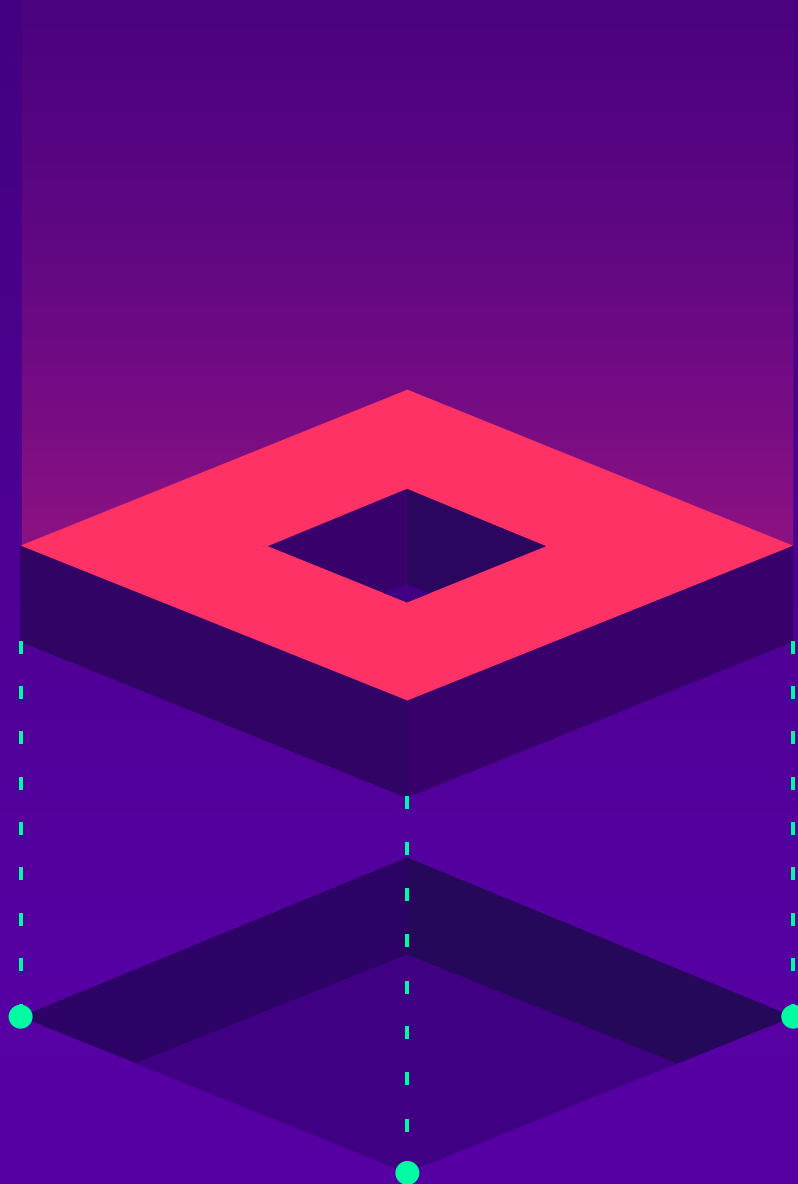


UXPin

Prototyping for Product Managers

Simplify Your Requirements



The UXPin logo consists of the text "UXPin" in a sans-serif font, centered within a thin black rectangular border.

Prototyping for Product Managers

Simplifying Your Requirements

Copyright © 2016 by UXPin Inc.

All rights reserved. No part of this publication text may be uploaded or posted online without the prior written permission of the publisher.

For permission requests, write to the publisher, addressed
“Attention: Permissions Request,” to hello@uxpin.com.

Index

Introduction	5
How Do Prototypes Clarify Your Requirements?	7
Less risk of confusion	7
Flexibility and usability	8
Improves the quality of technical decisions	9
Better stakeholder buy-in	11
Conclusion	12
Before You Prototype: A Quick Checklist	13
Conclusion	17
A Hands-On Prototyping Exercise	18
Conclusion	37
Agile Prototyping Case Study	39
Challenges	40
Solution	40
Result	41

Author



With knowledge of the U.S, European and African markets, Germaine is a Product Manager who uses her 14 years of experience to help companies build products and services that resonate with users. Her varied skillset - Product Management, Writing, UX, Training, QA - allows her to evaluate the needs of each brand, and team to propose solutions that work best for each context.

Introduction

Without prototypes, product managers would need to rely exclusively on user stories to drive the product development cycle.

The problem is that user stories are only brief descriptions of the user needs. They don't say anything about product vision or potential complexities. So you can't really rely on user stories exclusively to validate the design and overall user experience.

In an Agile world, prototypes are a mandatory complement to user stories. They allow product managers to quickly communicate and validate functionality. Because they are visual, they are easily understood by all stakeholders and Agile team members.

In this e-book, we'll help you clarify requirements and minimize risk by explaining:

- The most practical benefits of prototyping.
- The items you should prepare before prototyping.

- How to build the most useful prototype.
- A hands-on prototyping exercise.

I've based all the advice on my nearly 10 years of digital product management experience.

Ready? Let's get started!

Germaine Satia,
Enterprise Product Manager

How Do Prototypes Clarify Your Requirements?

Prototypes are the ideal tool to launch important discussions early in the development process when failing fast won't jeopardize budget or timeline.

Let's look at the key benefits of prototyping and how they clarify requirements.

Less risk of confusion

Lengthy, 50-page requirements are simply not reliable ways of conveying all the subtleties of a digital product. In my experience, readers quickly get bored and even worse, misinterpret what is written.

Prototypes, on the other hand, expand each user story into a tangible part of a system. If user stories express task-level goals, then prototypes help us see the horizontal feature set (breadth) and vertical feature set (depth) required to fulfill each story.

If you've ever created a formal requirements document, you know how much patience and imagination is required.

The cycle of documentation just becomes too expensive. Just look the specs documents one of UXPin's customers used to create.

LookThink's old 30+ page product requirements document before prototyping.

Header

Encounter_73457- Barbara Johnston
 Encounter Status: **Rejected**
 Encounter History (3) and related files

14 - Uneditable fields (3)
 Original File: Professional | Last edit: Not yet modified

STARTED 3 CMS RESPONSE ERRORS
 0 errors changed - 3 remaining
 CMS response errors remain until next CMS response.

9 unvalidated changes
 1 - Ignored fields 1 - Fix later
 You have no changes that require validation.

Validate encounter

View as B37 format | View error report

Find

FILTER:
☐ CMS Response Errors ☐ Validation Warnings ☐ Unvalidated Changes ☐ Ignored ☐ Fix Later

SUBSCRIBER

Subscriber Name	Barbara Johnston
Payer Subscriber Number	123456
Receiver Subscriber Number	55893795A

IDENTIFICATION

System ID	Medical Life Health Plan
Claim Type	Aetna
Patient Control Number	987989554
Medical Record Number	Medical Life Health Plan

DETAILS

Submission Type	Initial
BI Type	13
Frequency	1 - Admit through discharge claim
Total Charge Amount	1780.34

PARTNERS

Trading Partner	Medical Life Health Plan
Subscriber Name	Aetna
Subscriber ID	987989554
Receiver ID	987557547890
Payer Name	Humana
Payer ID	90920

BILLING/PHARMACY PROVIDER

Billing Provider Name	St. Johns Hospital
Billing Provider NPI	08430392

RENDERING/PRESCRIBING PARTNER

Payer Subscriber Number	123456
Receiver Subscriber Number	55893795A

Subscriptions

Subscriber ID	987989554
Receiver ID	987557547890
Payer Name	Humana

DETAILS

Frequency	1 - Admit through discharge claim
Total Charge Amount	1780.34
Total Paid Amount	340.64

CHANGE STATES HERE

NOTE: This is a representation of the **SECOND** functionality ONLY. This does NOT contain the edit feature at this time. The edit feature will stay largely the same as the existing application.

The default view when arriving at the "business view" is the entire simulated CMS-1500 form presented in a more readable user friendly format on the left and the by default, the Error Log showing on the right. Users can jump through the document by errors and the log on the right should highlight and scroll as well.

Previous error Next error

Each field will have a hover state. Fields at error will have an additional hover that reveals the cause of the error.

Hover here for error example

The default view will NOT include empty fields. The user can however display these fields with a checkbox.

☐ Show empty fields

The user can easily filter by errors only in the filter section.

(Click to view the errors)
☐ CMS Response Errors
 (Click to hide the errors)
☒ CMS Response Errors

Other filters are dependent on those items being present in the result set. They should grey if they are not actionable filters e.g. there will not be any unvalidated changes until the user changes a field in the data.

☐ Validation Warnings ☐ Unvalidated Changes ☐ Ignored ☐ Fix Later

User can click to view a consolidated view of the errors in this encounter.
[View error report](#)

Once inside the error report, the user can assign the encounter as well as print, export, email or open in a new window.
[Assign Encounter](#)

Complete the assignment by clicking assign.
[Assign](#)

The fields then show the user assigned.

User can also assign a field from the edit state when editing a field.
 Double click to assign from the edit state...
[Assign](#)

The name will change on the field data.

User can also assign the encounter from outside the error report at the top overview section.
[Assign](#)

Example Action Button
[Assign](#)

LookThink's new product requirements created contextually with UXPin prototypes.

Prototypes, on the other hand, are much faster to create. Instead of reading requirements, people *are experiencing them*. Content is easier to navigate, and the taxonomies are crystal clear. If changes are required, the reviewer can comment directly on the affected part of the UI.

Far more efficient than wading through documentation that requires serious imagination.

Improves the quality of technical decisions

UX decisions are only as good as the implementation. Look deep enough, and you'll see that miscommunication is almost always to blame for poor implementation.


If developers and designers both speak the common language of systems interactions, why not communicate requirements in that medium?

Even a lo-fi prototype can help developers quickly see dependencies between elements and interaction models. An early evaluation might even prompt developers to suggest better solutions the designers never even considered. All before anyone ever opens Photoshop, Sketch, or a code editor.

Outline your core requirements in the PRD, then link to external sources for more information. As customer Autodesk shows below, treat documentation as a knowledge hub rather than paper trail.

Bundle and Education Notification

UX/UI Specifications



Resources

PRD: <https://docs.google.com/document/d/141iA5lU5chllkQFcGclFTzi3h0nk5-pzu6FDlRr2XE/edit>

Zeplin iOS: Bundle and Education Notification - Android
<https://app.zeplin.io/project.html#pid=56d2c13a5cc455621cdcc714&dashboard>

Zeplin Android: Bundle and Education Notification - Android
<https://app.zeplin.io/project.html#pid=56d2c1632234dd9a59761aed&dashboard>

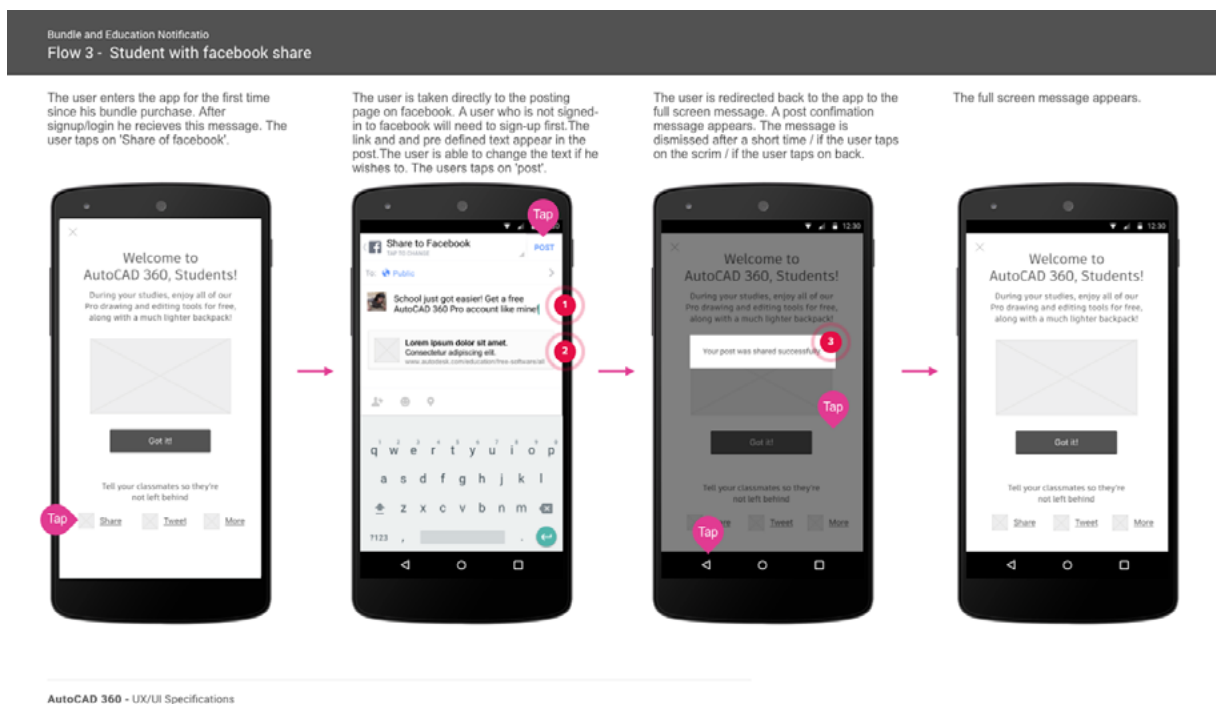
UX/UI Dropbox: UX Team > AutoCAD WS > Design > Mobile > v3.5 > Bundle and Education Notification

Table of Contents

- Flow 1 - Bundle User
- Flow 2 - Student without share
- Flow 3 - Student with Facebook share
- Flow 4 - Student with Twitter share
- Flow 5 - Student with More share
- Flow 6 - Student with share and back
- Flow 7 - Chinese student share
- Flow 8 - Student with share error

AutoCAD 360 / UX/UI Specifications

Simple requirements portal created in [UXPin](#) by Autodesk



Annotated user story flow created in UXPIn by Autodesk.

Better stakeholder buy-in

Stakeholder buy-in is often tricky.

They need the project to fulfill business needs (e.g. increase revenue or NPS score, decrease support call volume, etc). For them, user needs are just a means to the end.

When focused on the bottom line, it's easy for stakeholders to treat requirements as a laundry list. After all, they can't see the consequences of their requests. Why not add another feature (or feature set) if it makes the product more robust?

The problem is that the product vision isn't tangible. Everyone sees the product differently, and therefore reacts differently.

A prototype opens their eyes. The laundry list dissolves, and the product comes into focus. The consequences of additional features are immediately real for everyone from a junior designer to a Product VP.

Conclusion

Agile development requires a great deal of flexibility from all team members. Even with the use of Agile user stories, it's not easy to dive into the fine details around design and user experience.

Therefore it's essential to make prototypes a standard part of the Agile process and documentation. The visual nature of prototypes allows everyone from stakeholders to developers to align their thinking and work toward the same goals.

Before You Prototype: A Quick Checklist

In my experience, it helps to think of prototyping as its own mini-project.

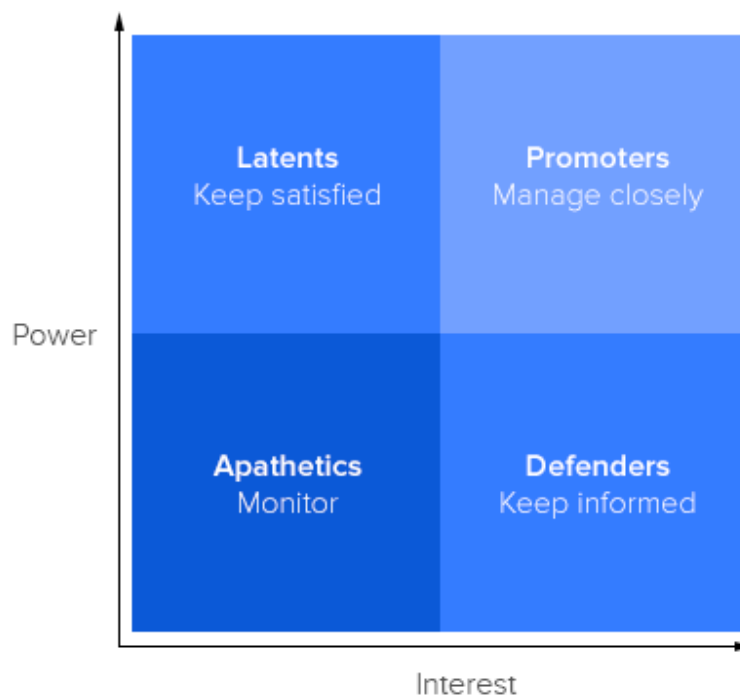
Here's a checklist of 4 questions to always ask yourself before you prototype.

1. Did I Identify All Stakeholders?

Stakeholders frequently come from within the company.

However, I've also worked on a few projects driven by a particular customer. In such cases, don't forget to identify the key stakeholder(s) from the customer's organization.

Once you've identified all stakeholders, prioritize them on the next page matrix.



2. Did I Interview All Stakeholders?

Once all stakeholders are mapped out, set aside time to talk to them individually. Schedule the highest priority stakeholder interviews first (Defenders and Promoters from previous chart).

Your goal is to:

- Understand each person's vision for the project.
- Identify the must-have, non-negotiable requirements for the project.
- Clarify business objectives.
- Note each stakeholder's personal concerns.
- Identify any unavoidable deadlines (such as upcoming conferences or tradeshow).

For the most insightful questions, use Kim Goodwin's excellent guide:

1. [General Stakeholder Interview](#)
2. [Marketing Stakeholder Interview](#)
3. [Engineering Stakeholder Interview](#)
4. [Sales Stakeholder Interview](#)
5. [Executive and SME Stakeholder Interview](#)

Remember that the clearer the business goals, the less risk of scope creep. "Increase revenue" is too broad, but "Increase revenue 15% within 3 months without increasing churn" focuses everyone's thinking.

3. Did I Talk With Customer-Facing Teams?

When it comes to understanding customer needs, the customer support team is usually one of the richest sources of information. They're right on the front lines of dealing with customer pain points.

Talk to them and other customer-facing teams (such as sales and account management) about recurring requests or complaints.

You'll uncover surprising user needs that business stakeholders either don't know or are unwilling to admit.

4. Do I Know the User and Buyer?

Can you assign a name, age, and profession to your end user? Can you articulate their problems, behaviors, and goals?

To anchor your scope, include all the answers to the above in the [user personas](#).

Think ahead to how users will purchase your product. Are you building software for children, which means the parent makes the purchase decision? Or are you building software for enterprise collaboration, which means the CTO makes the purchase decision?

If your user and buyer are two different people, create separate buyer personas.

For example, if your product is for collaborative work in an enterprise environment, then your prototype should account for functionality around tracking changes, modifying user access rights and the use of private encryption keys. End-users might not care that much, but these security-based features are critical for the technical buyer.

When analyzing users and buyers, consider the following tools and tactics:

Tools

- [Customer Journey Maps](#)
- [Empathy Maps](#)

- [Lean End-User Personas](#)
- [Buyer Personas](#)

Tactics

- [Contextual Inquiries](#)
- [User Interviews](#)

Conclusion

Once you've gathered requirements from multiple sources, you're ready to start culling your user stories through prototyping.

In the next section, we'll dive into a hands-on exercise to show you how to prototype the riskiest user stories.

A Hands-On Prototyping Exercise

Before we dive into the nitty gritty, understand that the goal of a prototype isn't to polish all design ideas.

If you're working on a team with a UI designer, that person will fine-tune the final design based on the prototype feedback. Your goal is to reveal hidden dependencies and overlooked requirements. That way, the team doesn't uncover nasty surprises two sprints later.

Let's explore how to go from a list of user stories to a focused prototype that reveals unforeseen risk.

1. List all user stories

Let's say you're building a mobile app that allows users to print their phone pictures in a photo book. For this type of app, here are some of the primary user stories:

- As a user I can create a new account
- As a user I can select pictures from my phone gallery
- As a user I can select my photo book size

- As a user I can apply visual effects to my pictures
- As a user I can pay for my book with a credit card
- As a user I can access my order history
- As a user I can access the Terms & Conditions and Privacy Policy

At this stage, don't worry about the fine details of each user story. You simply want to list the user actions so that you can map out the **user flow** and identify any gaps.

2. Prioritize user stories by risk

In my experience, little benefit comes from prototyping a user story like this one:

As a user I can access the Terms & Conditions and Privacy Policy

This is a very low-risk user story because the screens involved are static and read-only.

For prototyping, prioritize user stories involving user-submitted data and processing of user submitted data. This is where the most risk often lies and a clear user experience is most required.

For the photo book example, let's examine the potential risks of each user story.

ID	User Story	Comments	Risk Level
1.	As a user I can create a new account	Accounts are necessary to keep each user's data separate and private	High
2.	As a user I can select pictures from my phone gallery	Since this is a mobile app, it's essential that users can connect to and access the phone's image gallery	High
3.	As a user I can select my photo book size	The book size selected on this screen should be the same book size used at the printer. There's a chance that book size codes could get mixed up, resulting in the customer receiving the wrong size	Medium
4.	As a user I can pay for my book with a credit card	Users will be inputting personal and banking information. We need to decide if the info is stored for future use. If so, then technical implementation will need to address security and data encryption needs.	High
5.	As a user I can access my order history	This is a read-only screen. Only potential risk is errors related to order cost or book type.	Medium
6.	As a user I can access the Terms & Conditions and Privacy Policy for the app	No user data is being received or manipulated in these screens. Read only content.	Low

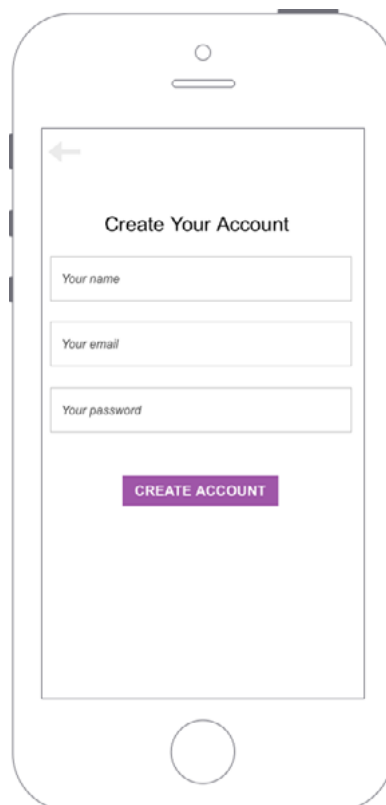
3. Prototype the riskiest user stories

Now that you've identified the high risk user stories, it's time to visualize them through your prototype.

When prototyping requirements, use low or medium fidelity. You want designers and developers to better understand how the specs might manifest themselves as interaction models and content structures. Don't get ahead of yourself by prescribing branding and visual design.

Let's take a look at the prototypes for user stories 1, 2, 3 and 4. I'll use [UXPin](#) since I use it most often for my projects, but the best practices apply to any tool.

User Story 1: As a user I can create a new account



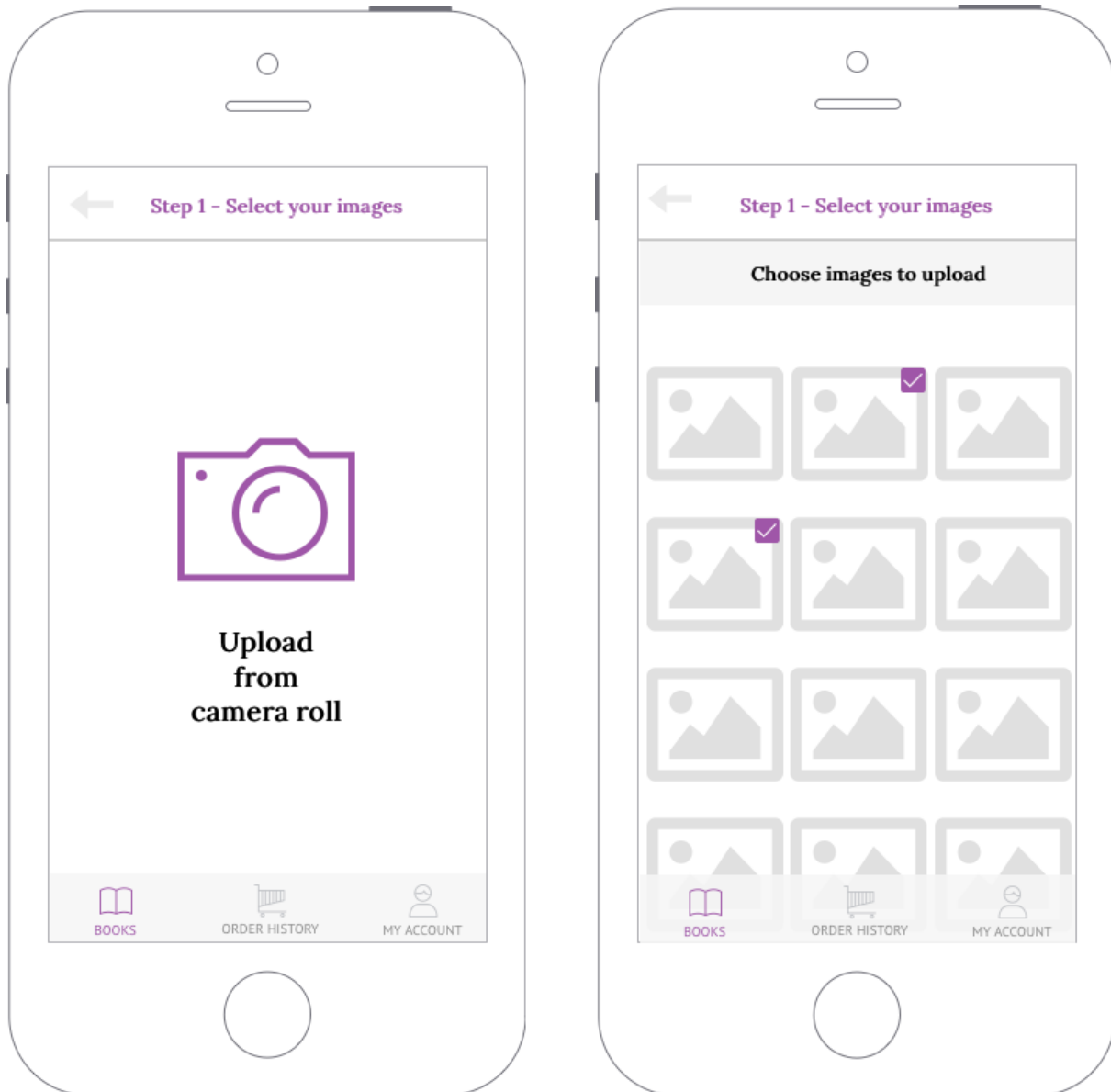
The first screen for the Photo Book App highlights the information that users must provide in order to create an account. This is a simple but critical screen to the technical team because it immediately clarifies the type of data stored in the backend:

- Name (will contain only letters)
- Email (will contain letters and numbers)
- Password (may contain letters and numbers, depending on the organization's security rules)

Based on the above requirements, stakeholders will likely ask about the rules for passwords. For example, should they contain only letters, letters and numbers, letters and special characters?

By presenting a prototype, you can discuss the best rules for secure passwords. Without a prototype, developers wouldn't know the type of information stored and visual designers wouldn't have any guidelines about the quantity of displayed information.

User Story 2: As a user I can select pictures from my phone gallery



The prototype for User Story 2 involves two screens that must appear in a particular order:

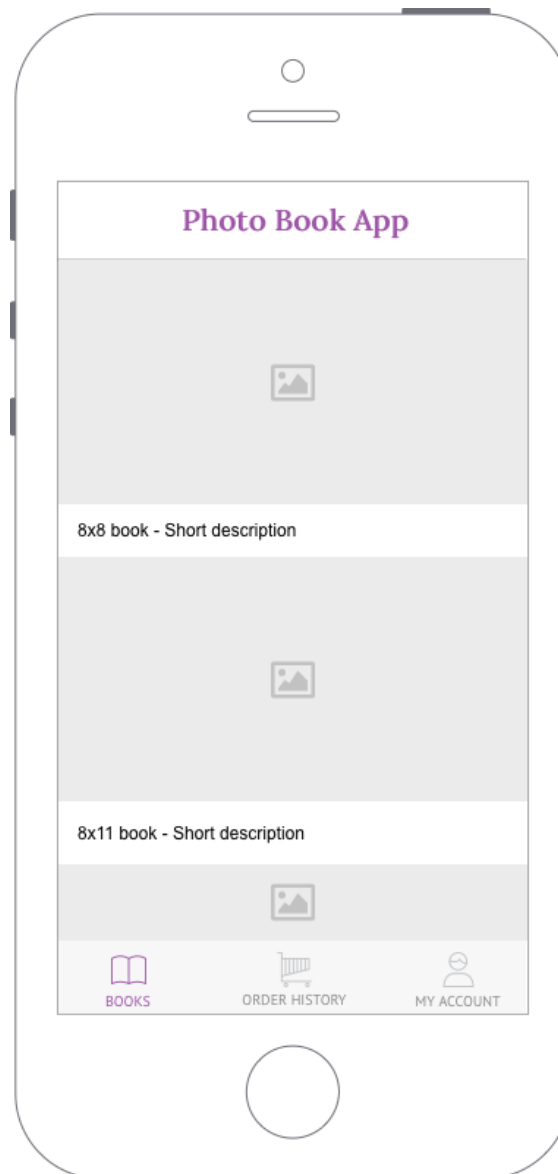
- User presses the camera icon in the first screen
- The second screen displays all the images stored in the user's phone

In addition, the prototype showcases another important aspect of the user experience: ensuring users can easily see their selected images. In the prototype, this functional need is represented by the checkboxes placed in the top, right corner of each image.

What's important to note here is that the app must provide feedback to users about selected images. While the prototype uses a checkbox, the visual designer may choose another option that works better for the screen size, brand, and overall user flow. For example, a transparent overlay could be placed over each selected image, or the color of each image border could change to indicate that it is selected. Luckily, you can now quickly discuss and refine these items as a result of experiencing the prototype early on.

The technical team also benefits from this prototype because they can now see the rough number of images that need to be instantly uploaded from local storage into the app.

User Story 3: As a user I can select my photo book size



We prototyped User Story 3, even though it's a Medium Risk story.

While you should always prioritize High Risk stories when prototyping, there are cases where it makes sense to prototype a Medium or Low Risk screen.

For our Photo Book App, once users select their photos, the above screen for User Story 2 acts as the transition point towards check-

out. We can't avoid any mis-steps in such a key moment when users are selecting the product.

While it's nowhere near perfect, we've at least created a tangible reference for designers, developers, and marketers. The team can now all discuss:

- How might book dimensions, price and short descriptions accompany images of the photo books?
- How might we help users preview their selected images in the final photo book size?
- Should the book print the photos in the book in the order they were uploaded? Or do we allow users to reorder the pages?
- Do we include an additional step that allows users to select a cover image? Or do we randomly assign one from the images they uploaded?

User Story 4: As a user I can pay for my book with a credit card

← 8x8 Book

Step N of x - Payment Information

Name (as it appears on card)

Card number

Expiration Date

3 digit code

☐ Save for future purchases

DONE

BOOKS ORDER HISTORY MY ACCOUNT

The prototype for User Story 4 showcases the basic payment information we need to collect. This allows everyone to discuss the rules for each field. For example:

- How many numbers can the user enter in the credit card field? Do we accept all card types (Visa, Mastercard, American Express)? American Express cards have 15 numbers while Visa

and Mastercard have 16 numbers. We must account for these variances.

- Should we consider adding Paypal as a payment option? If so, how do we prioritize that integration for developers in upcoming sprints?
- When fields are left empty or invalid data is submitted, how does the system respond?

This prototype also highlights a very important functional requirement: saving the user's payment information for future use.

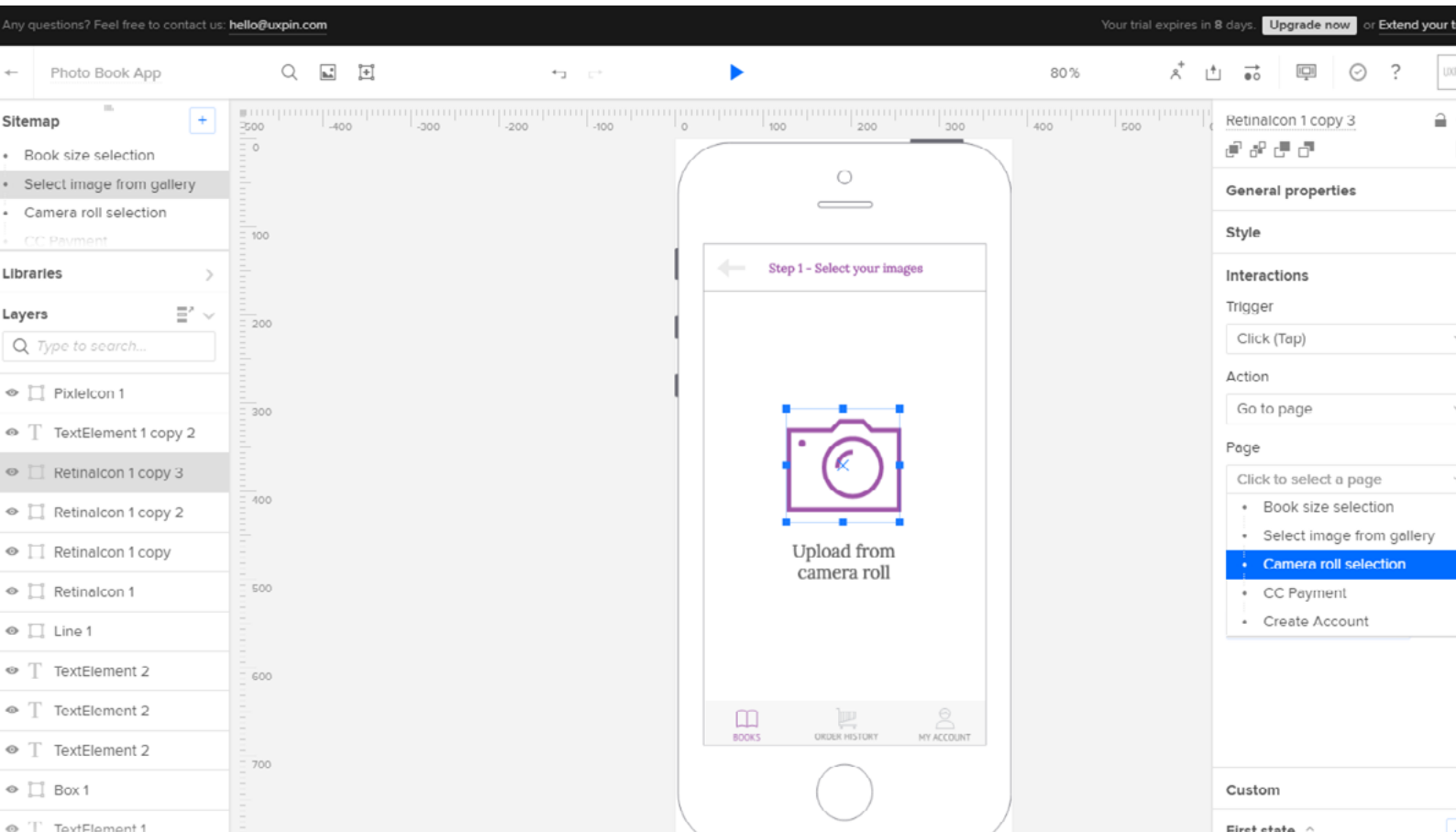
While this is just one little checkbox in the UI, the technical implications are quite serious. We'll need to research legal requirements for storing personal information. In addition, the technical team must consider what might happen if servers suffer an attack or virus. Would customer data become publicly available?

At each step of your prototyping exercise, focus on creating screens that clearly communicate the desired functionality and actions. By using built-in components like buttons, icons, image placeholders and phone frames, you create a user friendly prototype that helps everyone visualize the final product and associated risks.

Now that we're done building all the screens, we would add interactions to all the key content:

- Calls to action leading to new screens
- Icons and images that expand or open new screens
- Form fields that react to user input

In [UXPin](#), we'd simply click on the desired element and then select the appropriate interaction and end-state:

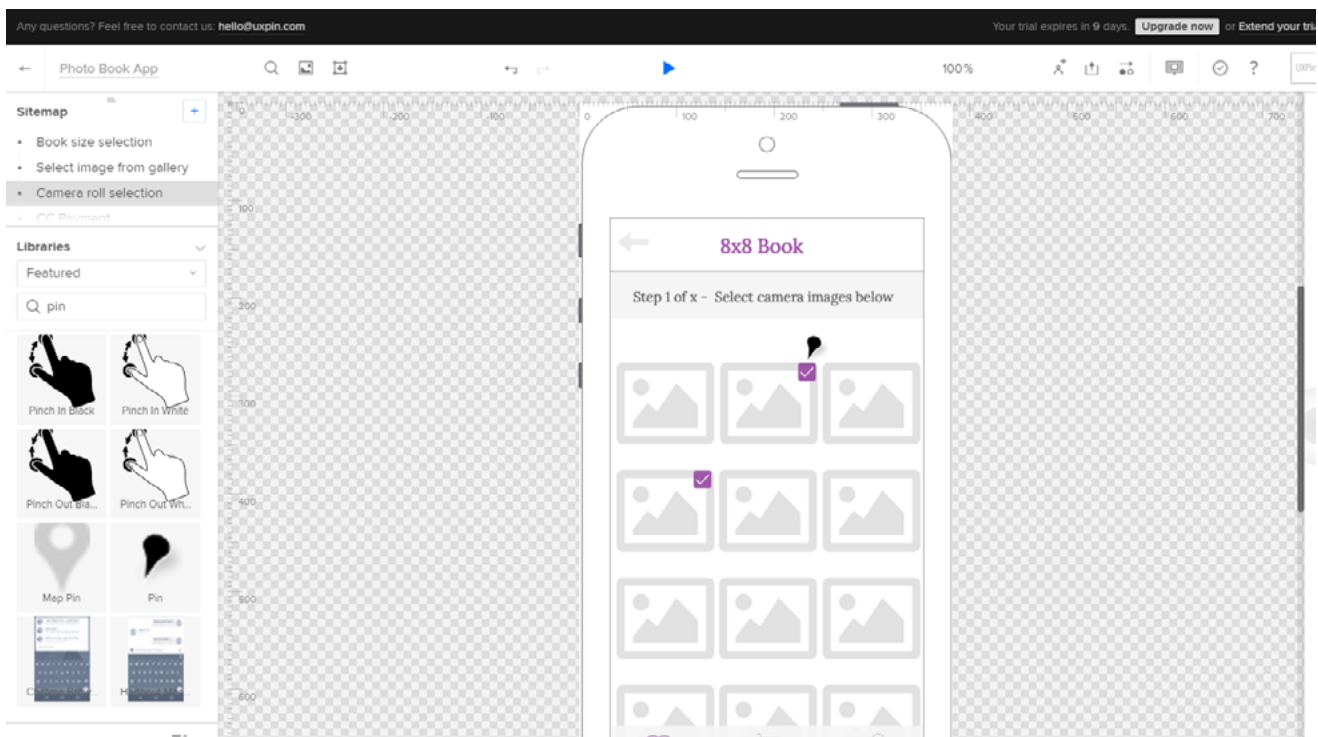


Next, we'll annotate and share our interactive prototype for feedback.

4. Add comments and questions to your prototype

As you build your prototypes, you'll likely have open questions that can only be answered by developers, designers, or management stakeholders.

To keep track of these questions, add them as annotations to your prototype. In UXPin, a pin icon signifies an annotation.



These pins will also display for every stakeholder with whom you share the prototype.

Usually, it helps to annotate:

- **Regional restrictions:** For example, prices in the U.S. use a period (.) to separate the whole number and decimal portions. For example, \$25.00. However, in the Euro zone it's common to use a comma (,) instead. For example, €25,00. Annotate these

types of nuances so that it's clear to the technical team that this is non-negotiable.

- **Nice-to-have functionality:** If some requirements can be added on in a subsequent version, point this out in your prototype. In the Photo Book App, the option to save the user's payment information isn't a must-have for the initial release. It's a convenience that can be added later once you've released the product. You'll need more time to work on implementing the correct data encryption and security to support that functionality.
- **Items that might be costly:** If you're worried about a particularly functionality being too costly to implement, then immediately highlight your concern in the prototype. Use the prototype as an opportunity to discuss the effort needed to implement the items that you find most worrisome. Then, adapt your final requirements based on the feedback you receive.

5. Share the prototype for consolidated feedback

When it comes to reviewing feedback, you can set up a group meeting with all stakeholders and present the screens.

However, a better option is to provide some lead time during which all stakeholders can review the screens in your absence. This is where I've found the share option in [UXPin](#) most useful.

It's possible to export all the screens as a zip file, PDF, or HTML for stakeholders. But in my experience, I find it simplest to just send a live preview link (via email or SMS).

Share

×

URL

E-mail

SMS

QR Code

Export

Copy and share this link

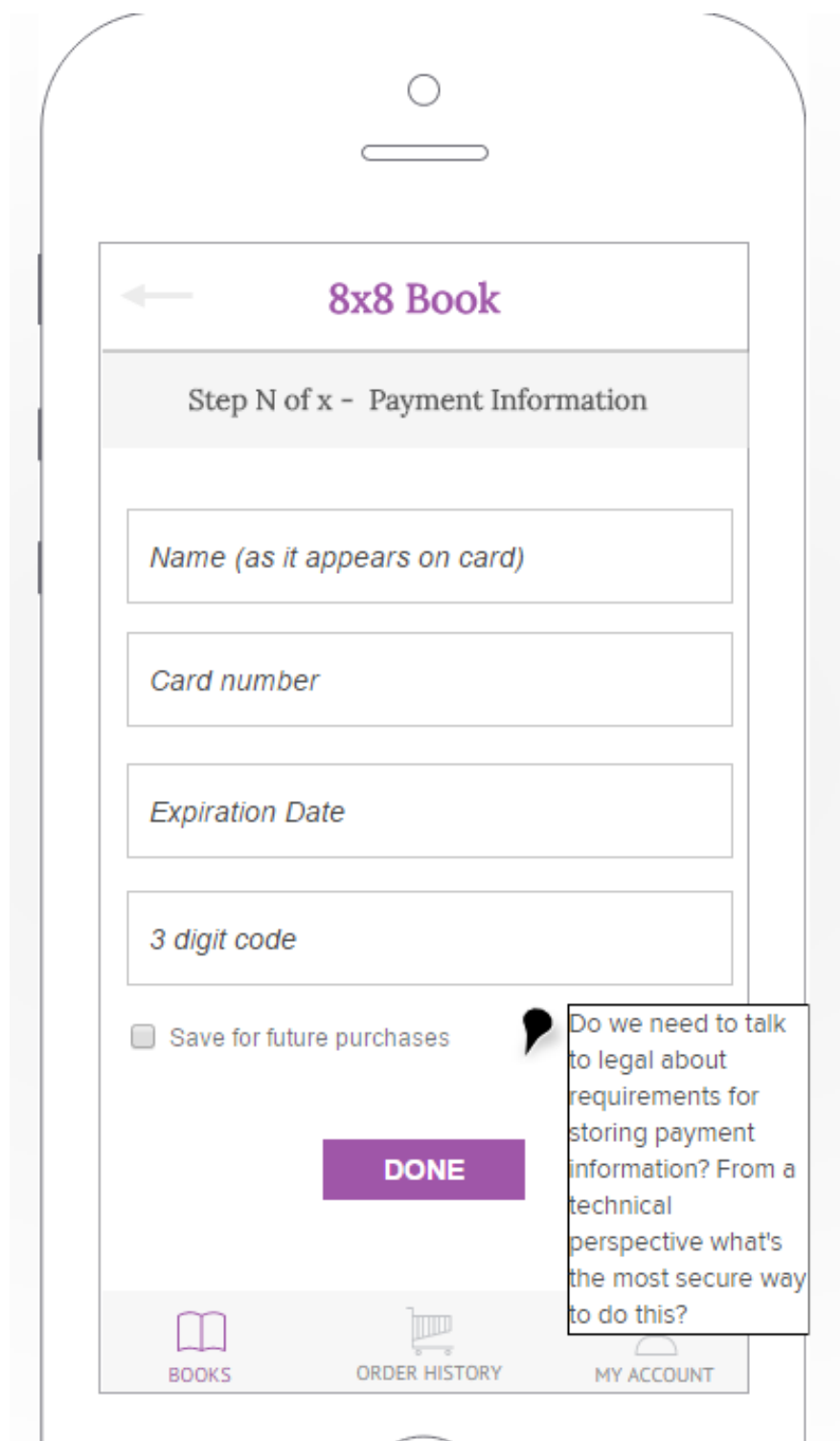
☒ Show comments
☒ Show sitemap

<https://collaborate.uxpin.com/54c0aa9e7e8e05fe6ce43d29625bfd3882b68304#/pages/53684234>

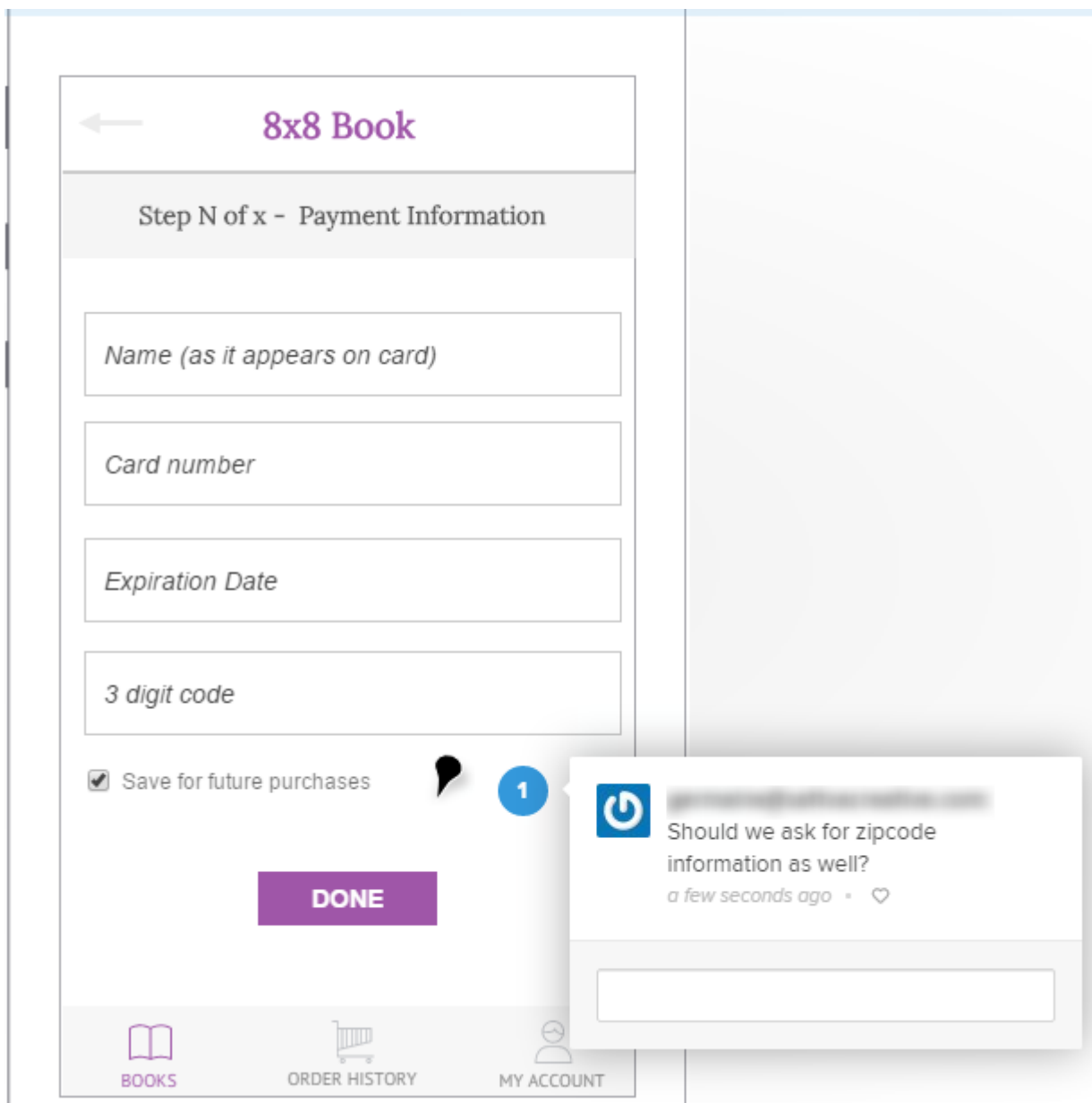
☐ Require password to view this prototype

Sharing a link to your prototype provides the following benefits:

1. Stakeholders can interact and evaluate the prototypes without feeling rushed. The feedback is more thorough and less knee-jerk.
2. Stakeholders can evaluate the prototypes with minimal bias. In a group meeting, the PM will usually set the context for the screens. However, outside of the confines of a meeting, the screens need to speak for themselves. If the basic functionality is not clear as presented on the prototype, this quickly becomes evident to the stakeholders.
3. When you share the link to a prototype, stakeholders can re-view all the comments and questions. Stakeholders can hover over any pin to view its details as shown below. For example, user story 4, which deals with storing payment information. You can leave questions regarding the legal and technical requirements for this functionality.



While reviewing the prototypes, stakeholders can then add comments directly on top of the design.



You can see how it's much easier to collect and review feedback on prototypes, regardless of whether everyone involved is co-located or remotely based.

4. For a more complex project, it's a good idea to review feedback in a group setting. This is an opportunity to focus on the feedback that requires buy-in from other stakeholders.

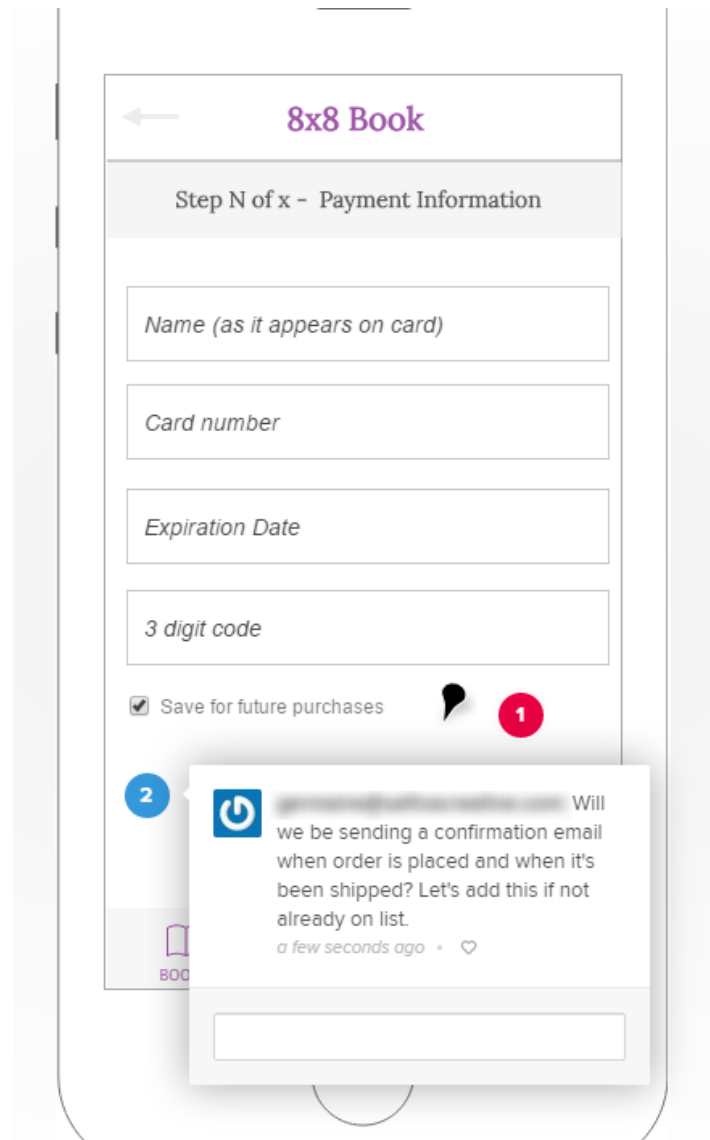
Here's a few tips for actionable feedback:

- When you send your team the design, ask for feedback regarding these items: product vision, feature completeness, and flow between pages. Explain that they shouldn't obsess over the UI design since that may change completely later.
- When responding to comments, ask why. Frame the question in useful and non-condescending ways. Try something like: "That's an interesting perspective. Have you seen any data that brought you to that conclusion or is it more of a gut instinct?" If someone says it's a gut instinct, reframe their situation from a user's perspective: "While that makes sense, as a user, if I wanted to accomplish [user story], I'd most likely need [actions] to happen because of [user & market research]."
- Tease out the core problem behind the feedback with the [3 question rule](#).

6. Analyze feedback, update prototypes and user stories

Once you've analyzed all your consolidated feedback, it's time to update the prototype and user stories.

For our Photo Book App example, the review process brought up an important question that we missed in the original user stories: the need to send confirmation emails when an order is placed and after it's shipped.



For any ecommerce platform, transactional emails are essential for reassuring customers that their order is on its way. Without this, the user experience for the Photo Book App is seriously degraded. Therefore, we need to update our user stories.

As a user I will receive an email when I submit my order.

As a user I will receive an email when my order has been shipped.

In this case, a prototype of the email is also warranted to ensure we account for all legal requirements for receipts and invoices. Failure to respect legal requirements for receipts can lead to se-

rious consequences for the business down the road. We would therefore consider this new user story as High Risk.

Overall, the Photo Book App example we explored is fairly simple. But you can imagine how the more complex your product, the more user stories you might otherwise miss unless you prototype.

Each time you update the prototype, reshare it with stakeholders to validate the changes are satisfactory. After the first revision, designers and developers should see enough of your vision to create the first formal build to [test with users](#).

And as we all know, the earlier we revise and validate the vision, the less costs and risk we accumulate.

Conclusion

Prototypes should always be part of the specification process. In my past projects, they've made it much easier to notice gaps and redundancies in the user flow.

Furthermore, prototypes are a great time-saver for product teams. You no longer need to rely on lengthy specifications documents to get your point across. You can build a quick prototype, annotate it, and share it with your team members to get feedback.

As part of an Agile process, prototyping helps everyone unpack and prioritize requirements before each sprint. At each point, we're able to whittle down requirements into a focused vision.

When it's all said and done, we end up with a more airtight product that stands a better chance of beating the competition.

If you found this guide useful, check out the case study in the next section.

Agile Prototyping Case Study

Closing the Gap Between Design & Development

Based in Seattle, [LiquidPlanner](#) offers a predictive project management platform to thousands of companies worldwide. Its enterprise customers include LinkedIn, Sumo Logic, and Redapt.

In this customer success story, you'll see how UX Designer Edward Nguyen and his team used [UXPin](#) to deliver better products faster.

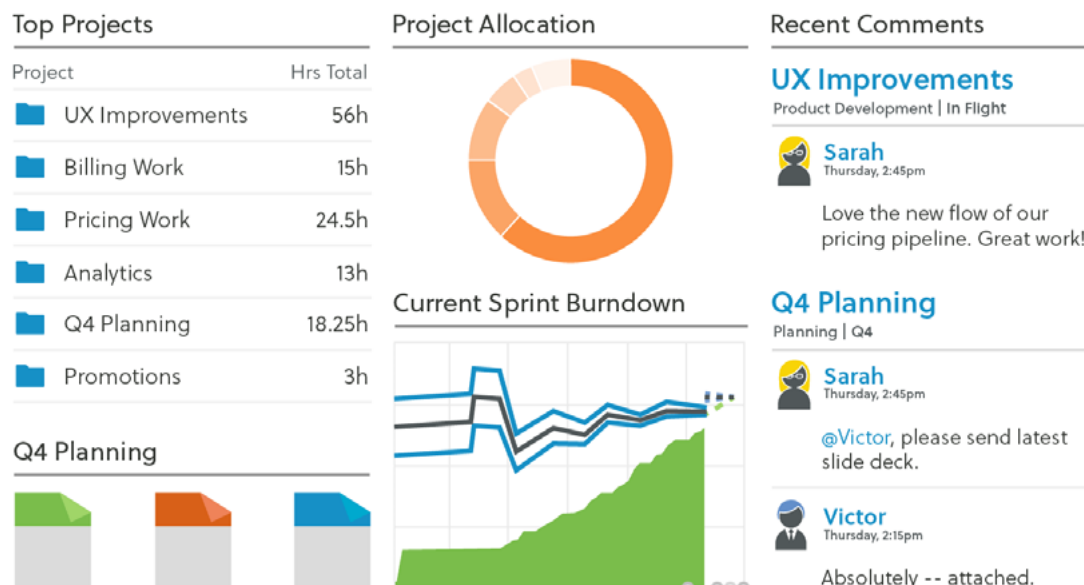


Photo credit: [LiquidPlanner](#)

Challenges

LiquidPlanner's product development process was losing efficiency when shipping on tight timelines:

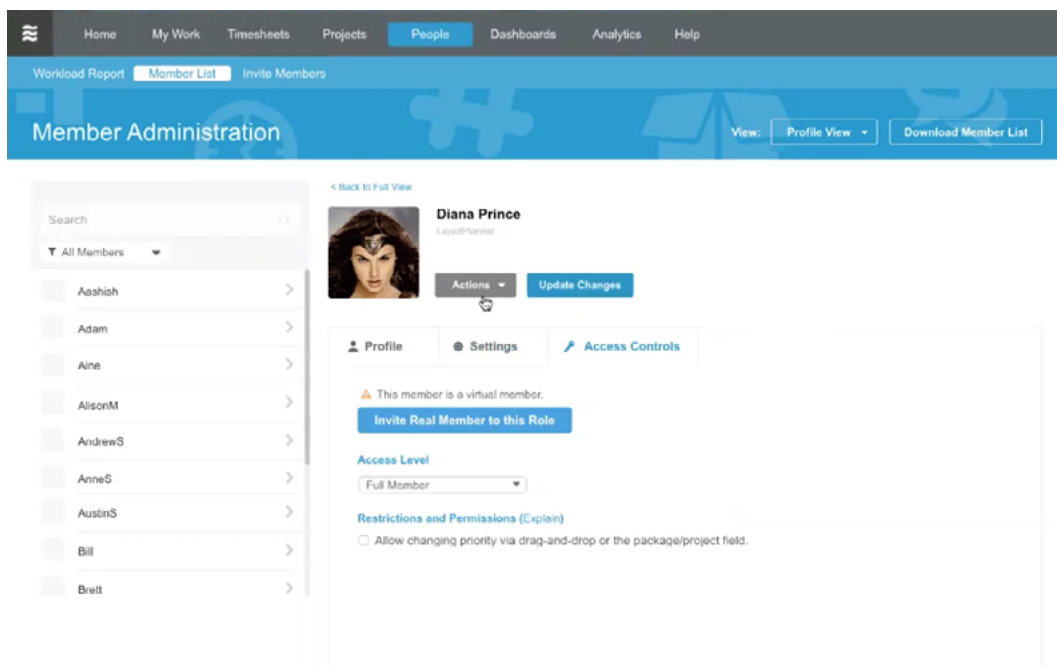
- **Inefficient development costs.** For each detailed prototype demanded by enterprise use cases, the design team sometimes required 4-8 hours of senior developer time.
- **Inefficient designer costs.** If developers weren't available, the design team might need up to 8-10 hours to code each prototype themselves.
- **Inaccurate design tools.** ZURB Notable couldn't support the interactions demanded by enterprise products. Axure lacked the collaboration demanded by LiquidPlanner's Agile process.
- **Timelines slowed by documentation.** Lack of efficient prototyping required the design team to spend up to 8-10 hours before a sprint to work on feature specification with product managers.

"UXPin is a requirement for us. Notable, Adobe XD, Omni-graffle – none of those tools work for us." Edward says. "You absolutely need UXPin's pattern libraries and interactions to validate complex user scenarios quickly."

Solution

To bring the right ideas to market faster, LiquidPlanner turned to [UXPin](#):

- **Accurate testing of complex interactions.** “The new engine and interface are awesome,” Edward says. “A user even mistook one of my hi-fi prototypes as the real deal, telling me to thank our dev team.”
- **Minimal documentation for acceptance criteria.** “QA now uses the prototype as acceptance criteria since they’re so real,” Edward says. “Before they’d either require heavy documentation or just test without criteria.”
- **Unlocks new solutions for product problems.** Because designers aren’t forced to code designs, they can prototype and test more ideas for a faster path to certainty.



With UXPin, LiquidPlanner created a prototype so powerful that users thought was already fully developed.

“It’s amazing to confidently tell my team that I can validate a design in a few days” Edward says. “I can prototype on Monday, test it Tuesday and Wednesday, and show results on Thursday.”

Results

With efficient collaborative design in [UXPin](#), the team can achieve certainty with greater confidence and less cost:

Company Results:

- **Frees up UX development resources.** The design team can create accurate prototypes to test ideas with users without any code.
- **Saves development costs.** “I’m no longer stressed about coding,” Edward says. Prototypes that previously required hours to code now only take minutes.”
- **Speeds up design by 50%.** Designers used to require 4-7 days with a developer (or more) to go from idea to tested prototype. Now they can confidently test ideas with users in 1-3 days.
- **Cut down documentation costs.** “When I showed two developers a 15-20 page spec document alongside a prototype, they said they didn’t even need the documentation.” Edward says. “They preferred the prototype as the specs. We don’t need a ton of documentation anymore.”

Product Results:

- **Launched a new enterprise feature in 4 months.** The team shipped a new dashboard feature prototyped in UXPin in less time (January to April 2016).
- **Increased speed of adoption.** Of the 17,000 dashboards ever created, 10% were created 2 weeks after launch.
- **Increased user adoption.** 75% of new dashboards are now created with the feature prototyped in UXPin. A majority of high-revenue enterprise customers also enjoy the new feature, as it facilitates their complex projects.

Edward Nguyen
UX Designer at LiquidPlanner



“UXPin gives our developers enough confidence to build our designs straight in code,” Edward says. “If we had to code every prototype and they didn’t test well, I can only imagine the waste of time and money.”



UXPin

- ✓ Create and collaborate with your entire team in one place
- ✓ Get real time project updates with our Slack integration
- ✓ Go from lo-fi to hi-fi in a single tool
- ✓ Import files from Photoshop and Sketch

Start using it now!

www.uxpin.com