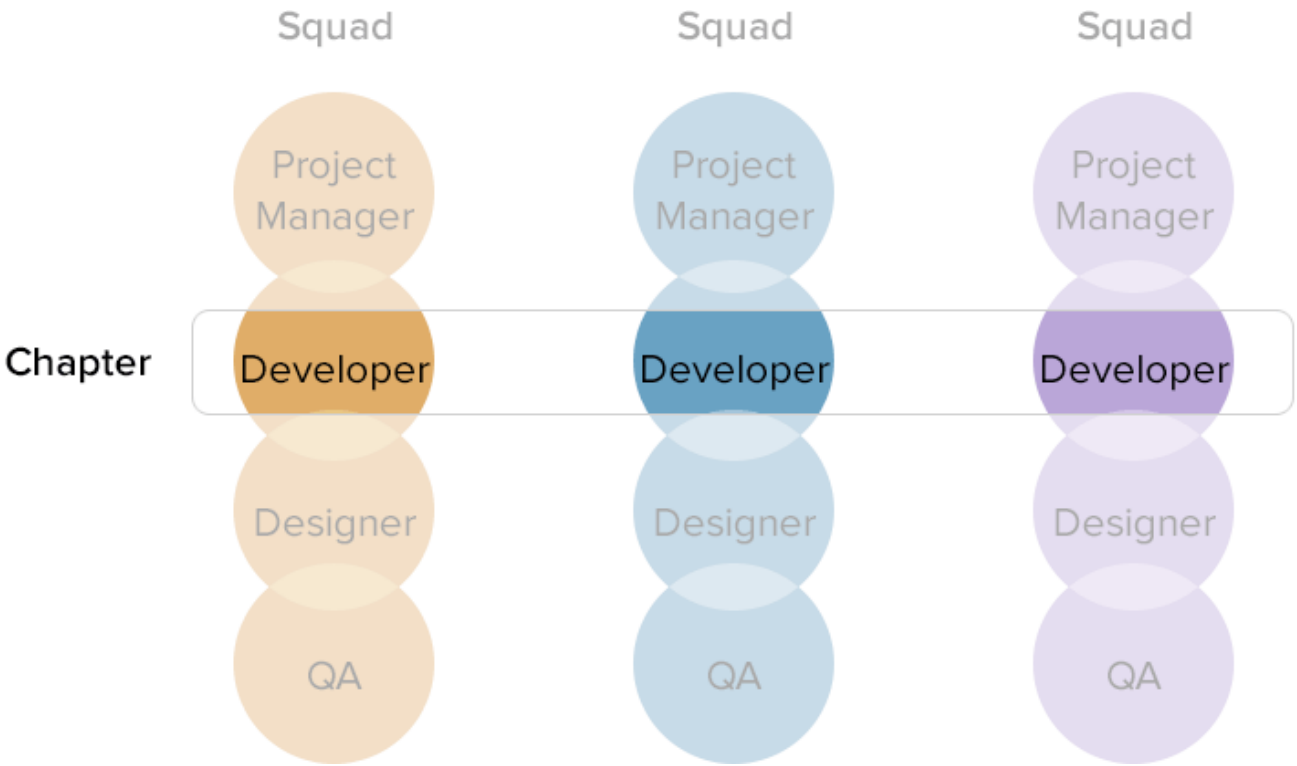


While squads can operate on different schedules, we've standardized around 2-week sprints and monthly releases to simplify project management. As the above graphic shows, squads can easily turn into silos. To spread knowledge and avoid duplicate work, the same roles across squads will form into chapters.

Chapters

Chapters are single-discipline teams, e.g. your standard “UX team” or “development team” lead by a manager or director.

The goal of a chapter is career development and improvement of best practices. While each squad might hold a daily 15-minute standup, each chapter might only need a weekly 30-minute session to discuss learnings across select projects. Over time, each chapter improves its craft and processes.



Since we’re a startup with just one product, we haven’t yet grown beyond the chapter. However, enterprises with multiple product suites can easily grow into larger tribes.

Tribes

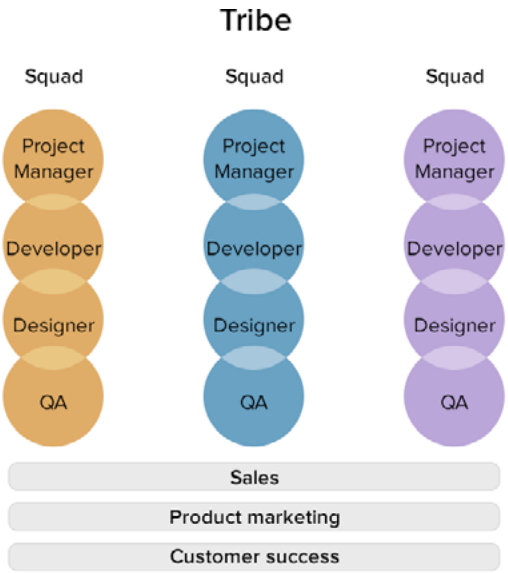
Multiple chapters form a tribe, which is equivalent to a line of business in a large enterprise.

For example, a file hosting enterprise may use the following structure:

- Tribe 1 focuses on the virtualization product (web and mobile)
- Tribe 2 focuses on the hosting product (web, on-premise, mobile)
- Tribe 3 focuses purely on back-end infrastructure (web, on-premise, mobile)

To improve autonomy, don't separate tribes based on platform. Doing so cuts the product in half and increase dependencies for even small feature improvements.

At the tribe level, you may also involve outside teams like sales, marketing, and customer support. To minimize bureaucracy, Spotify recommends that each tribe doesn't exceed 100 people.



To learn more about how different companies adopt the model, we recommend these resources:

- Scaling Agile at Spotify
- Product Team Evolution at Buffer
- How Hudl's Product Team Works

Create Your Design System Early

To scale your team and processes long-term, you need one source of truth for design standards, code standards, and assets for both disciplines. That's when a design system comes into play.

The earlier you have a design system, the less design inconsistency and technical debt you'll accrue long-term. Otherwise, each new designer who joins your team will just add their own inconsistencies to the product.

For example, our product teams in Poland and the U.S. had been operating without a design system for over 5 years. When we finally started building our own in early 2017, we uncovered plenty of inconsistencies and redundancies – for instance, 116 different colors in our palette.

We learned this lesson the hard way, but you certainly don't need to.

Enter the Design System

A design system is more than just a style guide or pattern library — it's the blueprint for product development.

All the design principles, visual assets, and patterns are thoroughly documented. All code references are included for each piece of design. As a result, design can scale right alongside development.

Popular examples of design systems include Salesforce Lightning, HPE Grommet, and SAP Fiori.

Instead of a developer or designer in another office asking about the correct pattern to use, all the elements are easily accessible. With lego-like components available, you ensure the team implements proven design solutions. Iterations become more efficient as everyone since everyone speaks the same language.

As described in the *Eliminate UX Debt* e-book, component-based design works a lot like modular homes:

- A home builder doesn't design new doors and windows for each new house. They standardize on a set of pre-manufactured parts that they rearrange as needed. Similarly, we identify an array of common elements that repeat throughout our products. 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.

How to Build a Design System

Here's the 5-step process we recommend for creating your design system.

1. **Create your interface inventory**

   To quickly identify current inconsistencies, start by creating an inventory of everything in your product UI: color palette, text styles, and UI patterns. The inventory is your strongest tool for selling the need for a design system.

   1. Review the interface and code and list all the colors and text-styles you can find.
   2. Take screenshots of UI patterns or copy patterns from your projects. Place all the screenshots in one place.

3. Categorize your patterns by their purpose (e.g. buttons, form-fields, navigation etc).

4. Mark inconsistencies between the patterns and create a presentation for your team.

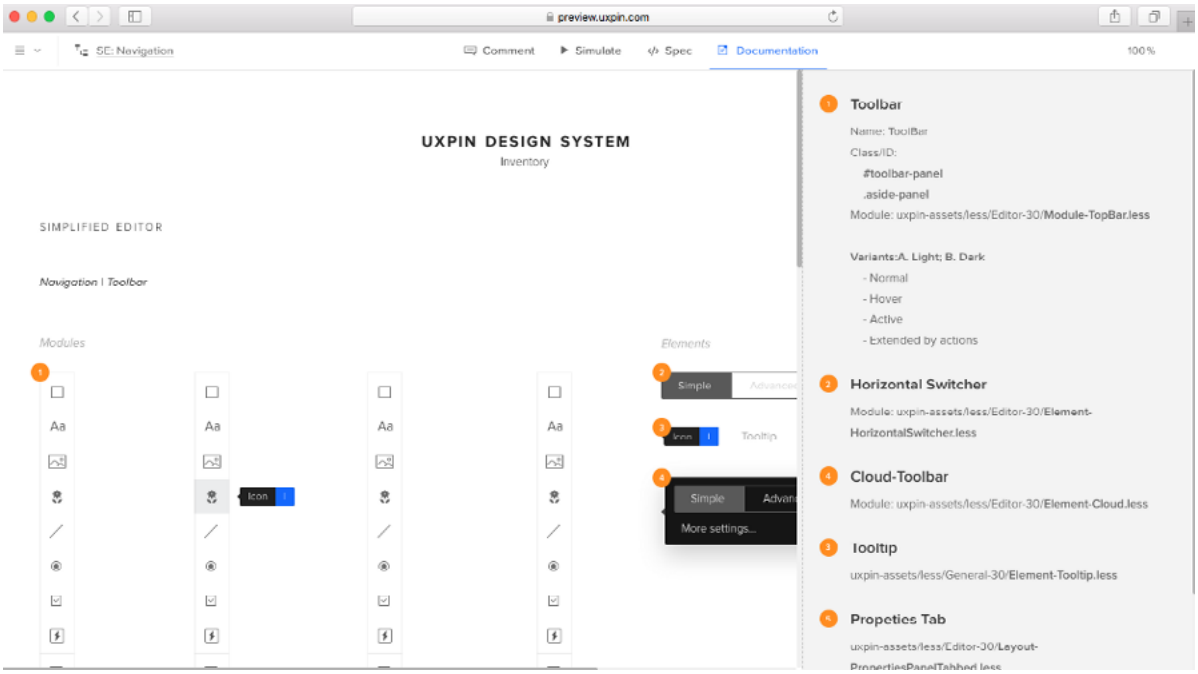


Photo credit: Creating a design system in UXPin

2. Get buy-in from the team

As you present the inconsistencies reflected in your interface inventory, emphasize the ROI of the design system.

Engineers respond well to the source control, improved modularity, and increased autonomy. Business stakeholders respond well to the faster time to market, increased product value, and less resources wasted.

3. Define your design principles

Before you build your new system, create a set of general principles for a coherent experience. What universal values should designers to keep in mind?

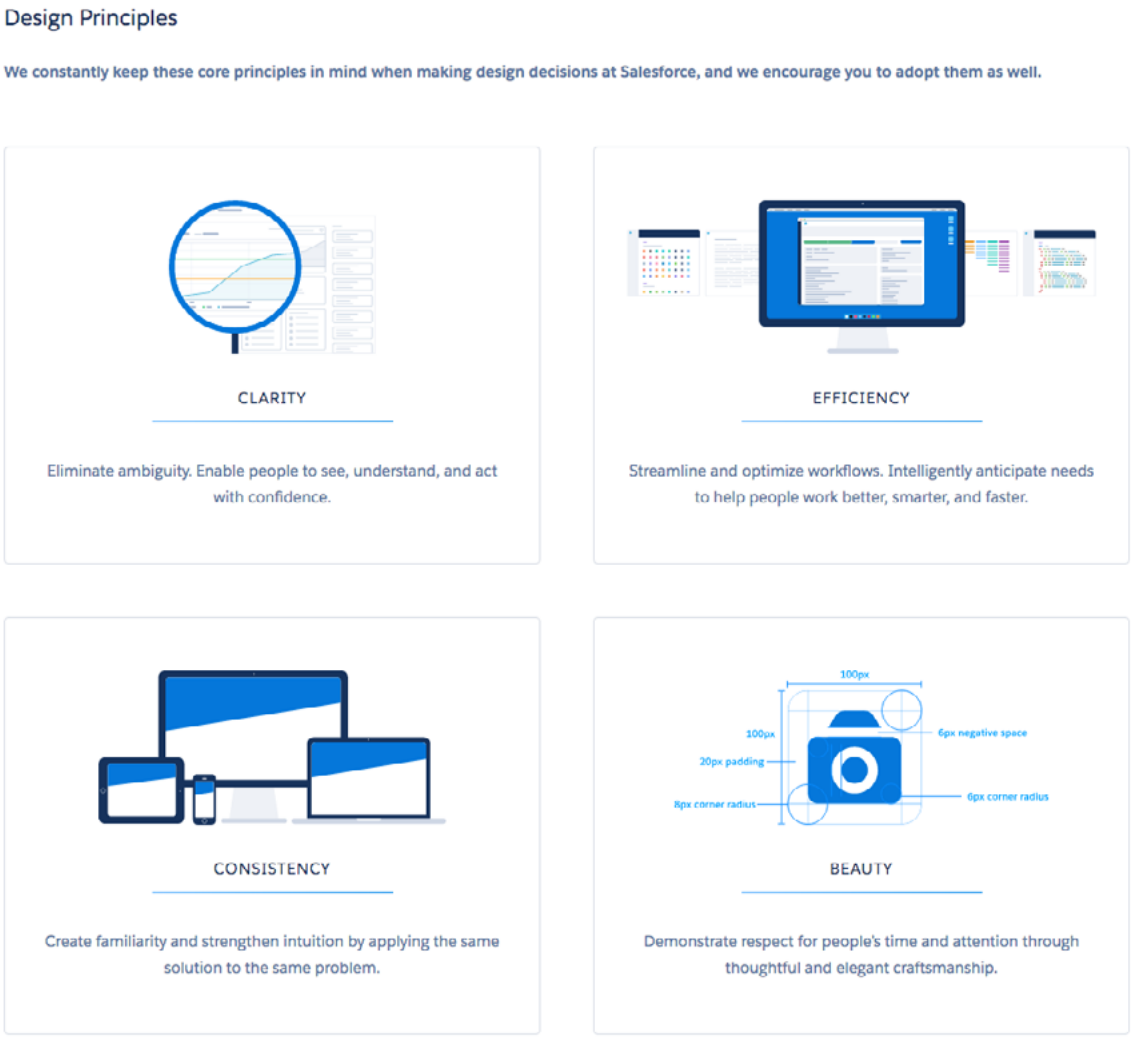


Photo credit: The design principles behind Salesforce Lightning.

Use your principles as a review heuristic for every new pattern proposed for the design system and every new project.

4. Unify your visual design

Think about the most fundamental and repetitive patterns in your interface.

Colors, text-styles, icons will probably come to mind first. Perhaps also some interactive patterns (hover on clickable elements), border visual properties, or maybe animations? Discuss with the team to decide which version of these elements will be canonical. Then, document them as part of the system.

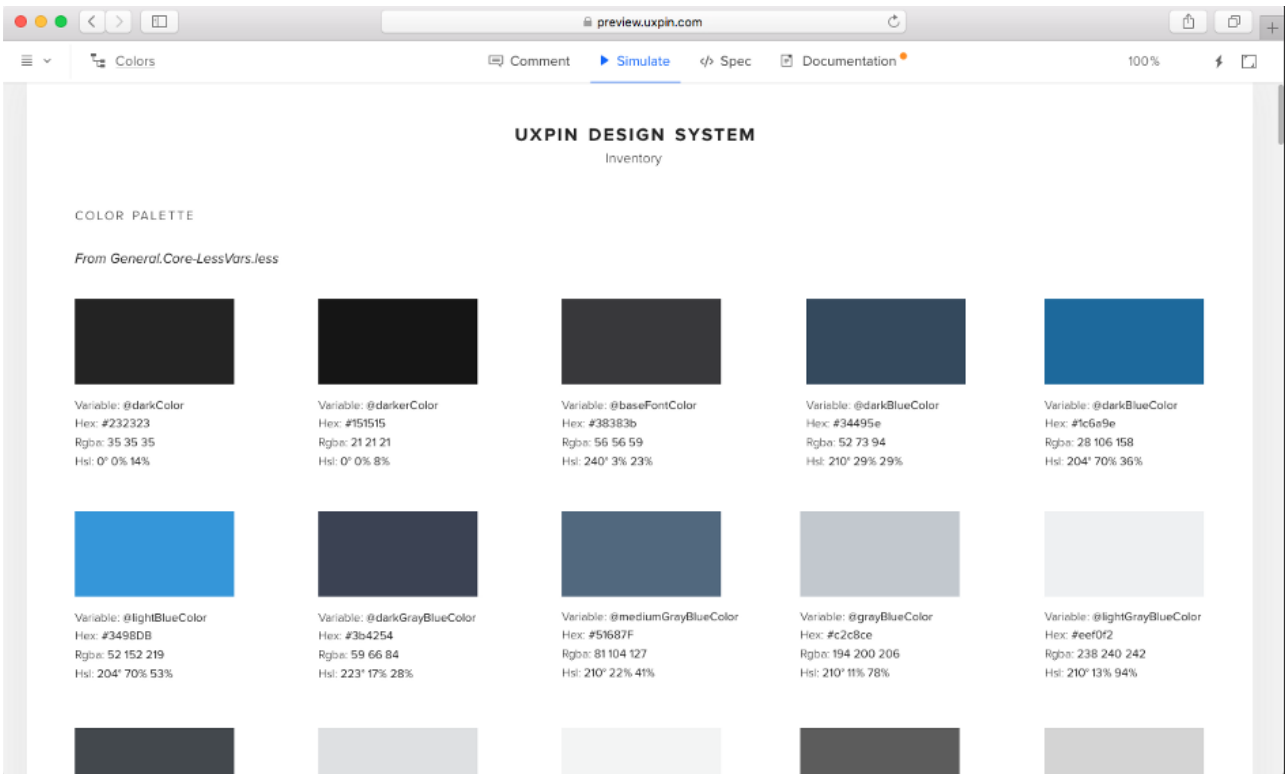
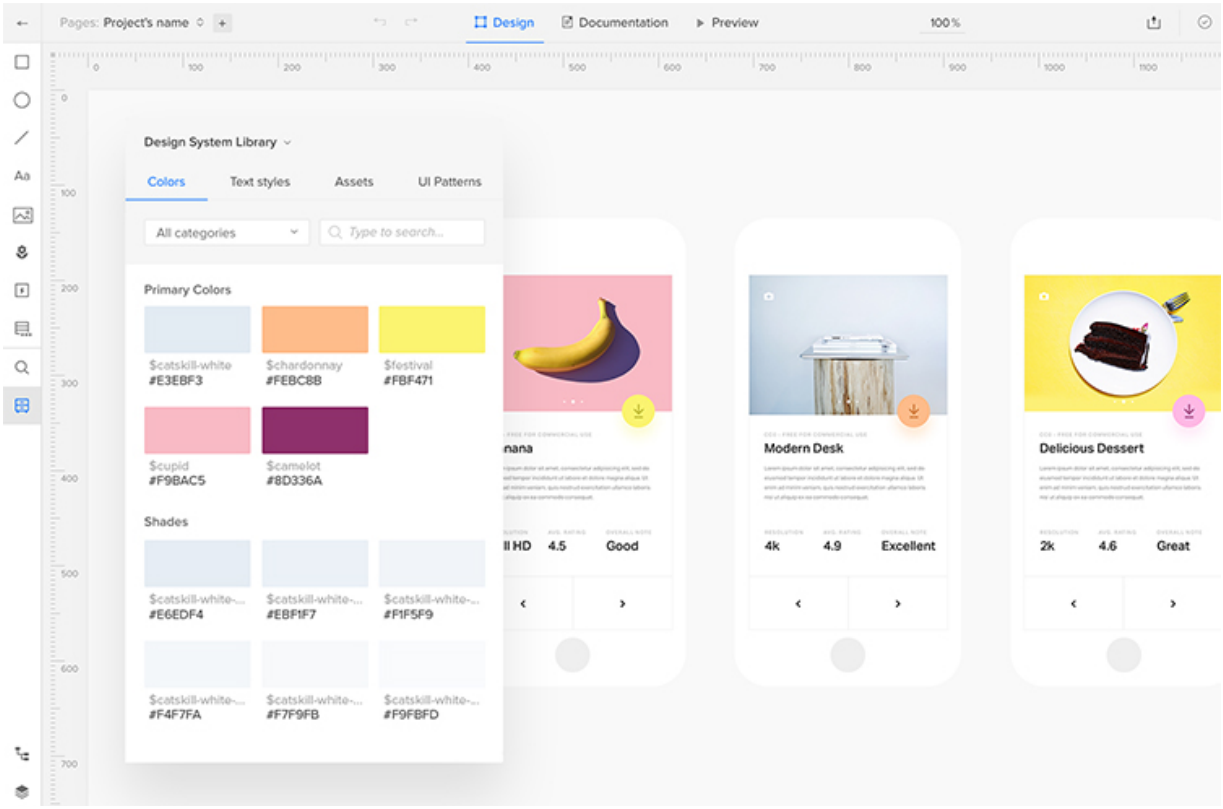


Photo credit: Our internal design system created in UXPin.

5. Create your interactive component library

Once you have the foundation well-defined, start adding your approved interface patterns to the shared library. Keep them updated and encourage the team to use them in every subsequent project – your efficiency and consistency will improve drastically.

Finally, plan a process of suggesting new patterns to make sure that everyone in the product team has a say in the evolution of the design system.



An interactive design systems library in UXPin

Remember that a design system is an ongoing process. In the beginning, dedicate at least a few hours a week of designer and developer time to maintain the system.

To learn more about design systems and get a full list of resources, download our *Actionable Guide to Starting Your Design System: A 100-Point Process Checklist*.

Democratizing Assets

On the administrative end, you should also standardize how you host and access product assets. Communicate a clear framework to the whole team.

For example, we use the following system across all our teams for every release:

- **Code assets**: Lives in Github.
- **Product planning**: Roadmaps live in ProdPad, while schedules live in TargetProcess.
- **Product documentation**: Lives in the UXPin design system, which syncs with all our projects. As a result, code snippets, use cases, and other documentation is always up to date.
- **User documentation**: Personas are created in UXPin and shared across the relevant projects. Additional artifacts like sketches and journey maps are saved as JPG or PDF and uploaded into the appropriate UXPin projects.
- **Design artifacts**: Sketch files live in Google Drive and are eventually imported into UXPin. All iterations of wireframes and prototypes are created in UXPin.
- **Design language**: All assets and patterns live in the design system within UXPin.
- **Usability tests**: Recordings live in Google Drive available to the whole company.

Since virtually no assets are isolated on individual desktops, we experience less version control issues, design inconsistencies, and misunderstanding.

Clarify Roles & Goals

After you've established a team structure and design system, you need to ensure people know how the whole ecosystem works.

Roles are more important than job titles. To minimize the email threads about who-does-what, clarify everyone's roles within the organization and on each project.

Roles & Goals in Organization

Beyond a quick org chart, consider creating a 1-page document together that outlines:

- **Overall vision**: What is the essence of the customer experience delivered by the product organization?
- **Quarterly OKRs**: What are the 3-5 top goals for product design and product management? How do those break down into specific measurable results?

- **Responsibilities by team:** What activities will product design own? What about product management? And project management? Who is maintaining the design system, and who can contribute to it?
- **Points of escalation:** Who leads each squad or chapter, and how can they help resolve issues?

Team,

As we grow the company and scale to the next level. Here is how the **Product** is built at UXPin.

Goal: Build a high quality design workflow product for teams

Responsibile:

Sunita (Product & Design Management), Daniel (Project Management), Pawel N (Development)

Main focus of the Product & Design team:

1. Creating a product road map and strategy with the right features to create an increase in sales pipeline
2. Set prioritization for releases
3. Create all the necessary artifacts (mockups, prototypes, visuals, high level user stories) needed for prioritized stories
4. Responsible for an up to date external roadmap for sales & marketing to be able to share with customers
5. Responsible for competitive matrix
6. Responsible for timely release of product (Sunita+Daniel+Pawel)
7. Monthly company product update

Who is the product team:

Sunita, Radek, Adam M, Batek D, Ben, Kamil, Adam Z., Sebastian

Main focus of the Project Management team:

1. Breakdown of the user stories into detailed tasks for estimation
2. Conduct sprint planning, sprint demos, sprint retrospectives & release demos
3. Manage the issue process with inside sales customers
4. Maintain communication flow with stakeholders, development team and the company regarding projects progress
5. Share lessons learnt after with the company after each project

Who is the Project Management team:

Daniel, Myron, Pawel W, Bartek K

Main focus of the Dev team:

1. Estimate the use cases prioritized by product team & broken down by project management team
2. Build the use cases articulated by the product team and prioritized
3. Fix bugs/issues based on SLA's with CS and sales team
4. Maintenance
5. Work closely with project management & product & design team
6. Keeping high level of quality/maintaining technical debt

Who is the Dev team:

Pawel N., Adam R., Adrian, Jacek, Jan, Jarek, Kamil, Karol, Krzysztof, Łukasz, Maciek, Małgorzata, Marcin G., Mateusz, Paulina, Paweł K., Paweł O., Piotr J., Piotr K., Przemek, Robert, Slawek, Szymon, Tomasz, Wojciech

Questions, Comments, please let us know.

Email from Sunita Reddy (VP of Product at UXPin) during her first month explaining our different "chapters".

As you can see above, the document shouldn't be exhaustively detailed. Keep it lean, since you want to encourage people to actually read it. If you follow the Spotify team model, describe each squad, chapter, and tribe in separate documents.

Roles & Goals in the Project

Each kickoff document should list the responsibilities for each team member in a simple chart. A separate section outlines the business goals.

Here's how we clarified roles for a recent redesign project.

Name	Position	Responsibility
Kamil Zieba Ben Kim	Product Lead	• Provide overall vision of the project from business AND user points of view (sync with CEO regarding the vision and priorities); • Determine the priority items for each iteration of development • Provide feedback on the execution after Sprint Demo; • Generate changes if necessary, approve/decline changes generated by others;
Kamil W.	Technical Lead	• Technical Guidance of the project; • Making decisions towards the architecture approach; • Code reviews at the end of each sprint;
Sebastian W.	Design Lead	• Provide guidance to the developers on the UI/UX design; • Conduct a UI/UX consistency review at the end of each sprint;
Kamil W. Robert H. Slawek P. Adrian S. Tomasz J. Przemek D.	Developers	• Estimate the work which will be completed; • Understand the scope of the assigned work; • Write clear, properly commented code; • Report on any issues/problems with the task to the whole team; • Ensure proper communication with the team; • Follow the guidelines/processes and methodology according to management approach and development best practices;
Myron K.	Project Manager	• Manage and Improve Execution of the Project; • Serve as a communication link between the Team and other departments/stakeholders etc.; • Help the Team reach Project Goals; • Scope/Milestone breakdown of the project, deadline setting;
Sunita R. (VP of Product) Marcin T. (CEO) Daniel A. (VP of Projects) Pawel N. (CTO) Ryan R. (VP of Marketing) Krzysiek S.	Stakeholders	• Responsible for the "business direction" of the project; Introduce/Approve/Decline Changes into the project; • Responsible for prioritizing work inside the project according to business requirements and market demand.

And here's the sections we use to break down the goals for an upcoming project:

How will we measure project's success?

Management criterias: TBD
Business criterias: TBD
Business Goals TBD
Behavioral goals (for features): TBD

Once you've shared the roles and goals across the organization and project, you've set the correct expectations for every undertaking afterwards.

Overcommunication is the Norm

The first casualty of distributed design is always communication.

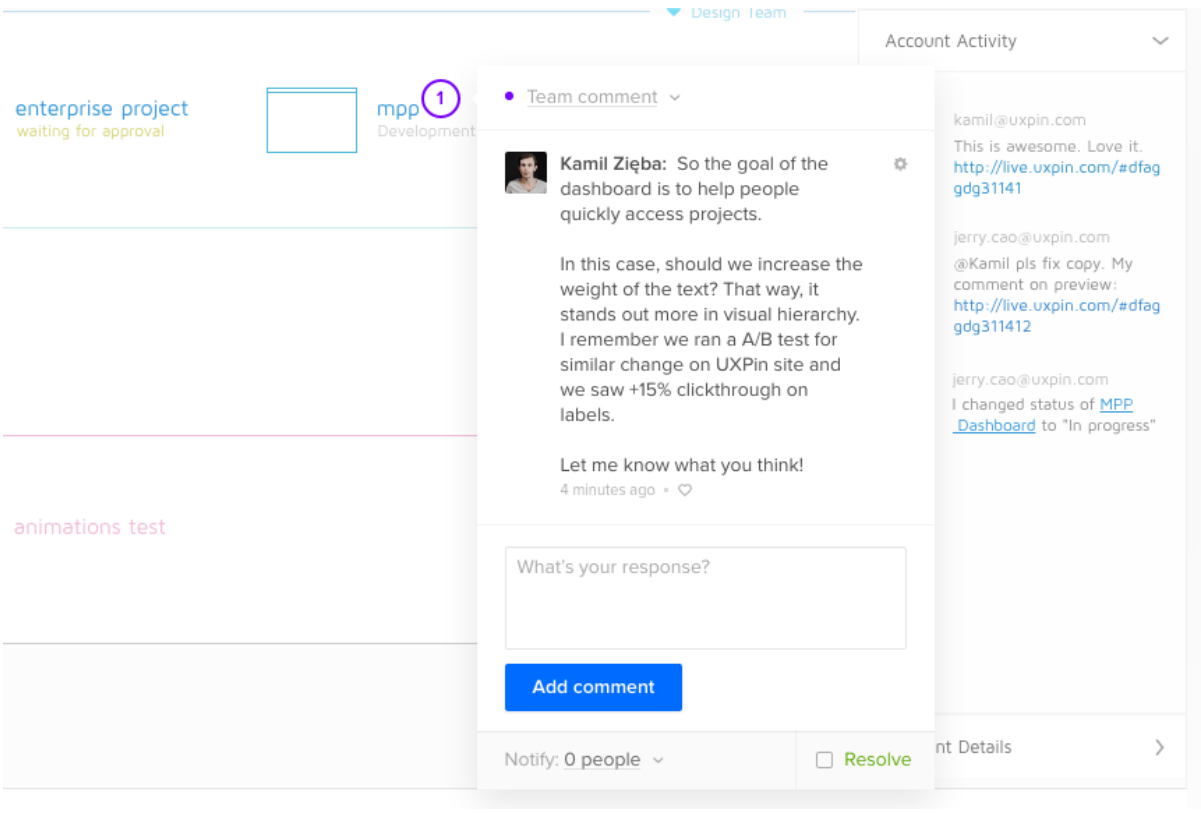
Any distributed team can attest to the great ideas that warped into confusing email chains. The only solution is overcommunication.

When offering feedback, provide as much rationale as possible. Predict the recipient's questions as much as possible. Otherwise, you might need to wait 6-8 hours just to see the other person is requesting information already at your fingertips.

Consider these guidelines:

- **Reiterate the goal**. Your feedback needs an anchor. Otherwise, if the person disagrees, you get caught in back-and-forth without a clear rallying point.
- **Cite supporting evidence.** Imagine waiting 8 hours until the next day just to see the comment "Have we tested this before? Please attach relevant insights.". Lay out all supporting research in your first comment.

- **Be as specific as possible.** A comment like “This color looks off” is difficult to interpret correctly. Is the color actually wrong? Or does the background make it appear unsuitable? State *why* something doesn’t work and how you *might* fix it (not how it *should* be fixed).
- **Ask for recommendations.** Overprescription is just as bad as vague feedback. If you suggest a solution, clarify that it might not be the only option. Otherwise, the other person might interpret it literally, which can lead to hours of wasted work on top of time zone delays.



Commenting on a feature redesign in UXPin

In the above example, notice how the feedback incorporates all of the above points. Kamil clarifies the goal, cites past A/B tests on a similar project, explains the element he’d like to change, and then asks for the team’s thoughts.

In doing so, the comment minimizes the risk of someone asking for information that he could have clarified originally.

Standardize Your Workflow

Without a universal “battle plan”, each location may suffer from fragmented processes. A standardized process creates one point of reference.

You won’t standardize a process overnight. Treat standardizing a process as its own design problem requiring discovery and iteration:

1. Hold a 30-minute video conference call with senior members of the product team to learn their current processes. Diagram each person’s process, then compare for similarities and differences. Note any gaps or redundancies.
2. Create the new process for an upcoming project as the pilot experiment. Choose a bite-sized project (e.g. a feature update rather than a full redesign).
3. Show the first iteration of your the process back to the people who filled in the spreadsheet and participated in the interviews. Revise based on patterns in feedback.

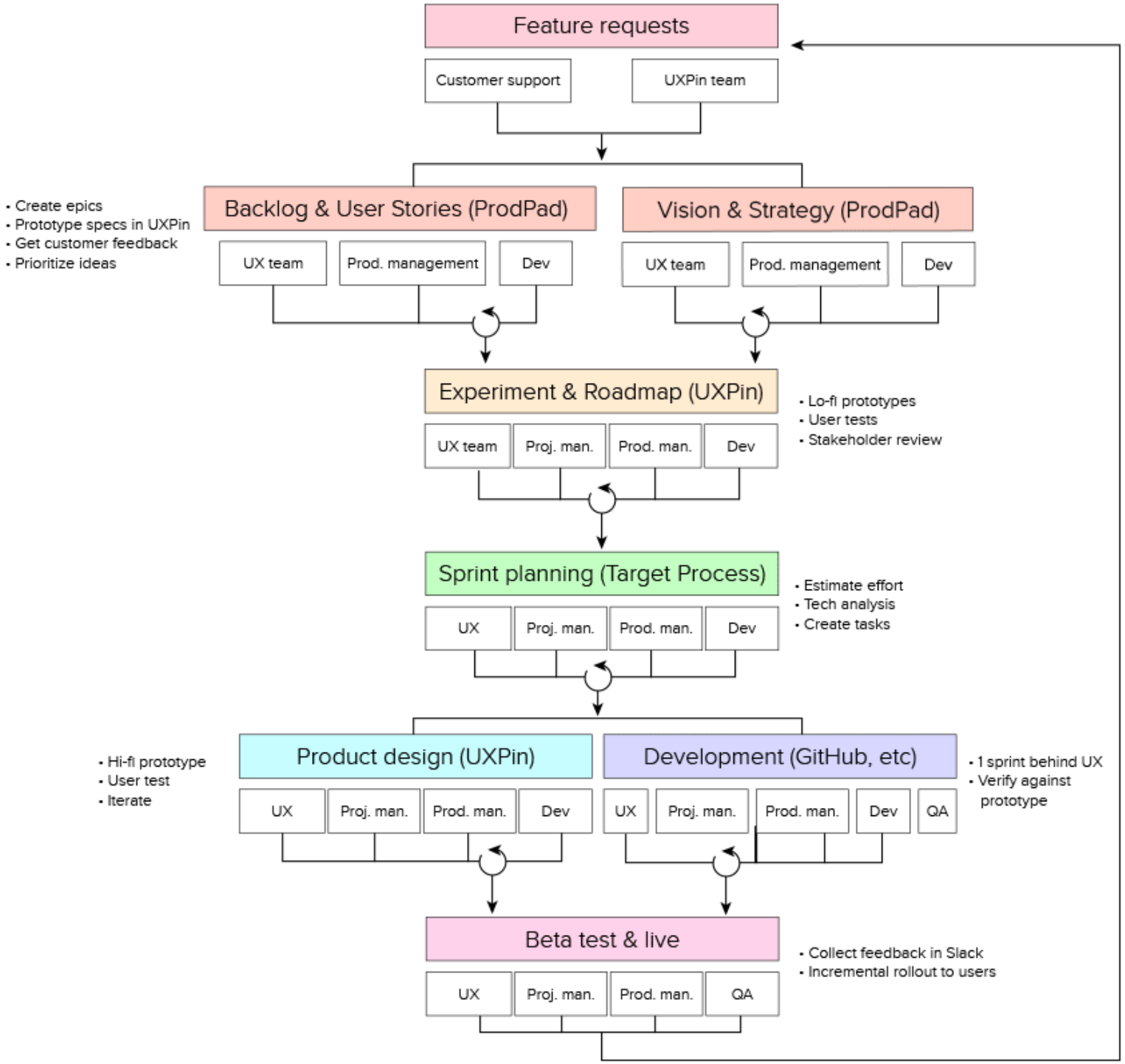
4. After sharing your finalized version with everyone, hold daily calls with each location to learn how the process performs in the pilot experiment. Where do roadblocks still occur? Who's experiencing issues with collaboration? Continue iterating until the team falls into a natural rhythm and you notice a decrease in questions about process.

5. As people become comfortable with the process, streamline your tool suite. Create a spreadsheet for everyone on the product team to write down their set of tools. Review the document, then see where you can eliminate redundant tools or add new ones. Now you have an efficient process and the right tools for execution.

Finally, it helps to create a flow diagram to visually summarize:

- Where to send feature requests
- Who's involved in each stage of product design and development
- Where roadmaps, schedules, and design/development assets live
- How to prioritize and validate concepts
- How to plan design and development sprints
- How to design, implement, iterate features

For example, our VP of Product Sunita Reddy (based in the U.S.) uses the below visual on her daily international status calls to clarify process questions. The below version is our second iteration since standardizing in April 2016.



Adapt to Virtual Workshops

In our experience, distributed design is most challenging in the earliest stages of product development. Due to a murky problem definition, it's easier for insights to get lost in translation. Everyone isn't working off the same whiteboard in a room together, so it's harder for a team to go back-and-forth until they reach alignment.

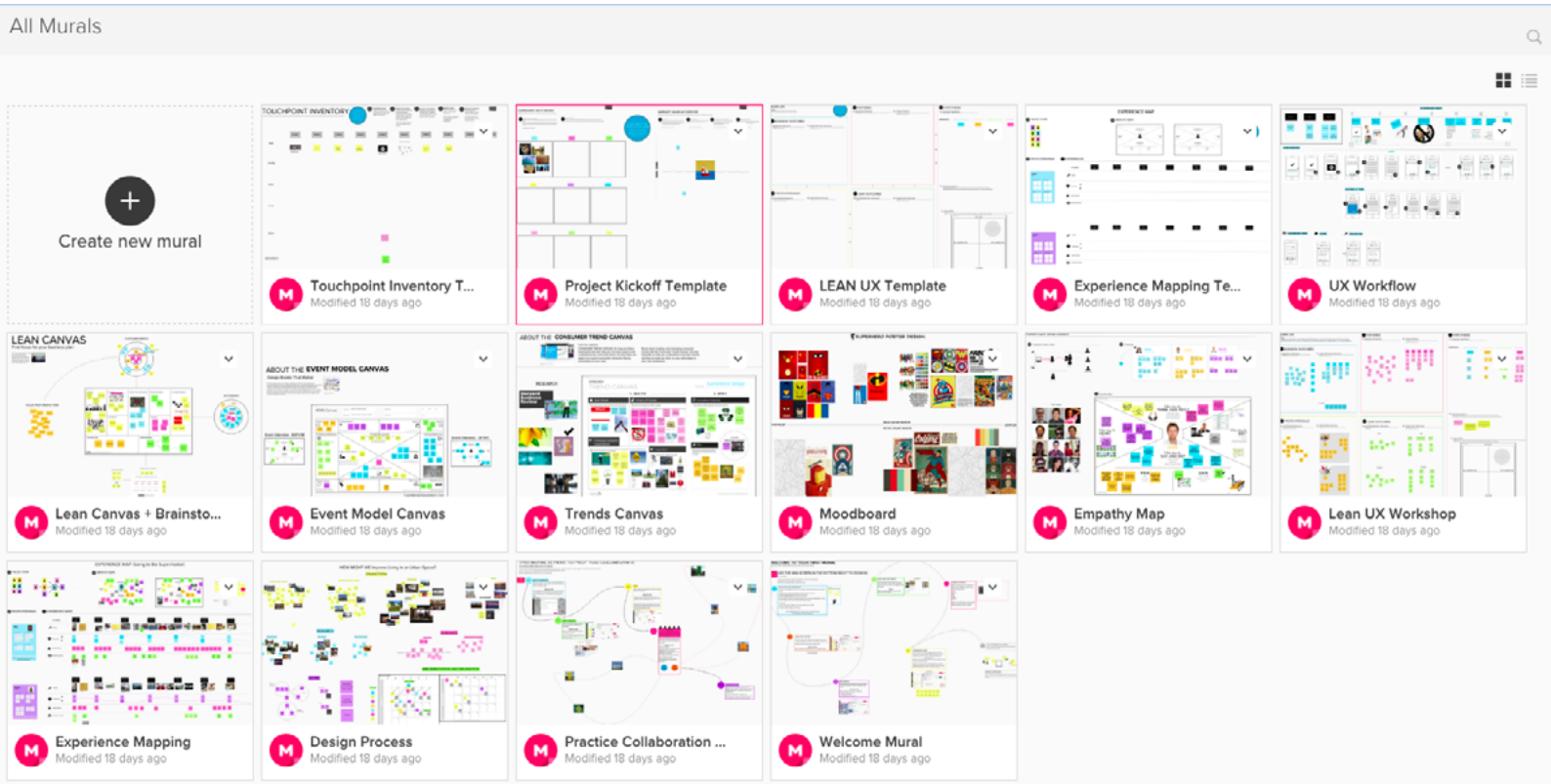


Photo credit: Mural

Luckily, with a few modifications and tools, you can conduct successful distributed workshops and kickoffs. To make the most out of the exercise, try using the following tools:

- **Camera** – We recommend any of the IPEVO series of cameras (the IPEVO 2 is also specifically recommended by UX experts Jared Spool and Jeff Gothelf). In a pinch, you could inconveniently tip your laptop webcam to focus on the sketch, but the resolution won't be as great.
- **Projector and screen** – Chances are not all of the participants will be distributed, which means you'll need a method to display the distributed work to those who were able to gather in person.
- **Screen-sharing software** – Your preferred method of screen-sharing (like GoToMeeting or Join.me), so that all the distributed members see the same visuals as everyone else.

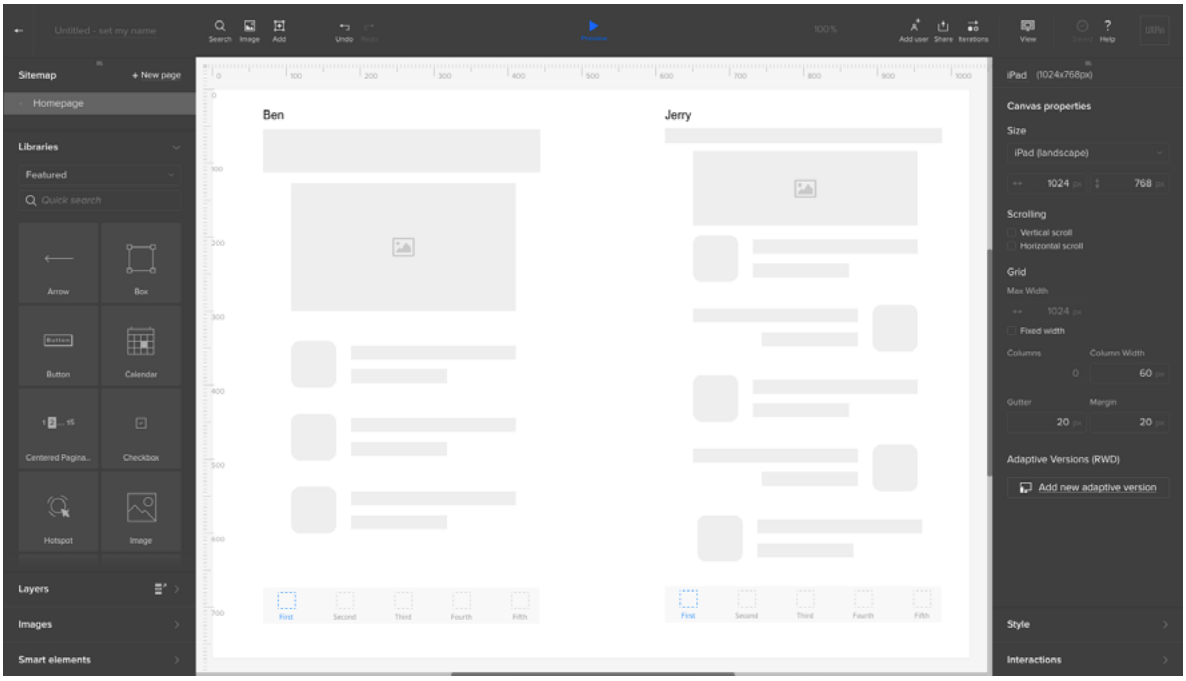
Of course, you can also brainstorm with a free collaborative sketching tool like Google Drawing, in which case you won't need a USB camera or projector. However, keep in mind that you'll likely want a stylus, and the thought of "sketching software" might intimidate non-designers.

If budget allows, we highly recommend investing in a comprehensive platform like Mural. It's designed specifically for ideation among product teams and even eliminates the need for the tools mentioned before.

Make sure to timebox your distributed design studio exercise. For example, if you're leading a workshop that lasts 1 hour, only allow 10 minutes for the first iteration. The time limit actually encourages more unfiltered ideas and less worry over details. Always ask if people in the other location have questions, since silence doesn't imply agreement.

In addition to sketching workshops, our Chief Design Officer Kamil Zieba sometimes holds prototyping workshops in UXPin. In this case, everyone logs into UXPin, joins the "Sandbox" project, then hops onto a Zoom video call. One full cycle with 3-4 people includes 10 minutes co-designing, 10 minutes critiquing, and 10 minutes iterating. We repeat the cycle based on the complexity of the problem.

The exercise works well for fast-paced projects where we need to generate interaction models quickly.



Timeboxed prototyping workshop in UXPin with two participants.

To learn more about remote UX workshops, we recommend these resources:

- Remote Collaborative Brainstorming & Sketching
- Designing With Remote Teams
- Remote Design Sprints

Create Roadmaps Together

As ideas start to take shape from brainstorming sessions, your team needs to prioritize based on user and business value.

Using tools like Zoom and Google Sheets, you can create a detailed roadmap without endless email chains. Roadmaps represent iterations of your strategy – not the delivery schedule.

Our roadmapping sessions are usually spread out across 5 days. We use Zoom to conduct and record the videoconference, then categorize and prioritize everyone's ideas in a Google spreadsheet. At the end of the exercise, we end up with an external roadmap and internal roadmap.

1. **Day 1: Goal Setting / Strategy Setting**
   The first day is a straightforward overview session. Developers, product owners, and designers attend to discuss:

   - **High-level business goals**: Are we aiming to increase the average size of deals? Do we also want to appeal to new personas?

- **The end deliverables**: VP of Product clarifies that the team will create two roadmaps. The external roadmap is presented to sales and select customers for validation, while the more detailed internal roadmap is for product leads to create use cases.
- **Long-term vision**: VP of Product explains the vision that each feature in the roadmap will support. Since we're a startup, we limit our vision to 6 months due to the fast pace of the market. In more established enterprises, it's not uncommon to create up to a 5-year vision for individual business units.

Once you factor in answering questions from each member of the team, the session lasts around 1 – 1.5 hours. At the end of the session, every team member is asked to add in their ideas for features into the Google spreadsheet. The spreadsheet categorizes all ideas based on target user persona.

2. **Day 2-5: Culling the List**

The prioritization part of the exercise requires 3 days because you may end up with 50 items listed in the spreadsheet. Due to the depth and detail of discussions, 2-3 hours are required for each session.

You can scale the exercise down or up based on your team size and product complexity.

Now, each person explains why they included their items in the spreadsheet, citing any supporting research. The entire team reviews each item based on three criteria:

- **Viability** – What business value does the idea offer?
- **Desirability** – What user value does this idea offer?
- **Feasibility** – Can we actually build the idea with current resources? If not, when might we?

If you decide an idea is viable but haven't validated its desirability, tag it with an asterisk for more user research. Move each idea into different categories such as usability improvements, maintenance, nice-to-haves etc. Timing in the roadmap (3 months, 4-6 months, 12 months) also depends upon the balance of the three above criteria.

The workshop-based approach to creating roadmaps sets the expectation that the document will evolve, allowing product owners to then iterate based on stakeholder feedback and user research.

Validate Ideas with Remote Research

In a distributed team, remote research is the best choice for scaling user insights quickly – whether that's validating roadmaps or MVP builds.

Let's say 10 designers are scattered across multiple locations. Instead of waiting for the person with the right equipment and environment to run user tests or interviews in sequence, all 10 can conduct research simultaneously.

Most of the 20+ usability tests and 80+ user interviews we ran for our recent redesign were remote.

In terms of planning, we recommend a 5-day research sprint, followed by 3-4 hours of weekly remote interviews and testing. The process significantly increases user exposure hours across multiple offices.

Let's explore best practices for remote moderated usability tests.

Recruit Participants

Since UXPin is an enterprise product, we handpick all our participants based on strict background and experience levels.

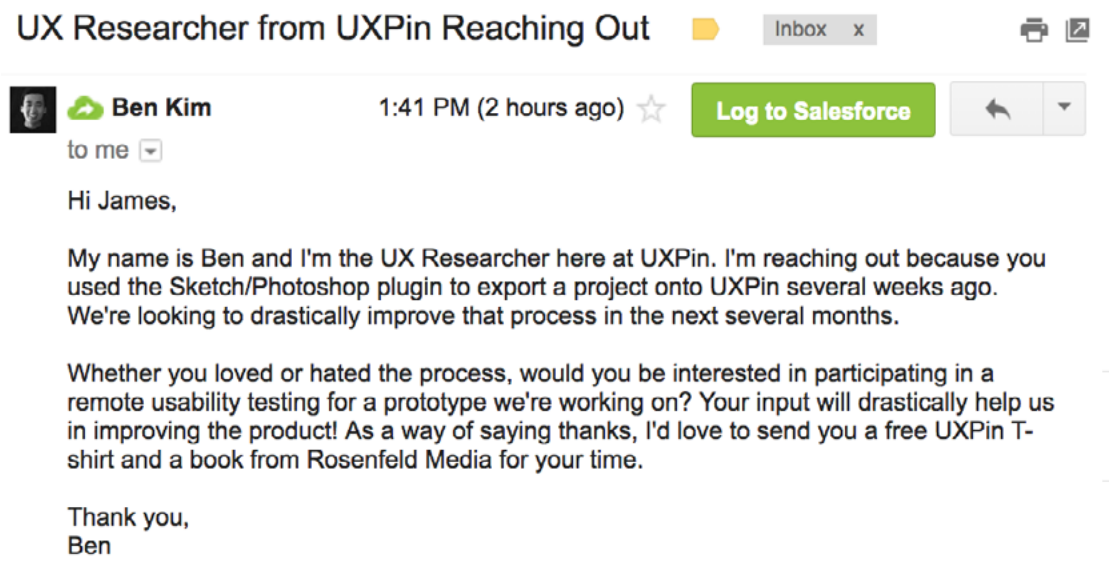
When screening current users, we'll first send out a qualification survey. The criteria is based on profiles we've created from behavioral data in Kissmetrics.

When screening external users, we'll use Linkedin to segment by job title, experience, and company. We'll then use the free Rapportive Chrome extension to validate email addresses. To follow up, we use the Streak Chrome extension to schedule follow up emails.

In all our outreach emails, we emphasize the following points:

- Feedback is completely confidential and won't be used outside of the company
- We're seeking their expertise (always helps to make people feel important)
- Exactly how much time is required (no more than 1 hour for user interviews and usability tests)
- How we compensate their time. Generally, some company swag will work just fine, but we also send over $20-$50 Starbucks or Amazon gift cards.

Contact three times as many people as you'd need to interview or test. For example, in our recent redesign, our user researcher contacted roughly 230 people for 80 user interviews and 70 people for 25 usability tests.



UX Researcher from UXPin Reaching Out

Inbox x

Ben Kim 1:41 PM (2 hours ago) Log to Salesforce

to me

Hi James,

My name is Ben and I'm the UX Researcher here at UXPin. I'm reaching out because you used the Sketch/Photoshop plugin to export a project onto UXPin several weeks ago. We're looking to drastically improve that process in the next several months.

Whether you loved or hated the process, would you be interested in participating in a remote usability testing for a prototype we're working on? Your input will drastically help us in improving the product! As a way of saying thanks, I'd love to send you a free UXPin T-shirt and a book from Rosenfeld Media for your time.

Thank you,
Ben

Conduct Remote User Interviews

Before you start the interview, select a conference tool that records video. Whenever possible, invite a non-designer to attend too. We always invite at least one developer and PM.

Once you start the interview, adapt a few best practices of in-person interviews:

- First, state the purpose of the interview. For example, here's how our user researcher starts interviews with current customers: "Let me quickly go over with you the topic of this call. First, I'm really curious to learn more about how your company uses UXPin and

how it fits into your product workflow. Second, I'm going to show you some prototypes and share some ideas on possible future features."

- If you aren't interviewing current customers, start with broad questions about their current process. For example, "How does your product team collaborate at Citrix?" You can drill down into details with follow-up questions like "Take me through what you did the last time you wanted feedback on your prototypes."
- Since the participant isn't in-person, you can't read body language as well. Pay even closer attention to facial expressions and long pauses. Any signs of discomfort or confusion are red flags for you to ask a follow-up question.
- On a related note, don't cave into the urge to fill awkward silences with conversation. Give the participant time to respond.

At the end of an interview, ask some quantitative questions. For example, our user researcher recently spoke with 7 companies about 10 potential features in the next release of UXPin. To close the interview, the researcher asked users to rank the usefulness and urgency of each idea from 1 to 10.

The quantitative questions helps make subjective product ideas more objective. The qualitative questions offer insights into how the product fits into a user's current process.

Conduct Remote User Tests

For remote tests, you can conduct either moderated or unmoderated sessions.

Moderated remote tests allow for the most real-time interaction with the participants. We recommend scheduling them for in-depth qualitative feedback. Moderated tests are especially useful for early-stage prototypes where users may need someone to guide them back on task.

Unmoderated remote tests help to scale insights quickly. Since nobody can guide users if they get confused, unmoderated tests are recommended for more polished prototypes.

Some of our customers have reported that successful test plan will include 30% moderated tests (with 10-15 users) and 70% unmoderated tests (with up to 30 users).

Since we run mostly moderated remote tests, here's some lessons we've learned through the years:

- Building rapport is even more crucial for remote moderated tests, but even easier to forget due to the physical and emotional distance. Don't dive into the test just yet – spend a minute or two asking about their day. Think of it as an investment towards more honest results.

- Just like remote user interviews, pay close attention to visual and auditory cues. If a user expresses confusion out loud, ask them "What are you thinking right now? Would you mind explaining why you're doing that?". If a user says "Hmmm..." or stops thinking out loud altogether, say something like "I noticed you were quiet there for a moment. Would you mind explaining what you were thinking?".
- Revise your task scenarios at least twice. You want to eliminate all possible bias while providing the right context for users.

1. **Circulate the Insights**

   As you interview and test users, remember to capture your insights for everyone with collaborative documents.

   You can link to the documents during standup and embed them in emails for larger projects. If further explanation is required, you can also hop on a call to discuss the insights and recommendations.

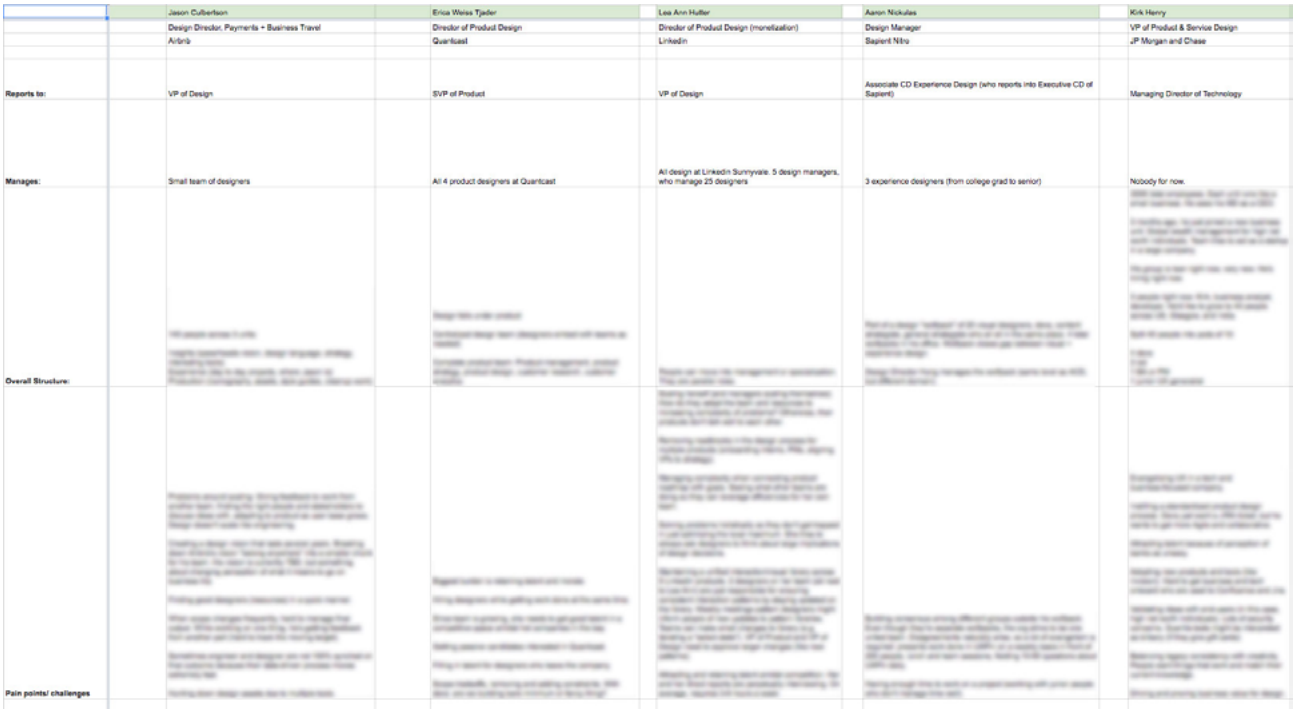
   Here's a few collaborative documents we find useful for creating a common understanding.

2. **Remote User Interview: Details Spreadsheet**

   The details spreadsheet are your raw notes from user interviews.

   On the X axis at the top, write out the names and job titles of participants. On the Y axis, write down the categories of insights.

In the below example, our Y axis includes labels such as “Team Structure, Pain Points, and Design Process”.



	Jason Culbertson	Erica Weiss Tjader	Lea Ann Hutter	Aaron Nickulas	Kirk Henry
	Design Director, Payments + Business Travel	Director of Product Design	Director of Product Design (monetization)	Design Manager	VP of Product & Service Design
	Airbnb	Quantcast	Linkedin	Sapient Nitro	JP Morgan and Chase
Reports to:	VP of Design	SVP of Product	VP of Design	Associate CD Experience Design (who reports into Executive CD of Sapient)	Managing Director of Technology
Manages:	Small team of designers	All 4 product designers at Quantcast	All design at Linkedin Sunnyvale. 5 design managers, who manage 25 designers	3 experience designers (from college grad to senior)	Nobody for now.
Overall Structure:					
Pain points/ challenges					

Raw notes for user interviews conducted with design leaders. Each column represents a person’s responses. Sensitive information blurred out.

3. Remote User Interview: Executive Summary

The executive summary is exactly as it sounds: a distilled version of the Detailed Spreadsheet. It acts as guidelines for designers and developers and as empirical evidence for business stakeholders.

As discussed previously, include the quantitative summary of feature rankings at the top of the document. Afterwards, bullet out the top insights learned in user interviews.

Q2 of 2016 Product Validation Interviews: Executive Summary

Companies Interviewed: Sapient, Ford, Deluxe, Tune, Legalzoom, LookThink
Dates Conducted: mid-June to early-July

Average Ranking of Features:

Feature	Avg. Usefulness	Avg. Urgency
[Feature 1]	5.83	4
[Feature 2]	3.75 with 2 N/As	4.5 with 2 N/As
[Feature 3]	6 with 1 N/A	6 with 1 N/A
[Feature 4]	5.5	4.83
[Feature 5]	9 with 2 N/As	7.75 with 2 N/As
[Feature 6]	7.25 with 2 N/As	7.5 with 2 N/As
[Feature 7]	7.5 with 4 N/As	8.5 with 4 N/As
[Feature 8]	6.75 with 2 N/As	5.75 with 2 N/As
[Feature 9]	5.75 with 2 N/As	6 with 2 N/As

Notes About Each Feature:

[Feature 1]
- Insight 1
- Insight 2
- Insight 3

[Feature 2]
- Insight 1
- Insight 2
- Insight 3

[Feature 3]
- Insight 1
- Insight 2
- Insight 3

4. Remote Usability Testing: Rainbow Spreadsheet

We adopted the Rainbow Spreadsheet based on advice from Tomer Sharon (VP of UX at WeWork).

As he describes in his tutorial, the document helps different people on a product team contribute their thoughts after observing a user test. Different tabs then help you to consolidate insights and prescribe recommendations.

	User 1	User 2	User 3	User 4	User 5
User feels interface is overwhelming					
Prefers "search" over browsing the categories					
Requested that "Accepts Credit Cards" be a top-level filter					
Wants photo gallery accessible on results page to assess restaurant ambience					
Bookmark feature was frustrating					
Needs clearer indication of price ranges					
Felt it was easy to sort restaurants by "Open Now"					
Could not find the Events tab					

Rainbow spreadsheet created by our CEO Marcin Treder for a Yelp redesign exercise

All of the above tactics are designed to help you and your team break away from the email habit. While the process won't always run perfectly, it's certainly a better alternative than user insights scattered across communication chains and desktop folders.

Estimate Effort With Planning Poker

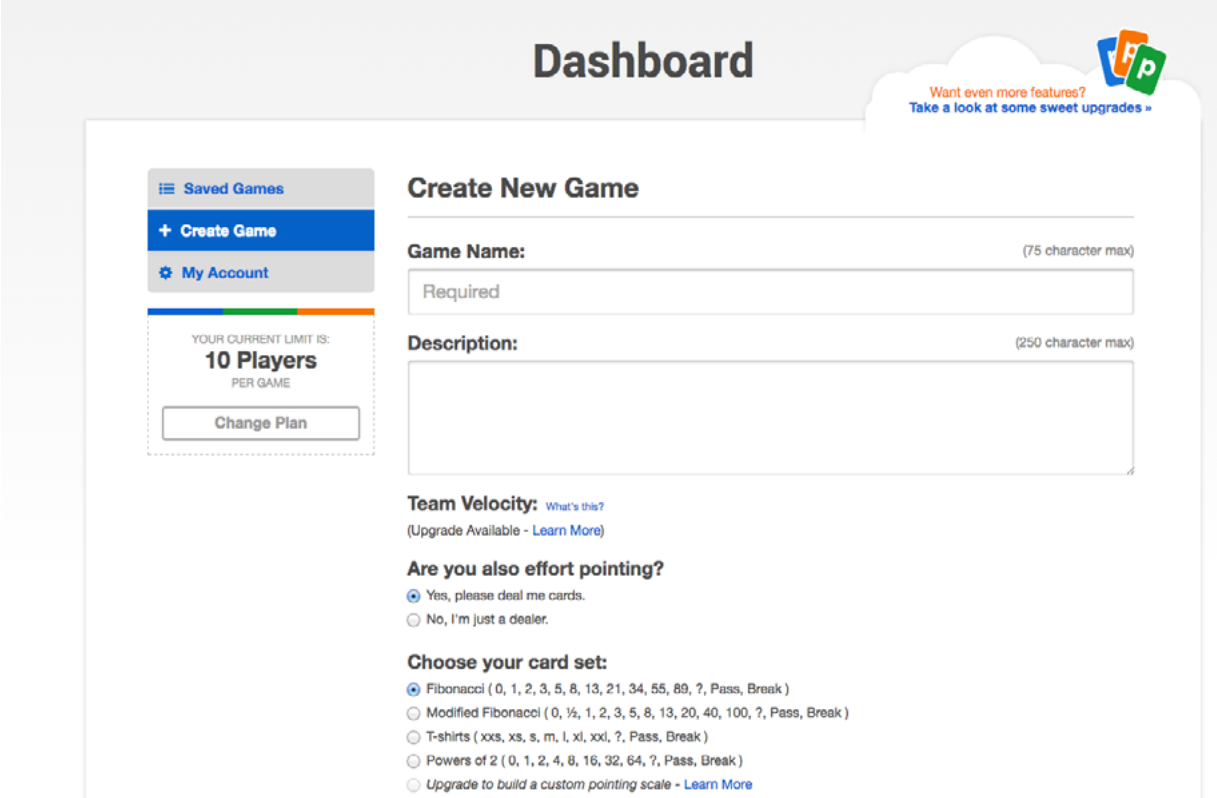
Once you've created a backlog of features from user insights, you need to evaluate effort for planning team schedules.

Planning poker helps everyone estimate the effort of user stories together. The earlier stage of roadmapping helps you assess impact – now you compare that against effort for final prioritization.

To kickoff the exercise, each team member is dealt stack of cards. Each card is a number that represents the number of story points or days a feature requires. As the product owner describes the feature or user story, each member lays out the card they feel best represents the work required. Since nobody sees each other's estimates before the reveal, you avoid anchoring bias (e.g. a person won't say "2 weeks" because the first person said "1 week").

In a distributed team, it's too easy for people not in the same location to stay quiet until it's too late. The beauty of planning poker is that it helps everyone air concerns and reveal hidden requirements

in a tightly focused setting. Everyone reaches consensus before any sprints are planned.



The free online tool Planning Poker makes the exercise accessible to teams in any location.

Here's how we've run the exercise:

1. Project manager Myron Kokhanovskyi invites the developers (Poland) and designers (US and Poland) for 30-60 minute session of online Planning Poker. The exercise is planned for when a product backlog already exists.

2. Myron (the product owner) explains the number of user stories the team will estimate. He then explains the first user story. The team is given 5 minutes to discuss the story and place the appropriate card.

3. Now Myron assesses all the cards laid out. The Planning Poker app also provides an average score of all cards. Myron now looks for the two outliers and ask them if they feel the average score feels fair. If so, he saves the score and the estimation is done for the user story. If not, the team discusses why and plays another round.

4. Repeat the process until consensus is reached. When we first started the exercise, it took up to 3 rounds per user story. Over time, however, our teams got into a rhythm and could reach consensus in 1-2 rounds. .

Although the consensus score isn't 100% accurate, the exercise reveals where false assumptions lie. Even if the consensus on a story is 5 when it's really 8, you're not totally off the mark (and you know what to watch out for).

More importantly, you'll better understand the specific tasks within each user story as the team discusses estimates.

Illuminate Requirements With Prototypes

As you start to plan feature work, you will eventually need to create documentation.

The more documentation you create, however, the greater the chance of misinterpretation by distributed teams. That's assuming they even read it cover-to-cover.

Instead of encyclopedic requirements, we've found that short PRDs paired with annotated prototypes minimizes misunderstanding. The short PRD creates context for the prototype, where the annotations offer further context on implementation.

Interactive documentation means that the design is less separated from the documentation. We've heard from some of our users that this hybrid process speeds up their design sprints by up to 50%.

Here's how it works.

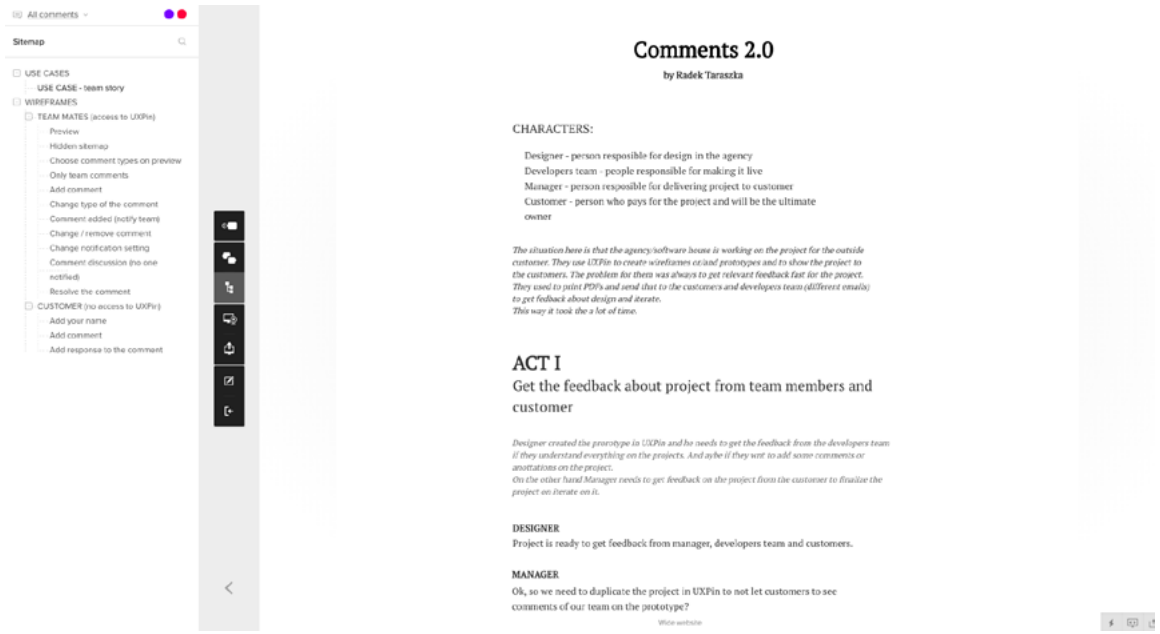
Treat your PRD as a synopsis (not a laundry list)

Your product design is the full story. The PRD should only be your summary, otherwise you're doing double duty.

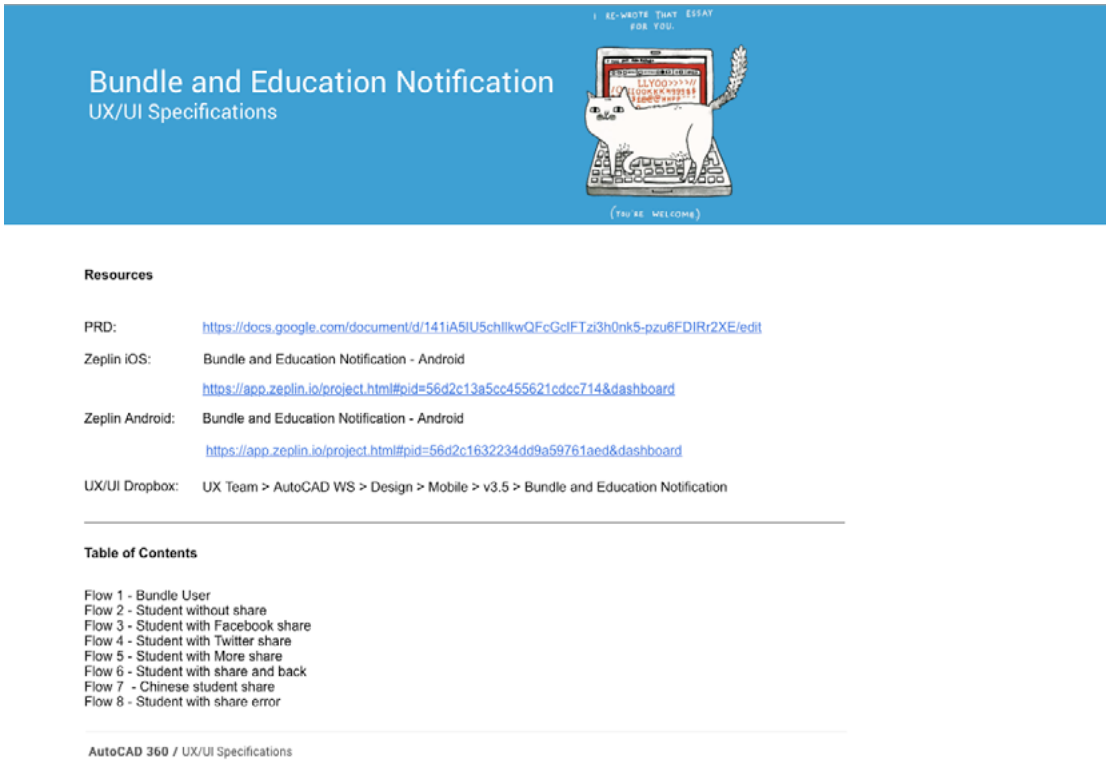
Summarize these points from our free PRD template:

- **Goals**. Why should the product exist? What specific business goals do you want to meet, and by when?
- **User Problems.** What are the top 3 problems you wish to solve?
- **User Stories**. You can either list out the user stories, or link to your product management platform.
- **Dependencies.** Are there any resourcing or timing issues the team needs to remember? For example, is anyone going on vacation soon?
- **Release Schedule.** Link to the schedule in your project management tool.
- **Related Documentation.** Link to personas, privacy policies, or any other detailed notes.

Our product team conveys the above information as a "Cover Sheet" in UXPin for each project. We create the documentation in a story format with personas as characters. We explain the problem, our goals, and then link out to the rest. We've found this concise format actually engages readers without sacrificing information.



Of course, you can always approach the PRD as a more straightforward knowledge portal, as Autodesk does below. In this case, they actually list out all the user flows in the document (instead of linking to separate projects).



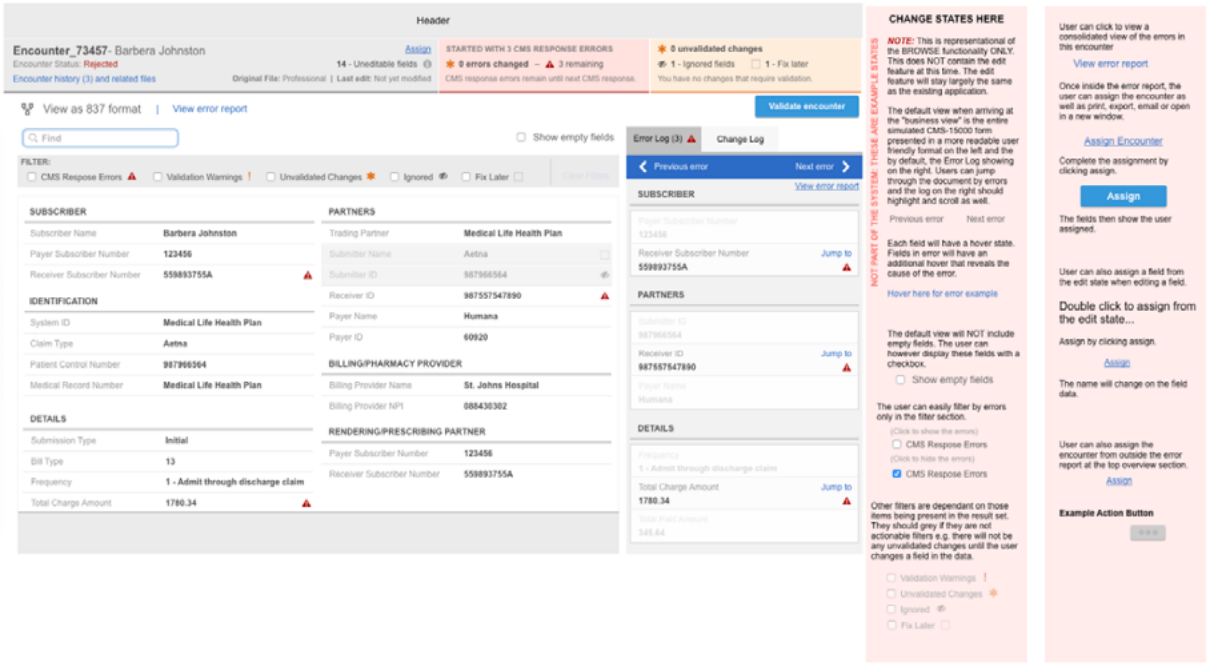
Requirements portal created in UXPin by Autodesk's AutoCAD 360 team

Once you've created your PRD, you can hold a 30 minute to 1-hour review to answer any initial questions, then dive straight into prototyping. As you prototype, you'll uncover even more requirements. The process becomes iterative and collaborative, rather than just a massive file dump for people to wade through.

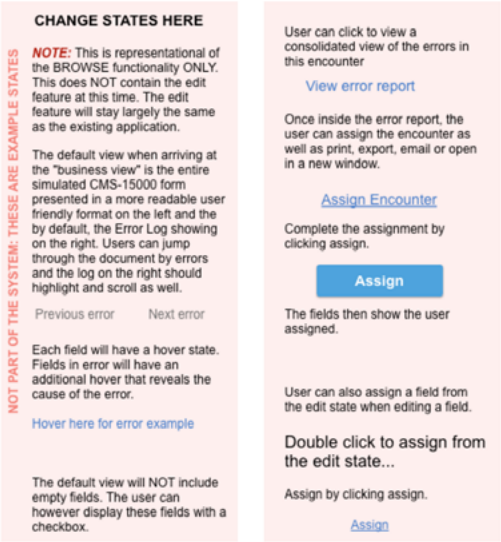
Annotate prototypes with details of each user story

Now that you've communicated the overall vision, goals, and constraints, you can include the details contextually.

In the below example, LookThink describes the nuances of the "Browse" functionality for an enterprise healthcare product. They explain the pending features, expected behavior, and the user flow for the function.



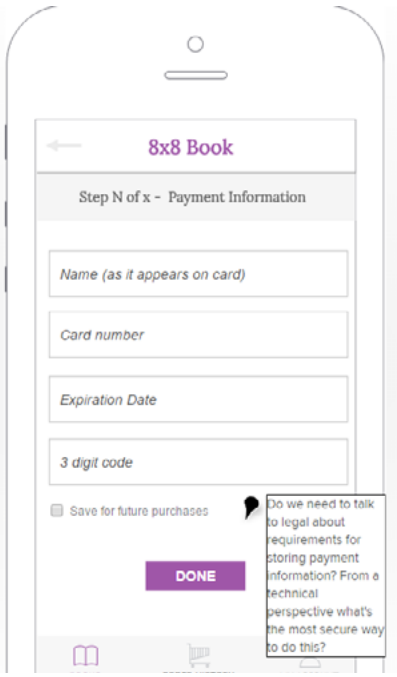
Annotated lo-fi prototype.



Zoomed view of annotations

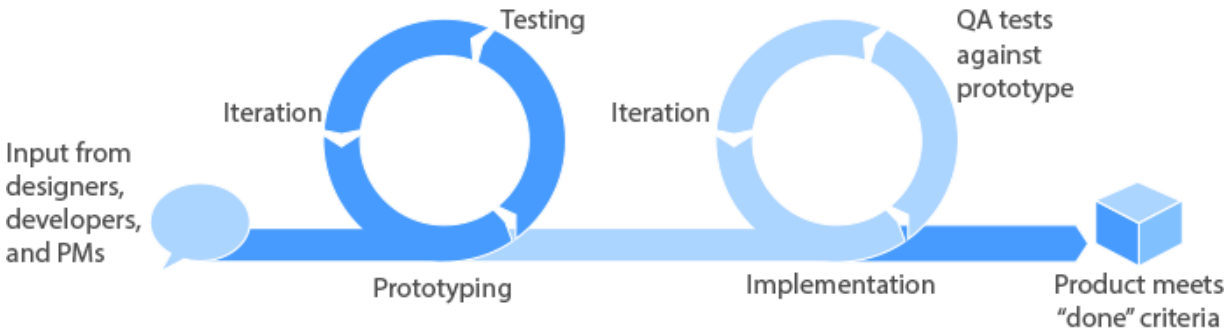
This level of detail in annotations isn't always required for advanced prototypes, but you can't go wrong with overcommunication.

As shown below, annotate any risky technical or legal questions. The distance between teams makes it that much more tempting to dive into heads-down work, so you want to unearth as much risk as possible beforehand.



With just the PRD and prototype links, the collocated and distributed team can both understand the scope and requirements. And in case requirements change, a PM only needs to revise the short PRD since the designers would naturally update the the prototype.

No more hunting around for scattered documents to find the latest definition of “Done”. The latest prototype now becomes the acceptance criteria for product and QA teams.



Adapt Your Daily Standups

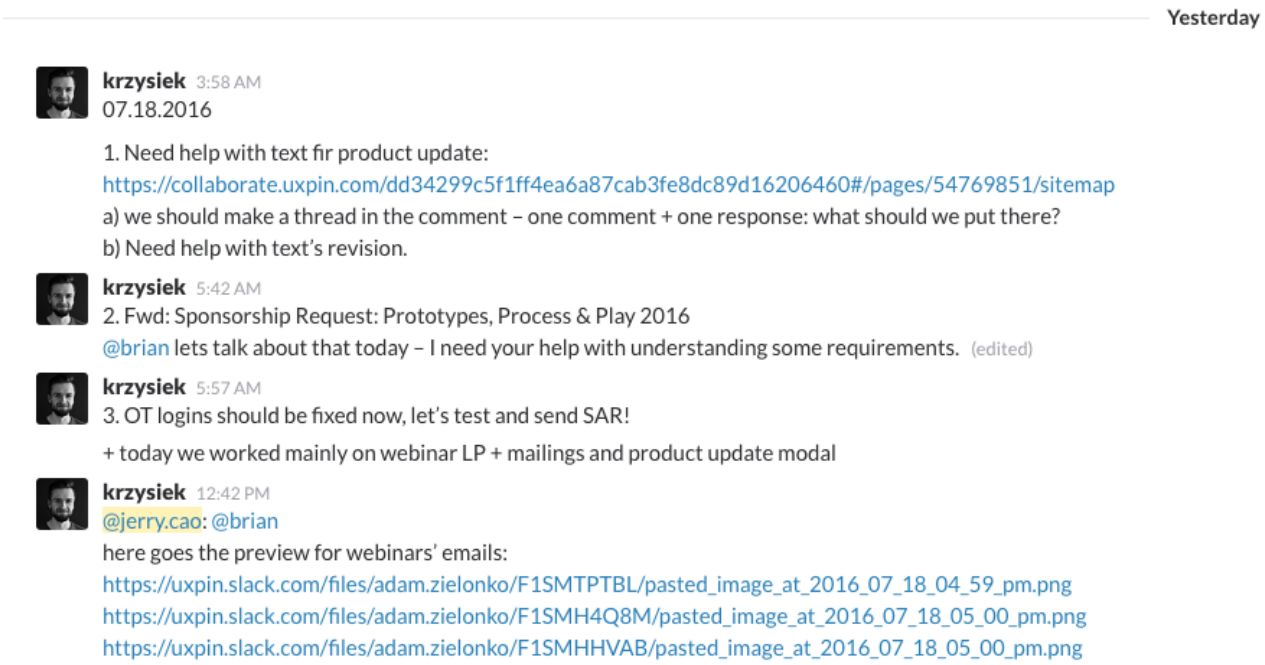
As everyone starts diving into their sprints, we've found it more useful to follow the spirit rather than the letter of agile.

Don't let dogma rule your process.

You don't need to run every stand-up as a daily teleconference. Designers and developers hate meetings. So we run a quick standup in Slack instead. The activity is perceived more as a quick conversation rather than a formality.

Within each squad, our product owners will review:

- What was accomplished yesterday?
- What's on the plate for this week?
- Any blockers for projects?



Since the task is timeboxed for 10 minutes, we save all the details for individual Slack conversations.

If you're working on a large team across different offices (or totally distributed), you need to further adapt the standup format for scale.

Consider appointing key people with leadership potential to speak with team members about the three questions above, then summarize the findings. Instead of a daily standup, you can hold it weekly or twice weekly.

Now, your standup won't morph into an hour-long timesuck.

Unify the Design & Development Workflow

As you build the product, you want to synchronize design and development as much as possible.

Collaboration should never start during the handoff process.

Here's a quick checklist we've created to keep both sides informed through the whole product development lifecycle.

During Design

1. **Discovery**
    - Designers invite developers to attend user interviews.
    - Designers circulate bulleted summary of user interview insights with developers.
    - Designers conduct a 30-minute to 1-hour stakeholder interview with at least one developer. Use Kim Goodwin's excellent questions.

- Designers get developer alignment on technical restraints for the design brief.

2. **Planning**
   - Project manager ensures developers (or at least the development lead) attends the kickoff.
   - Product manager and designer conducts user story mapping with developers to plan epics and sprints. We recommend tools like Craft or TargetProcess for distributed teams.
   - Project manager estimates build time for user stories with developers using tactics like planning poker.
   - Project manager plans design sprints 1-2 sprints ahead of development.
   - Designer verifies with developer if a common framework (Bootstrap, Foundation, etc.) will be used. Adapt grids and elements accordingly.
   - Designer verifies browser support with developers.
   - After each standup meeting, project manager reviews the backlog with designers and developers on the same call.

3. **Prototyping**
   - Designers must use approved assets from the design system.
   - Designer walk through user flows and lo-fi prototypes for developer and product manager feedback.

- Designers may want to consider designing extreme viewports first (smallest and largest) to accurately "bracket" your content. Consider how the design responds to screen sizes slightly smaller or larger than current assumptions.
- Designers incorporate rough copy from content team (not Lorem Ipsum) into the prototype within the first two iterations.
- Designers invite developers and product manager to attend at least one user testing session.
- Designers ensure prototypes account for major interaction states, error states, and transitions between states.
- Designers ensure prototypes account for data extremes (e.g. short and long last names, phone number formats, non-US post codes).
- User researcher circulates all user test recordings with bulleted summary of insights to developers, designers, and product manager.
- Designer collects feedback and approval from developers and product manager at each iteration of the prototype.

4. UI Design

- Designers only use assets from the design system.
- With each iteration, designers rename the project (v.1, v.2, etc.). Do not rename with "Latest" or "Newest". Upload every new version into a shared repository.

- Designers only use assets from the design system.
- Designers make UI decisions that create consistency for the experience and code base.
- Designers get developer buy-in on image formats and sizes.
- Designers account for all major breakpoints on a grid system with guides/overlays.
- Designers use whole font values and leading values (e.g. 15 instead of 15.75) to preserve typographic integrity.
- Designer holds a 30-minute to 1-hour review of final approved mockups alongside the prototype. Walk through project goals, user stories, interactions, states, and failure states with developers and product manager.

During Handoff

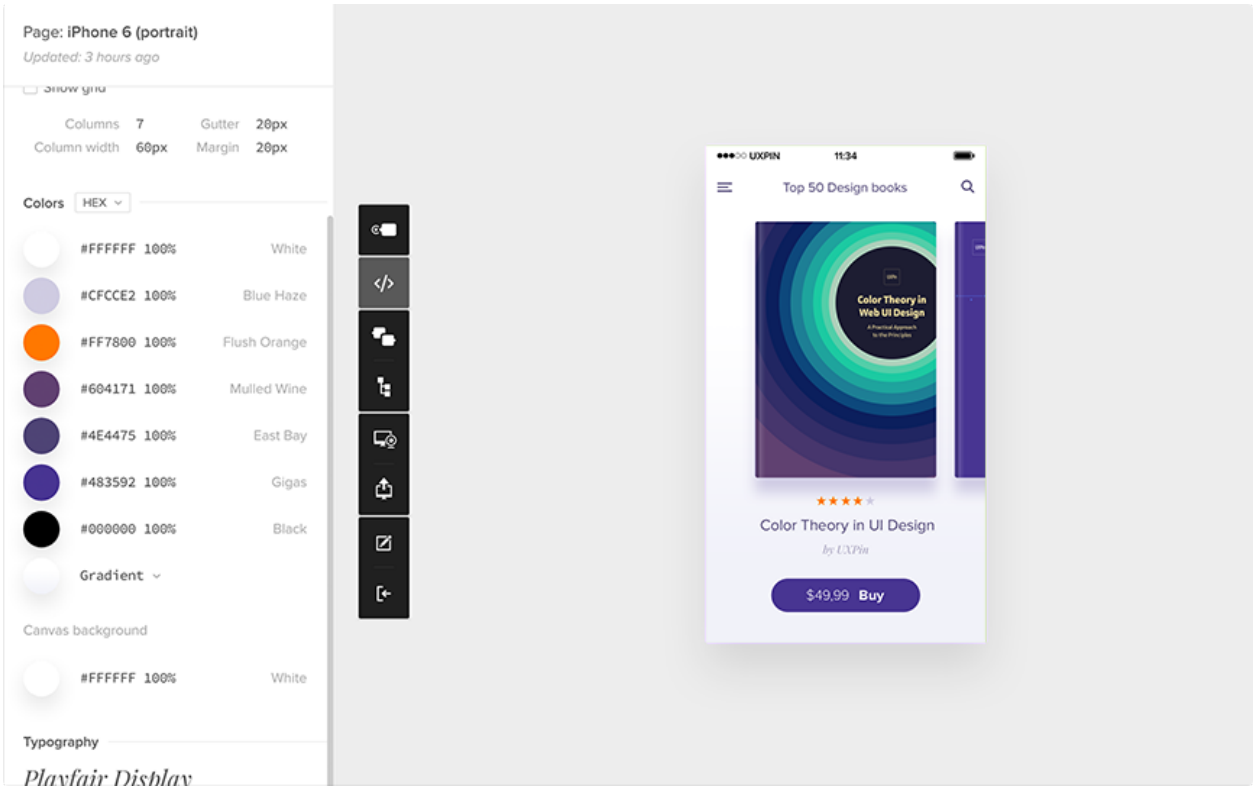
1. Visual Hygiene

- Designers delete all unused layers. Don’t just hide them, since that confuses developers.
- Designers delete all unused guides.
- Designers group and name layers appropriately based on UI modules (navigation, footer, etc.)

- Designers follow a common naming convention with developers as defined in the design system (e.g. a “widget” is not the same as a “module”). Consider using BEM notation.
- Instead of naming artboards with “FINAL” or “LATEST”, designers follow a standard versioning protocol outlined in the design system (v.1, v.2, etc.).
- For easier navigation, designers collapse all layers before handing off to developers.

2. **Assets**
   - Designers include SVGs wherever possible. For raster files, include versions at 2x.

3. **Documentation**
   - Product manager collaborates with designer and developer to check that documentation in the prototype matches what’s in the design system.
   - Designers use an inspection tool to auto-generate specs for developers. Avoid redlining as much as possible.
   - Product manager ensures all documentation stays updated to the final system as it evolves. Developers will refer to documentation to understand depth and breadth of the system, using the final prototype as a reference for acceptable behavior.



Automated documentation make handoffs more precise and collaborative. Simple plugins like Measure (for Sketch) automate redlining. Dedicated tools like Zeplin (for Sketch) adds collaboration on top of automated redlining. More robust collaborative options like our own Spec Mode above (for Sketch and UXPin) automates redlining while assigning code and use cases to interactive elements.

After Handoff

- During the QA process, designers perform “implementation audit” of each build against the final prototype.
- Designers view sprint demos along with product manager.
- Product manager ensures acceptance testing includes UX criteria according to final prototype.

Recommended Tools for Distributed Product Development

Distributed design would not be possible without the right collaboration tools.

In closing, here are the tools we recommend for closing the distance:

File sharing

- **Dropbox** – Large-scale file sharing and syncing.
- **Google Drive** – Free cloud data storage.

Design systems, product design, specs and documentation

- **UXPin** – Full-stack UX design platform. Design systems, prototyping, and developer documentation in one place for the whole team.

Usability testing

- **Usertesting** – For remote moderated and unmoderated usability tests. Tests start at $49 each.
- **Userzoom** – For large-scale moderated and unmoderated user tests. Only available for enterprise teams.

Development

- **Github** – For hosting code assets

Product management

- **ProdPad** – Feature prioritization, roadmapping, and sprint planning.

Project management

- **Planning Poker** – Sprint planning tool free for up to 10 players per session.
- **Target Process** – Visual project management for Agile and Kanban techniques.

Brainstorming

- **Memo Sort** – An online brainstorming and affinity diagramming program.
- **Google Drawing** – Allows for free collaborative sketching.
- **Mural** – Our preferred online UX whiteboard for collaborative brainstorming and concepting.

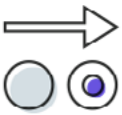
Communication

- **Slack** – Company chat and daily collaboration. Also useful for beta testing by creating dedicated channels.
- **Zoom** – Freemium teleconferencing tool that's more stable than Google Hangouts for intercontinental calls with up to thousands of people.
- **Sqwiggle** – Video chat tool designed specifically for distributed working.

The Full-Stack UX Platform

Your entire UX process in one place

Design:
Create lifelike prototypes quickly with Photoshop and Sketch integration.

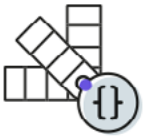


Iterate:
Built-in version control improves efficiency and eliminates confusion.

Document:
Cleanly annotate your designs. Insert custom code snippets that travel with elements.

Collaborate:
Get feedback and co-design on any project anywhere.

Scale:
Automate consistency and documentation with design systems (syncs with Sketch).



Implement:
Auto-generate style guides, assets, and specs for developers.

Try UXPin now