[whoisjuan]

Juan J. Ramirez

UX Designer • Product Hacker

I'm a passionate UX Designer, Product Hacker and Technology Obsessive who loves to create ingenious software and daring interaction concepts.

www.whoisjuan.me

LIFX - Chrome Extension

Year: 2016

Client / Company: Independent project using LIFX API.

Role(s): UX Designer, Developer.

Contributions: General Feature Concept, Main User Flow, Wireframes, Front-End Code (HTML, CSS, JS), Backend Logic (JS, Ajax, Chrome Dev.)

Final Result / Deliverable: A fully functional Chrome Extension that allows users to control their LIFX lights from their Chrome Browser. More than 700 active users.

Problem

LIFX is slowly becoming one of the leaders in the connected lights segment. Its quality has been widely documented by consumers and journalists. Most of them agree that LIFX exceeds the performance and quality of other competing products like the Philips Hue.

However, one common complaint among customers is that LIFX lights software still doesn't match the quality of their hardware.

One of their problems is the lack of control clients for different platforms. Since LIFX has an open RESTful API, I decided to tackle this problem and create the first Chrome Extension for LIFX (one of the most requested platforms and one that could also help users who desire native OS solutions to control their lights).

How to design and create a lean multi-purpose light control system for LIFX, that can live in a browser as an extension and can help users to achieve the level of control they get from the official mobile apps?

Process

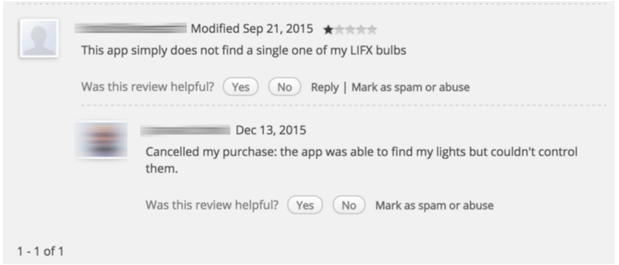
The whole motivation to create this Chrome Extension came from my own frustration. This frustration was mainly related with the fact that LIFX app offering didn't having two things I considered vital to have a reasonable User Experience with my lights:

- The ability of controlling the lights from my computer. As a heavy laptop/desktop user this is a no brainer. Being able to control the lights from my work station is far more convenient that unlocking my phone and going through all the steps required to do it from the app. Context switching is an expensive UX sacrifice, and having changing devices has a very high context switching cost.

- LIFX does a great job at providing mechanics that allow users to achieve different light states and save those states as scenes. However the hierarchy of their app is excessively plain. For users this can be confusing since it's hard to draw a line between setting up new scenes or activating previously created scenes. Even though their UI tries to be clear about this, as a user I find myself more times playing with new states than activating previous ones. This also has an expensive impact on the final user experience.

I knew that my frustration wasn't exclusive to my persona, so I did a small validation exercise to determine the size of the problem and get a clear idea on where should I put my design and development efforts.

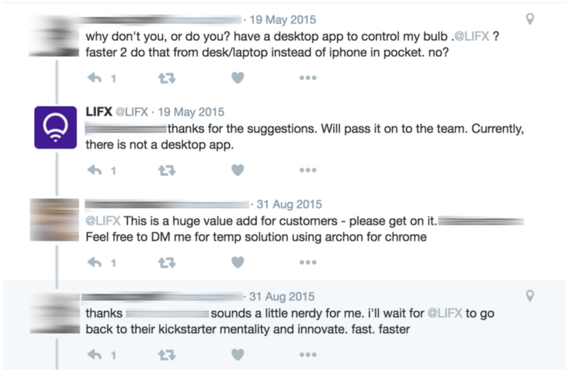
The first thing I did was to check if there were competing solutions. I did find a couple but I didn't have to dig further to find out that these solutions simply didn't work for LIFX users. Here are some of the reviews I found on an extension that claimed to be a LIFX compatible control:



[whoisjuan]

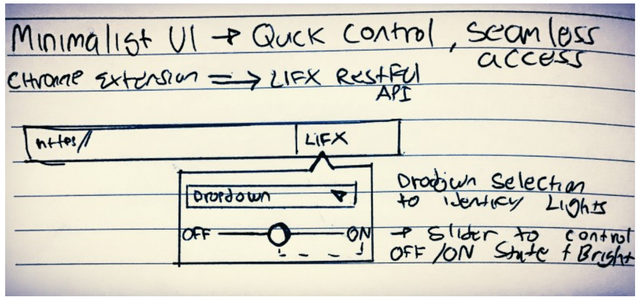
LIFX - Chrome Extension

I also gauge the interest for a Chrome Extension by doing a social media search. After a couple of minutes I was able to find requests like this one:



It was clear that users wanted a desktop solution to control their lights. A Chrome Extension was the perfect fit since it could be a OS agnostic solution and provide the value that many users were looking for.

After doing this validation I started working on some initial design concepts. I started with a simple heuristic that and sticked to that heuristic through all the design and development process. The heuristic I chose as the main driver of the process was: Minimalism.



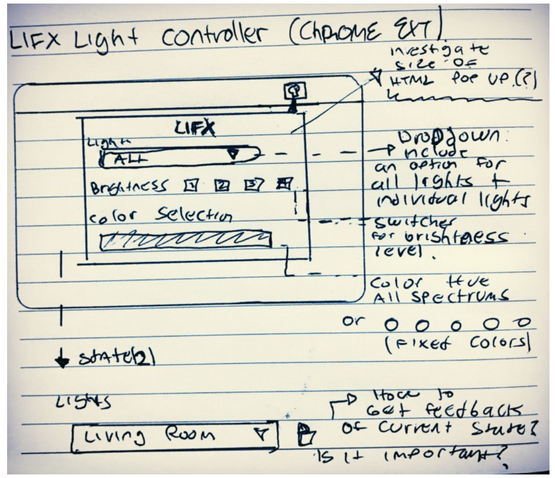
I chose Minimalism as the main driving heuristic for this project because it was syntactically what I would expect from any software that claims to be a good user experience to control connected lights. Minimalism in the context of this project means:

Fewer and less compromising choices.

A leaner and simpler user interface.

An unambiguous user goal and clear path for that goal.

Having this clear I started designing potential solutions that could exist in the constraints of a Chrome Extension dialogue. Here are some of the initial sketches and idea deliberation that encompass my design reasoning:



After exploring more options I decided that I wanted to started with a simple approach that would work on the premise that any user could control their lights in a simple way by following a simple path:

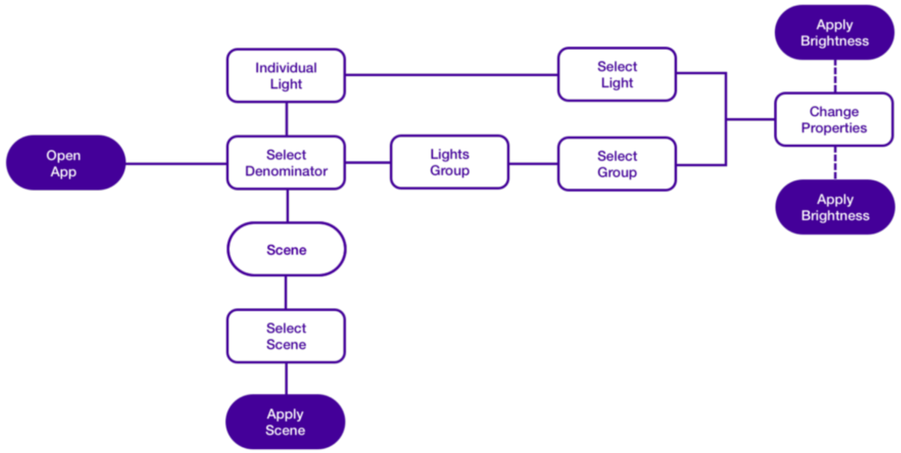
Choosing a Light/Group of Lights > Changing Brightness > Changing Color.

[whoisjuan]

LIFX - Chrome Extension

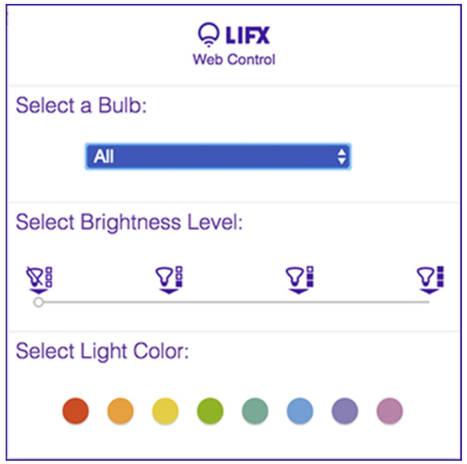
As a heavy LIFX user I had the confidence that most users asking for a desktop version of the LIFX app, were looking for a simple solution that would allow them to simply tweak the brightness (or turning off the lights) and maybe changing the color.

I used some diagramming and charting artifacts to map an flow architecture that could help me to visualize different angles that achieved the minimal desired path I established. I also explored alternative paths to apply Scenes, which are the built-in LIFX property aggregators. Here is one of the final flow artifacts I used to determine the extension layout:



Although I wasn't sure about what the dimension of the options appropriate to suffice most use cases, I stuck with my main heuristic and put together a design that gave users just a few brightness and color options.

Here is the main value proposal and design as explained in the promotion website of the extension.



LIFX Web Control is a Minimalist Light Control that lives in your Chrome Browser.

- Individual Light Selection
- 8 Curated Color Options
- 4 Brightness Options

The initial design I actually implemented and released followed the path I described before. The idea behind doing this was being extremely clear about the functionality.

Users who open the extension for the first time should be able to determine what to do in less than 4 seconds. This was my approach to achieve a UI with very low cognitive load.

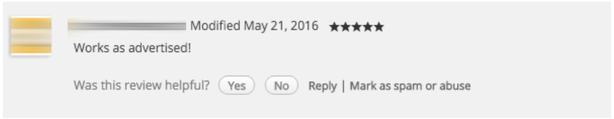
[whoisjuan]

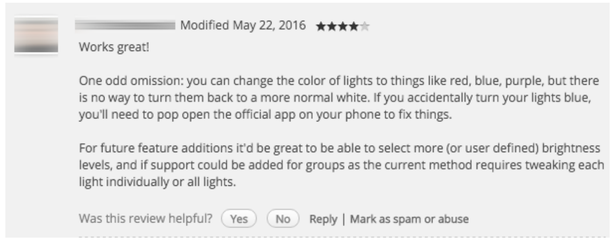
LIFX - Chrome Extension

Iteration

After releasing the initial version of the extension I decided that I wanted to wait and see if it could get organic traction. This would prove that the extension was indeed solving a problem.

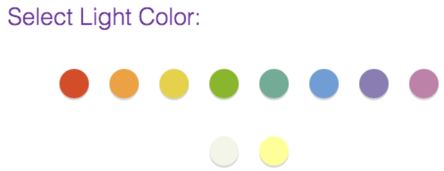
After one week of being released, the extension started to get some traction and users . Here are some of the initial reviews:





The initial reviews were pretty positive, but as shown in the example above there were some things I left out that were degrading the overall experience. I quickly learned that people did use groups (I don't), so it was pretty important to bring support for this. Additionally, there was an embarrassing thing I overlooked. Of course users want to change the color of their lights, but they are most likely going to change between warm and cool, rather than between spectrum colors.

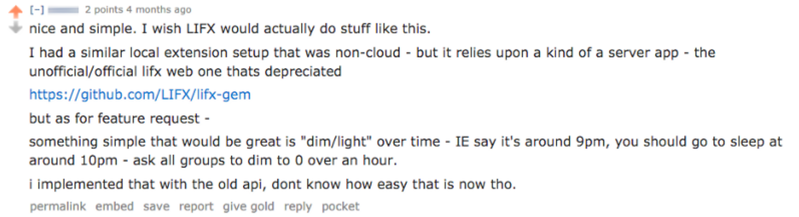
My first iteration of the extension was a quick update with support for groups + neutral color selection.



The process of iteration has been a continuous task. As the extension keeps getting new users, multiple new requests show up on my inbox.

Part of my process involves evaluating all the requests, determining the technical feasibility of those requests and then evaluating if they are aligned with the original heuristics of my design. If a request seems to oppose the Minimalist nature of the extension, I try to dig more into the problem and identify why a users is requesting a certain feature. Many times users might be dealing with a bug or perhaps they are hitting a usability wall. Staying true to your heuristics is a way to scope down problems and reveal unidentified issues.

Here is another example on how I approach this part of the process. For instance, not so long ago a user requested a way to set timed states in the extension:

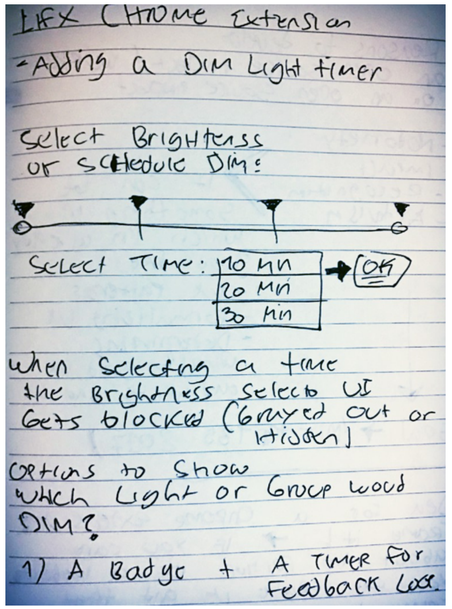


This would be an awesome feature. The API gives support for this so there's no need for developing a server side to handle this. However, at this point I had other competing requests. Since it's hard to gauge the impact of this feature, this is a moment of the process in which is important to rely on the original heuristics.

Is this a power-user feature? How much "minimalism" and simplicity is being sacrificed if this feature is added to the extension? Would users easily understand the functionality behind this feature? Is there a way to add this feature without adding friction to the whole process?

Asking these questions allow me as a designer to put these requests in the right perspective. It's also a good starting point for a quick evaluation exercise like the one pictured below:

LIFX - Chrome Extension



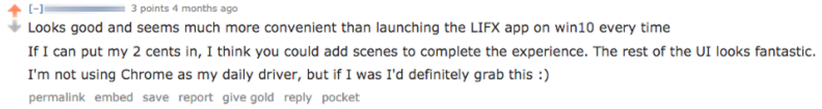
After doing a quick evaluation, it's somehow clear that adding this kind of feature can be expensive and could play against the main heuristic.

It's important to mention that this doesn't mean that feature is not valuable or not achievable. This is just a quick evaluation on determining the complexity of adding this kind of feature. Since there are multiple other requests competing for the limited time that I have for this project, I need to find a method that helps me to bring the greatest amount of value to the end user. Staying true to my original heuristics is an unmatched method to achieve this.

Metrics

Although evaluating feature additions under the light of a heuristic can bring clarity to the additive nature of creating software, it's definitely important to contrast any assumption with available data (if there's data to analyze).

One common request that I got in both emails and thread comments was the ability to activate scenes from the extension.

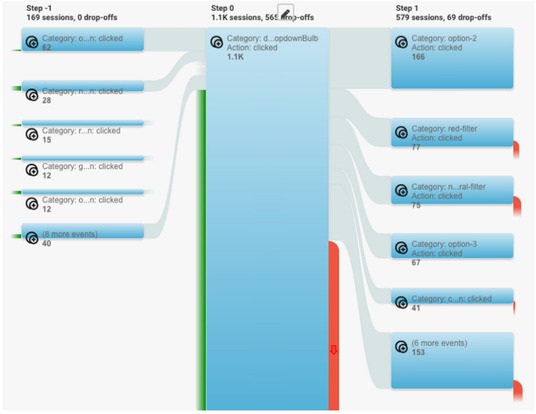


I personally don't use scenes that much, but I know they are an important piece of the LIFX Ecosystem experience.

In order to determine the best design for this problem I analyzed some of the available data I was capturing with Google Analytics.

One particular subset of behaviors that grabbed my attention was the use of the dropdown picker. At this point the extension had support for individual bulbs and groups, and both could be instantiated from a row of radio buttons and selected from the same dropdown.

Here is some data that shows the event flow for that particular dropdown:



One interesting thing that can be appreciated in this flow, is that there are multiple users doing certain actions, then selecting a bulb and then doing something else with the new bulb.

This is something expected if you have multiple bulbs, but also I originally assumed that users were going to control those bulbs through groups and not necessarily scenes.

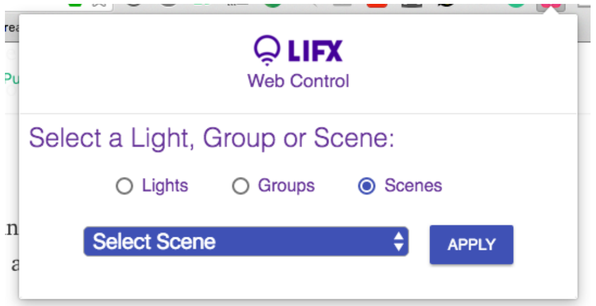
LIFX - Chrome Extension

This gave me an important insight about the LIFX ecosystem. Users don't care that much about groups and instead they use scenes as the principal mechanism to control multiple lights.

The raw data also proved me right on this assumption. The event that tracks the dropdown for groups is ranked very low against other user events.

After analyzing this data I was able to determine the importance of "Scenes" as a feature. Also the data gave me clear insights about the dropdown and the importance of the UI element in the general behavior of the app.

Originally I planned to add support for scenes using a complete new row of UI, but using the data I was able to determine that users were learning and understanding the concept of the dropdown picker. With this insight I was able to create an implementation that added scenes as another possible selection from the dropdown picker, instead of creating extra UI that would compromise the simplicity of the app.



Solution / Current State

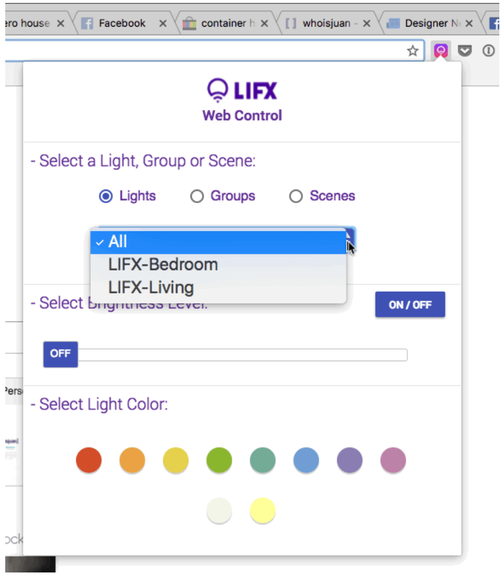
LIFX Web Control is currently the only available Chrome Extension for LIFX and the only platform agnostic solution for the LIFX Ecosystem.

Currently it has almost 1K users and most of them are actively using the extension.

Although I have limited time, I keep adding new features and fixing issues as they present. The latest update includes different things like the ability of toggling ON and OFF lights (before this was implicitly built into the brightness controller). This was another feature that I carefully evaluated using data and a heuristic analysis.

I also added features to enhance user on-boarding like oAuth Login and Notifications.

This is the current state of the extension with all the different improvements that had been implemented based on the collected feedback and usage data:



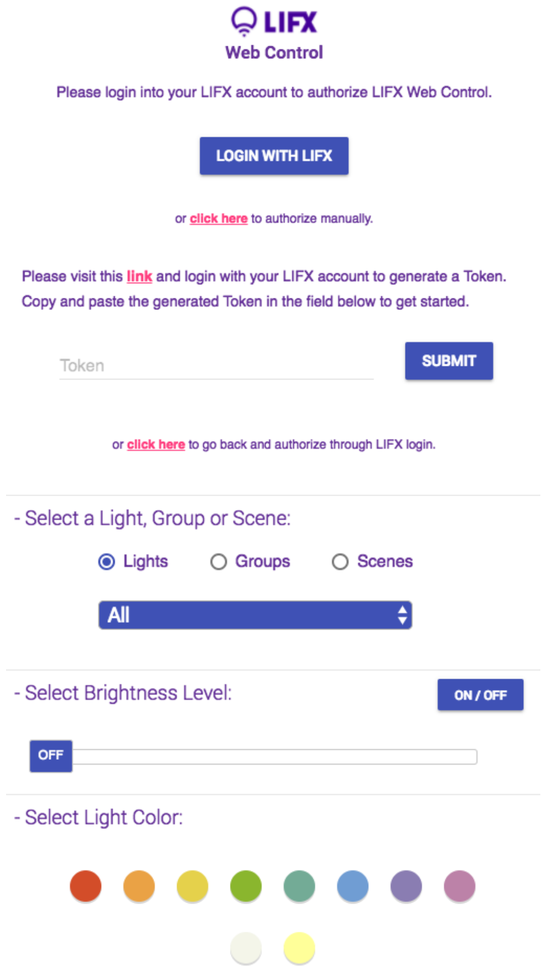
View GIF: https://cdn-images-1.medium.com/max/800/1*H2mkJhkYFxQ8mrQwpPnGfw.gif

You can also see the latest update in the Chrome Web Store, and of course if you're a LIFX user, I encourage you to download the extension and give me your feedback.

[whoisjuan]

LIFX - Chrome Extension

Other Images



[whoisjuan]

CoverPocket-UX Audit

Year: 2016

Client / Company: CoverPocket / CoverWallet

Role(s): UX Design and Product Management Consultant

Contributions: UX Audit, Wireframes, Prototypes, Specifications, Program Design.

Final Result / Deliverable: A full audit with UX suggestions and design proposals. CoverPocket didn't have a UX designer so the goal of the audit was to fill the User Experience gaps of their app (a personal insurance manager) and give them resources to take adequate UX decisions.

I also designed their whole referral system and provided all the major technical implementation specifications necessary for launching this program.

Problem

In 2016 CoverWallet was in the process of launching a consumer product called CoverPocket. CoverPocket is an insurance wallet that gives users a single hub for all their insurance policies as well as free access to an insurance consultant + insights on premium prices.

CoverWallet reached out to me seeking for advice. They specifically asked me two complete two major strategic tasks.

1) CoverPocket was implemented without a designer's input. They were worried about the customer experience of the app, as well as potential usability issues. They asked me to run and audit and make recommendations.

2) In order to promote CoverPocket they had the idea of implementing a referral program. They asked me to design and spec all the pieces of this program which was crucial in their marketing efforts.

How to perform a comprehensive User Experience Audit that covers all potential usability issues?

How to design an efficient and low cost Referral Program with very little technical footprint?

Process and Solution

My first task was performing a comprehensive UX Audit of the app in its current status. In order to achieve this I laid out a Usability Inspection and Analysis plan based on Nielsen's Heuristic Evaluation and Weinschenk and Barker Classification.

I ran the evaluations through all the reachable views and forced the app through multiple unhappy paths. I then scored the views and paths based on the baseline methodologies and ruled out false positives by excluding weak offenders.

Below you can view the full document with a disclosure of the methodology and all the criteria used to perform the evaluations.

1

[whoisjuan]

CoverPocket-UX Audit

2

Methodology

This report contains a usability inspection report for the app CoverPocket (for iOS).

The inspection and analysis found in this report was generated using two main methodologies:

1) Nielsen's Heuristic Evaluation [1]

2) Weinschenk and Barker Classification [2]

All the identified problems have a proper argumentation based on these inspection methodologies. None of the suggestions have an impact in the look and feel of the app unless the problem is derived from an aesthetics inconsistency.

False positives were ruled out by excluding cases in which there wasn't a strong contradiction to one of the proposed used methodologies (equal or less than 2 on a scale from 1 to 5)

* 1) http://www.nngroup.com/articles/ten-usability-heuristics/
* 2) http://www.measuringu.com/blog/he-cw.php

coverpocket

3

Abstract

In the usability inspection of the app the following issues were found:

- 4 critical problems that are explained individually. Suggested solutions are provided.

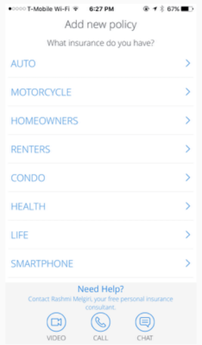
- 3 low severity problems that are explained in an "Other Problems" section.

- 5 low to medium severity usability problems associated with bugs that would be reported directly to the developers or in an email/call to the product team.

coverpocket

4

Usability Inspection: Upload after Sign-Up
Problem 1: Un-skippable Intro Upload



Problem:

Although is nice to introduce the user to the main task after signing-up, this usually needs to be an skippable activity since we don't know the current state of the user. Perhaps they don't have the policy in their hands and just want to explore the app.

Violated Heuristics:

- User control and freedom (Severity: 4/5). Nielsen

- User Control (Severity: 4/5). Weinschenk

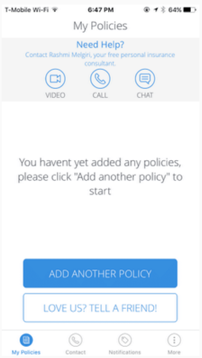
- Fulfillment (Severity: 3/5). Weinschenk

Solution: Allow the user to skip this section.

coverpocket

5

Usability Inspection: Main Screen
Problem 2: "Add Another Policy" Language



Problem:

"Add Another Policy" is inaccurate language for the task that needs to be performed here. When I'm getting the suggestion to add "another" there's the logical assumption that there's already an existent policy in my account which is not true in this case.

Violated Heuristics:

- Linguistic Clarity (Severity: 3/5). Weinschenk

- Consistency and standards (Severity: 3/5). Nielsen

Solution: Change to "Add a Policy" which satisfies all the cases for this task.

coverpocket

[whoisjuan]

CoverPocket-UX Audit

Usability Inspection: Upload Task Camera
Problem 3: Lack of a Proper Upload Flow

Problem:

When uploading a policy from the camera phone, there's a un-proper manage of the user control and status visibility. The main issue is that when user upload a picture they cannot see the result of it and they can't edit or retake the picture. This can generate multiple problems for agents and users alike.

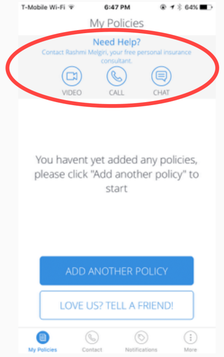
Violated Heuristics:

- Visibility of system status (Severity: 4/5). Nielsen
- User control and freedom (Severity: 4/5). Nielsen
- User Control (Severity: 4/5). Weinschenk
- Forgiveness (Severity: 5/5). Weinschenk

Solution: Provide a mechanism to preview a picture and retake if necessary.

coverpocket

Usability Inspection: Multiple Screens
Problem 4: "Need Help?" Element



Problem:

The "Need Help" element is a great addition that shows the relevance and added benefit of having direct access to human support in any moment. However this element from a UI standpoint breaks the flow of the screen navigation and greatly reduces the available screen real estate. These things can generate confusion on the final user. Also the presence of the element is not consistent in the different screens it appears (different positions and different screens without an specific reason).

Violated Heuristics:

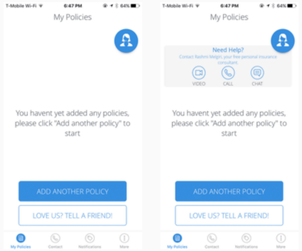
- Aesthetic Integrity (Severity: 4/5). Weinschenk
- Simplicity (Severity: 2/5). Weinschenk
- Consistency and standards (Severity: 4/5). Nielsen

Solution: Generate a different way to display this UI element, creating hierarchy and consistency. See appendix for suggested solution.

coverpocket

Usability Inspection: Multiple Screens
Problem 4: "Need Help?" Element */appendix/*

Suggested Solution: The suggested solution is floating button that displays the "Need Help" element at the user's will. This element should start open to explain the user the behavior associated with the floating button. This solutions provides consistency and simplicity and conserves the integrity of all user flow through out the screens. It also provides an agnostic way to add this element to more screens.



Prototype: https://invis.io/FK579KP5V

coverpocket

Usability Inspection: Multiple Screens
Other Problems (Low Severity)

1) Lack of a Headline Hierarchy:

Many titles, headlines, leading lines and labels have different sizes with no apparent reason. This creates a visual imbalance that makes some of the presented information hard to follow.

Solution: Have a clear architecture of sizes for Screen Headers, Headlines, Labels and Paragraphs.

2) Email on Contact doesn't open the email client of copy the email address (Contact):

An email address is provided but there's no way for the user to interact with this email address beyond them remembering it to write a mail. This creates a problem since the user might want to use this email address but there's an added friction to use it.

Solution: Let the user copy the address and open the email client when they tap it.

coverpocket

[whoisjuan]

CoverPocket-UX Audit

10

Usability Inspection: Multiple Screens
Other Problems (Low Severity) */Continued/*

3) Lack Interaction in a pending policy (My Policies):

There's no feedback or action provided by the app when a user wants to interact with a pending policy. Users might get frustrated when having pending policies that are don't generate feedback to their interactions and cannot be deleted.

Solution: When a user taps a policy the ideal solution would be showing them what they submitted. If this is out of the current product scope, then at least showing a notification giving them further information would provide a better way to control this state.

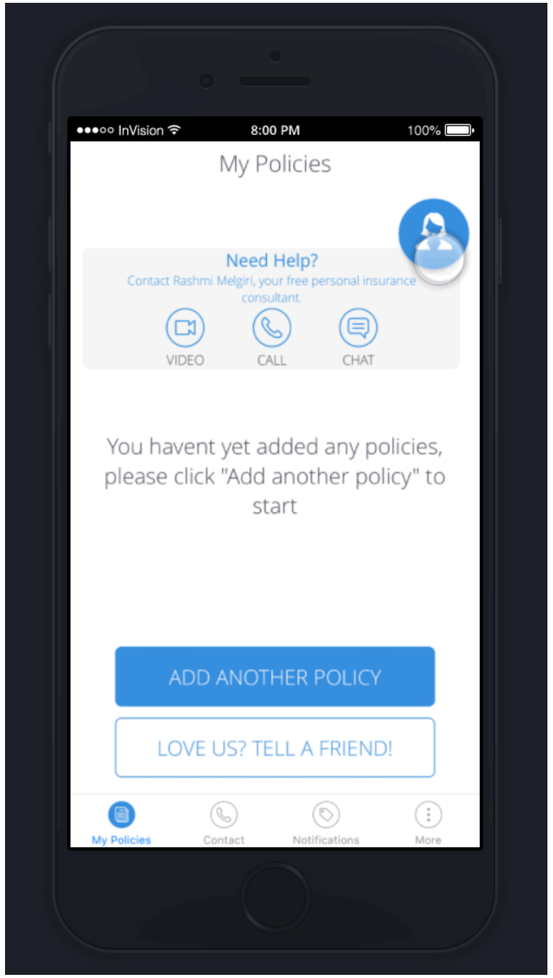
4) Lack of a proper information distribution in the policy view (My Policies):

The current policy view has a very unbalanced flow of information. One of the reason is that there's no proper distinction between the shown information and the actionable tasks related to that specific policy. This can generate confusion on the user or change the attention of the user from important information to least importan information, which can cause problems at the business end.

Solution: No current solution suggested. Further user testing and feedback is needed to understand how this view can be improved, but an initial suggestion would be improving the distinction between the elements to achieve more readability.

coverpocket

One of the main value propositions of CoverPocket service is the ability of reaching out to a free consultant directly from the app. In the version I evaluated this functionality was contained in a section called "Need Help". Although this component was providing a lot of value, its position within the app was breaking the screen navigation and disturbing the general real estate of the app. I suggested a small interaction to hide and control this component. I created as small prototype to pitch the interaction:



View GIF: https://cdn-images-1.medium.com/max/800/1*2O7_0ZsYnTfoouBDwRnKBQ.gif

[whoisjuan]

CoverPocket-UX Audit

The second task I was asked to execute, was to design a very lean and flexible referral program for the app. They had a referral budget and they wanted to reward referrers with actual cash.
I did several strategic recommendations on several areas. I made suggestions regarding conversion, flow triggers, technical implementations, value proposition and general program design.

I created a document outlining all my recommendations, however I had multiple interactions with CoverPocket's product manager, a back-end engineer and mobile engineer, in order to design and deliver an optimal system design. One of the biggest challenges was to design something that was minimum but remarkable, with very low technical footprint and with design room to be iterated, optimized and changed.

You can find the document with all the program design and technical specification below:

1

Referral Program: Product Specs, Wireframes and Implementation Guide.

Produced by: Juan J. Ramirez, PM & UX Consultant, MET. Carnegie Mellon University
Client / Product: CoverPocket

coverpocket

2

General

The Referral System is a set of business rules that allows CoverPocket users to promote the app within their network and get rewarded for doing it.

Users can earn cash rewards by responding to CoverPocket promotions or by promoting their invite links. When an invite link is used in a new sign-up, and a certain action happens after the sign-up (for example, uploading a policy) the referrer and referral get rewarded.

In some cases CoverPocket can also enable an additional rule of giving special rewards to users who sign-up with pre-generated marketing links.

coverpocket

3

Business and User Rules

- CoverPocket users can share the app with a personalized ref link through social media or direct channels as SMS or Email.
- When a new user register through a referral link and uploads his/her first policy, both the referral and referrer get a cash reward that is added to their accounts.
- The user can see his/her earned credit under the Tell a Friend Section.
- The user can connect a PayPal account and CoverPocket would deposit the earned rewards to that PayPal account in a weekly or bi-weekly basis.
- CoverPocket admins should be able to manually add and subtract credited rewards from the users accounts.
- CoverPocket admins should be able to review the legitimacy of a policy and invalidate earned credits.
- The accounts should reflect an updated total after a payout is made.

coverpocket

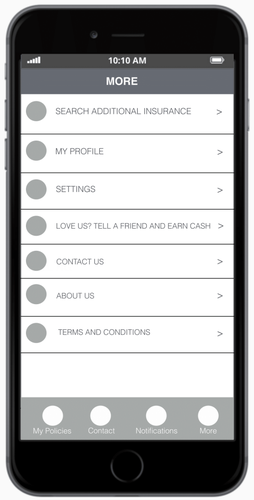
[whoisjuan]

CoverPocket-UX Audit

4

Wireframes

More

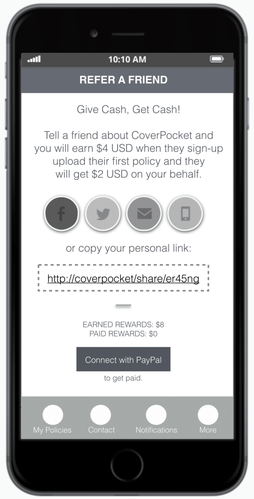


coverpocket

5

Wireframes

Share View: Basic

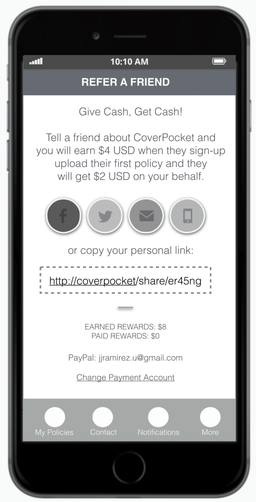


coverpocket

6

Wireframes

Share View: PayPal Connected



coverpocket

7

Implementation Guide (for iOS)

In order to implement the proposed referral program we will be using the following third party referral program SDK and cloud-based software:

- Branch.io (https://dev.branch.io/recipes/referral_links_with_incentives/ios/)

The initial rewarding system should be hooked to PayPal which would be the payment method and gateway to award the money users would earn in the program. For this we should use a script that manages the payments vid the PayPal payouts API:

- PayPal Payouts (https://developer.paypal.com/docs/api/#payouts)

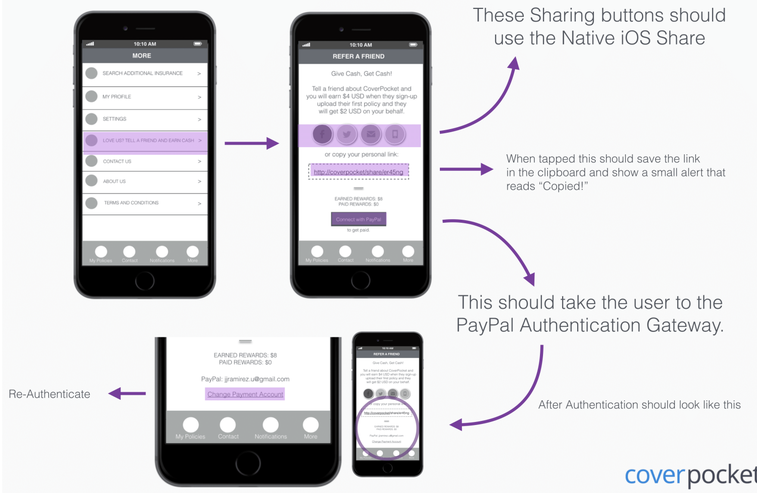
coverpocket

[whoisjuan]

CoverPocket-UX Audit

8

User Flow



coverpocket

9

Implementation Guide (for iOS)

The suggested **Referral Program Architecture** is the following one:

1. The Referral Program would be implemented using Branch.io custom integration. Branch.io provides an end-to-end solution that tracks referrals and keeps the referral records on their own system.

2. Since Branch.io keeps the records of the referrals and credits, there's no need to track this on the user column in Parse. This means we would have a completely separate architecture to keep track and manage the rewards. However you may still need to create a new column in the Parse DB to store the PayPal email of registered users and track the rewards that were paid.

3. CoverPocket would grant credits to both referrer and referral after a custom in-app event is triggered. The event would be: **uploading the first policy after signing up**. See the custom events documentation: https://dev.branch.io/recipes/advanced_referral_incentives/ios/#custom-events

4. A user who wants to participate in the referral program just needs to share his/her link which would be generated by the SDK. However in order to get paid he/she needs to provide a valid PayPal account. For this we should use the PayPal Login solution (https://developer.paypal.com/docs/integration/direct/identity/log-in-with-paypal/) so we can get the correct email from the user. Once we pull that email we must store it and link it to that user.

coverpocket

10

Implementation Guide (for iOS)

Continued:

5. Once the users start earning credits for their referring activity, this information would be stored on the Branch.io records. Using this information we should run a weekly job (a hosted script/program should do this) for (1) Paying-out those credits. (2) Updating the records after the payout.

6. If a user hasn't linked a PayPal account yet, this user should be skipped and instead an email reminding him/her to link a PayPal account should be sent.

7. For information about the amount of credits that should be honored to the referral and referrer, as well as the day of the week for the payouts, please talk with **Luis Pino.**

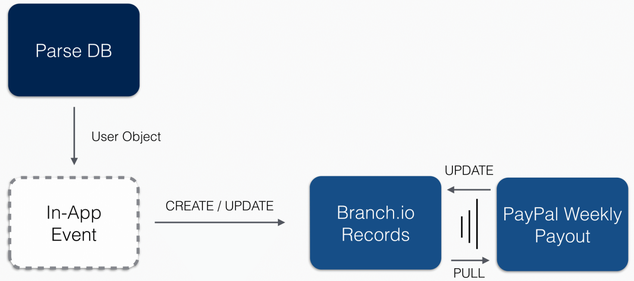
8. Additional Notes: (1) Please have in mind that the validity of a policy should be verified before granting a reward. Therefore the event that triggers the reward should be when the Policy is verified by an agent or it is uploaded manually. (2) If possible please provide a custom admin to remove and add credits based on username/email. (3) In order to implement this correctly you will need to get familiar with Branch.io Webhooks. You can find that documentations here: https://dev.branch.io/recipes/webhooks_and_exporting_data/

coverpocket

11

Implementation Guide (for iOS)

Below you will find a diagram suggested Referral Program Architecture and how it should integrate with the current infrastructure:



coverpocket

[whoisjuan]

Procore - Mobile Filters

Year: 2016

Client / Company: Procore Technologies, Inc.

Role(s): User Experience Designer

Contributions: General Feature Concept, Main User Flow and Story Map, Contextual Inquiry, Usability Calls, Wireframes and Mid-Fi Mockups, Clickable Prototypes.

Final Result / Deliverable: Final design pattern with all the development requirements. The new feature was developed and released in the iOS and Android Native Apps resulting in a 40% time improvement in data searching.

Problem

The Procore Mobile App is used by thousands of Construction Field Workers to capture issues, track progress and measure quality/safety. Procore Mobile is widely used by superintendents, project managers, foremen and other on-field workers.

One challenge that appeared with the growth of the product and the addition of more comprehensive issue capturing tools, was the cumbersomeness around finding specific groups of issues.

As a construction project move forward thousands of items are collectively recorded by different workers and stakeholders. Those items need to be referenced in later stages to be effectively addressed. Procore users were having an sub-optimal experience when it came to find and reference previously created items.

I was tasked with creating an scalable and re-usable UX pattern to solve this problem...
What kind of solution can provide an agnostic filtering pattern and an fast way to find specific groups of items with easiness?

Process

The first step I took to solve this problem, was to analyze anecdotical and quantitive data, in order to build an insight around the problem. Anecdotical data was gathered from 3 main sources: User Feedback Tickets, Customer Support Tickets and User Sessions Recordings.

Quantitive data was gathered from our analytics software (Google Analytics and AppsSee). I extracted pertinent information from the main events dataset and analyzed sessions that mainly occurred in our specific tools lists views. I analyzed funnels and focused in understanding the behavior behind the available filtering methods (*Segmented Controls and String Search*).

After gathering all the data and analyzing the general user requests, the problem became evident: Users wanted more granular control to filter down the items lists, but they had an special interest in filtering items by two specific properties (Assignees and Locations).
In order to create the right filtering pattern and filtering mechanics it was also important to understand the behavior behind data searching in Procore, particularly the nature of datasets and the nature of the filtering parameters. One clear challenge was the fact that Procore users are likely doing two types of data search:

1) Locating a know item within a fairly large list of items. 2) Locating a unknown item based of a set of known input parameters.

This is usually very unusual for filter paradigms since most of the time filtering mechanics are created to address specific dynamic searches or fuzzy searches, bur rarely both.

Procore - Mobile Filters

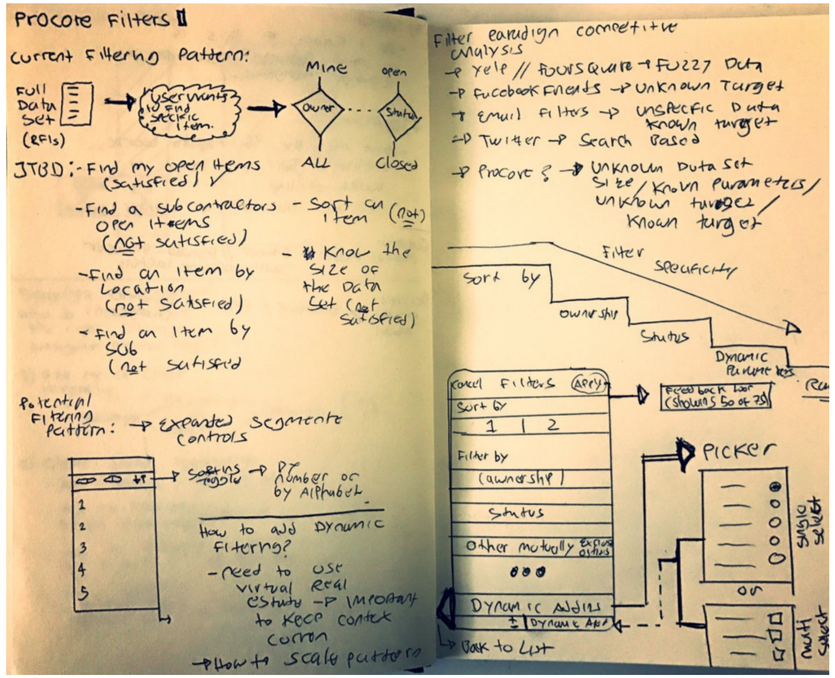
I did several analyses to dissect the problem. My first approach was analyzing the existing patterns (*Segmented Controls and String Search)* and determining what “Jobs To Be Done” were being satisfied by the existing patterns. Once I determined which JTBD we not being correctly addressed I started working in the a first pattern that could potentially solve the missing cases.

I then use that pattern to expand into other existing problems and issues and formulated new questions around problems like context and scalability.
One of the problems I unveiled in this phase was the limitations with screen real estate and the need to keep the users in context.

To solve this, I explored several solutions using virtual real estate (drawers and sliding views) as the baseline of the design. After several explorations I created a semi-realistic prototype of the filters view and functionality.
This gave several ideas to address these issues. The first one is that context wasn’t going to be lost as long as I designed clear feedback loop mechanisms.

The second idea is that as long as I kept a hierarchy of filters and a consistent message around the content and the mechanics, users were going to have an easier learning curve.

Here is a photo that illustrates my design approach to dissect the problem and conceptualize a solution:



After having an initial clear idea and a set of mocks and early prototypes, I went through a formal design critique process in which I got feedback from other UX designers and major stakeholders.
Finally, I performed a series of usability calls to validate the new real estate paradigms and did two onsite interviews in which I put the prototype on the hands of our customers. The calls and interviews proved that the paradigm was usable, easy to learn and highly functional.

The new filtering paradigm also was huge improvement in the use of screen real estate. The main two optimizations I did to achieve this were:

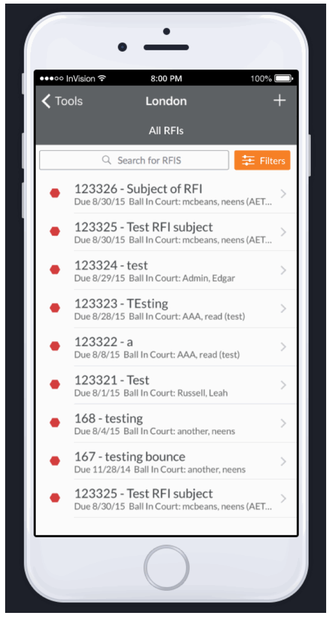
1) Segmented Controllers were deprecated giving extra room for the list elements and other potential UI elements.

2) A complete new view for filters was adopted, taking advantage of the virtual real estate that until now was rarely used in Procore.

[whoisjuan]

Procore - Mobile Filters

Here is the first prototype that contains all the initial patterns and elements including a feedback loop mechanism:



View GIF: https://goo.gl/RJhMgo

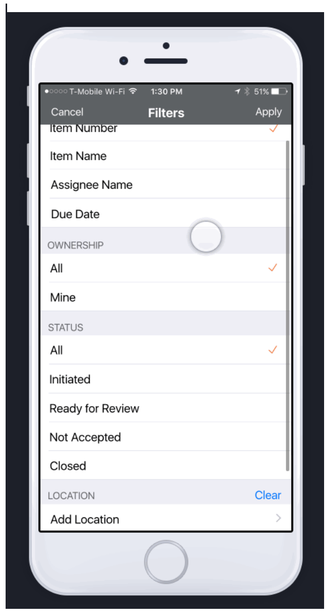
Solution

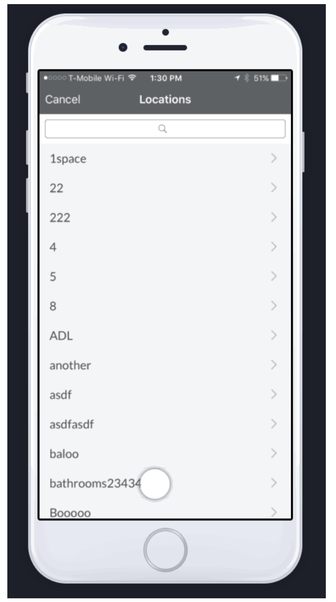
Since the initial design iterations proved to be successful, I decided to move into creating the production ready design. After discussing the feasibility of the design with the engineers I started re-working the User Interface elements. At this point it was clear that we wanted to stay close to the Apple Design Guidelines, so I created a new view using only UI-Kit elements.

The new designed contained all the required filters. I also took the decision of going with a full height view and rely on the slide-in animation to keep the users in context. My initial process and testing proved that as long as a feedback loop mechanism was in place, it was possible to take advantage of the complete screen viewport.

I did a new round of testing and interviews with the updated design and got a hugely positive response. After polishing some details, I created the final specs for the feature and the interactions.

Here is the final prototype that follows Apple's HIG with dynamic filters adding and filter clearing flows:





View GIF 1: https://goo.gl/OI53az

View GIF 2: https://goo.gl/ZpeSLf

Results

The Mobile Filters Initiative brought to life a very demanded feature that was needed to keep the mobile platforms in parity with the web app.

Before releasing the feature I gathered time duration data for actions like closing items, issue navigation and search.

I watched more than 4 hours of user sessions that happened mainly in the views that were getting the new filters. By doing this I was able to detect hundreds of user navigation bottle necks. Since I was confident those blocks were going to disappear with the new feature, I manually created a list of the top issues and used it as a baseline for measuring success.

After releasing the feature and getting and overwhelming positive response from our mobile users, I did a new round of data gathering and user session exploration. It was clear that the feature was a success.

The filters reduced the navigation time (within the item lists) by 40%. They also **reduced the amount of no-action sessions** (sessions in which users navigate within a tool but don't perform any workflow related action) **by 20%.**

[whoisjuan]

Sazhimi - Funnel Optimization

Year: 2014

Client / Company: Personal Project - Spotify - Shazam

Role(s): User Experience Designer, Developer

Contributions: General Feature Concept, Main User Flow and Story Map, Contextual Inquiry, Wireframes, End to End Development.

Final Result / Deliverable: A Web App to import Shazam tags into Spotify.

Sazhimi: A Case Study on How to Use a Shiny Landing Page to Boost Funnel Completion.

Have you ever faced a challenge when trying to engage your users with a set of tasks that are required to provide a valuable output?

This is a case study of a UX pattern that I designed to increase funnel completion rates in these situations. I designed it for a personal project called “Sazhimi” but you can adapt it to your own needs. It’s amazingly effective and easy to implement.

For context, Sazhimi is a little tool that helps you to create a playlist from your Shazam history. If you’re thinking: “ohh but that feature already exists in Shazam”…well, you’re not wrong. But let me tell you a couple of things. First, I created Sazhimi in the winter of 2014 when this feature wasn’t available yet in any Shazam app. Second, the whole Shazam-Spotify integration is a complete mess (from a UX perspective) and sometimes it doesn’t even work. Sazhimi remains relevant because it’s a lean and straight forward solution to export your tags into a Spotify playlist.

So how does the technology behind Sazhimi works? Basically, the tool is a mesh of Python scripts that are executed in an specific order while they report back to an static HTML view. The program will parse a file provided by the user, then it will extract relevant information from it, and finally using this information it will push some information in a third party service which in this case is Spotify. All these steps are executed in a very synchronous way and most of them need some kind of user input (uploading a file, authorizing an app, authorizing Sazhimi, etc).

So in few words, Sazhimi is a small tool that only works if the user is willing to provide a lot of input. It’s basically a super complex funnel that consists of the following tasks:

- Arrive to the website and understand what’s going on. (Some bounce rate here that can be improved with better aesthetics and copywrite)
- Leaving the website to open Shazam, logging into Shazam and then downloading an HTML file that contains their Shazam tags history.
- Coming back to the Sazhimi Website.
- Loading the HTML file from their local disk.
- Pressing the Upload File Button.
- Pressing the Login to Spotify Button, and then do all the required login and app authorization stuff.
- Press the “Sazhimify” Button to trigger the the list parsing, song fetching and creation of the playlist.

I quickly realized that most users were going to abandon this funnel, especially if it was not clear enough what was happening while they execute all these steps.

In order to make the process leaner and less exhausting for the user I started to brainstorm some techniques and UX patterns that I could use without increasing the technical complexity of the tool. Unfortunately most relevant patterns for this kind of process oriented tasks are too generic and wouldn’t improve the completion rate.

After some hours of additional brainstorming I came up with an idea.

What would happen if I use a **one page website layout** and I modify it in a way that works as a progressive waterfall of steps?

Since many of the required Sazhimi tasks are actually happening outside of the website this might actually translate into a very simple and effective page that would narrate the process in a very efficient way. Also,it would work as the main mechanism to achieve what the page itself is trying to deliver. Perhaps, it would persuade the users to complete the whole funnel without thinking they are diving into a boring and complex task.

Sazhimi - Funnel Optimization

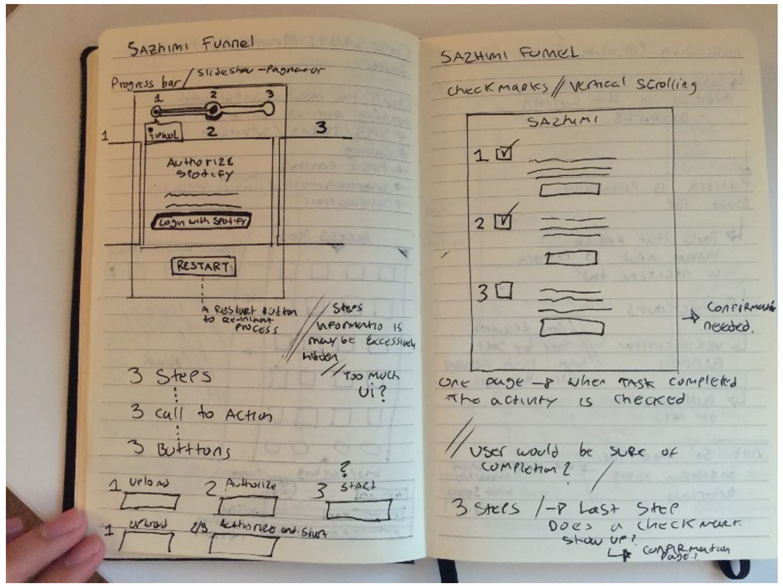
“Waterfalling” the Funnel

After I had a general idea on how the funnel should work, I started working on some concrete ideas on how to achieve a solid pattern.

What I wanted to do was very simple. I wanted to create a landing page that described the process step by step without separating the explanation from the actual work.

So instead of trying to find a way to first teach the users how to use the tool and then prompting them to use the tool, I wanted to create something that prompted the users to dive into the task while they were learning about it.

This is actually a simple concept but also very abstract. In order to translate this concept into a concrete interactive UI, I did some sketches of my initial ideas. Here are some of those:



One of my initial ideas was to use a very prominent progress indicator as the governor of the page interactions. I wanted to do this since it’s common UX knowledge that users are more likely to dive into a task if they know how much involvement, skill and time is going to demand from them. This is the same rationale behind why on a week night you decide to watch an episode of a Netflix show instead of a movie. You simply don’t want to commit to the time that takes to watch a movie so you take a decision based on the running time of your available entertainment grid.

After some thought, I decided to drop the idea of the tracker because Sazhimi has a series of sub-steps that cannot be tracked and this would introduce confusion and ambiguity to the pattern. However, there was a very valuable insight I discovered while exploring this idea. In the Netflix example, I said that on a week day you usually would choose the less time expensive option, and although this is true, you always end up binge watching House of Cards on a Monday night. This means that even though you took a decision based on time, you can easily commit to more time if the task is broken down in a rational way.

The Idea

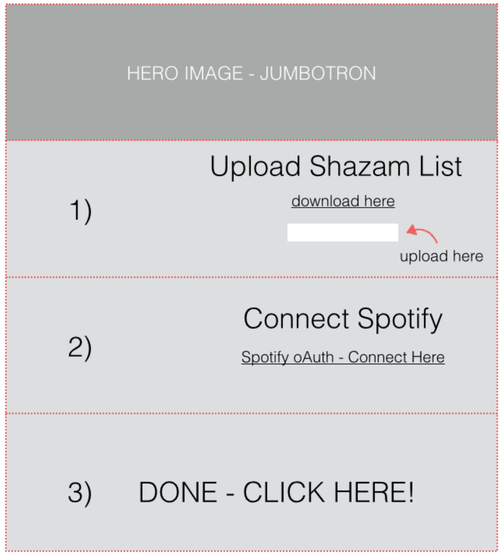
I took these insights and build a concept around them. In order to embrace the idea of a super-charged, predictable and plotted funnel I had to build something that did the following things:

- All the descriptions of the tasks should be available at the first reading pass of the website (the initial exploration). No hidden steps or small letter trades.
- Although the funnel requires a lot of input from the user, it should persuade them into seeing that the individualized tasks are very simple.
- The funnel should not have more than 3 steps. It certainly can have more than 3 tasks but it should sell the idea of only 3 steps. The reason for this is that there’s are some idiosyncrasies that relate “3 steps” with speed and easiness. (e.g:http://idioms.thefreedictionary.com/as+easy+as+123)

Sazhimi - Funnel Optimization

- It should give visual and contextual cues of progress that are not attached to a single component. In other words, it should clearly indicate to the the user that they moved in to the next task.
- It should be a waterfall with no go back option. If they need to go back is because the tool is not usable enough.
- Users should be able to abandon the funnel at any point and with no regrets about the time or effort invested into the tool.

After carefully exploring several wizard configurations and task completion experiences I decided to go with a one page website divided into sections.
This approach allowed me to execute the idea that I had from the beginning: blending the instructional information with the tasks.
Here is a ultra lo-fi wireframe with all the general ideas of what I wanted to achieve.



As you can see, my idea was to have something with the look and feel of an informational landing page, but with the mechanics to achieve the sync embedded in the content itself.
This is the most powerful concept behind this pattern. By simulating the look and feel of a promotional landing page I also inherited the simplicity of this kind of layout, but more importantly its effectiveness to deliver a message.
Additionally this layout gave me two very powerful tools:

- Navigational Scrolling: Modern landing pages have navigation capabilities through auto scrolling. You can create a navigation tree that takes users to certain scrolling position after clicking a link. I used this to take users from one task to the next one.
- Physical Dimension: This concept is a little bit tricky, but actually a website height can behave as an abstract loading bar, a progress tracker and/or a time estimator. If you find this confusing try to think how many times to do you scroll to the bottom of an article before reading it. The reason why you do this is because you're trying to estimate the length of the article. But also the same concept can create a meaningful mechanic to show progress, since you can show the user that by approaching to the bottom of the page they are almost done with the task.

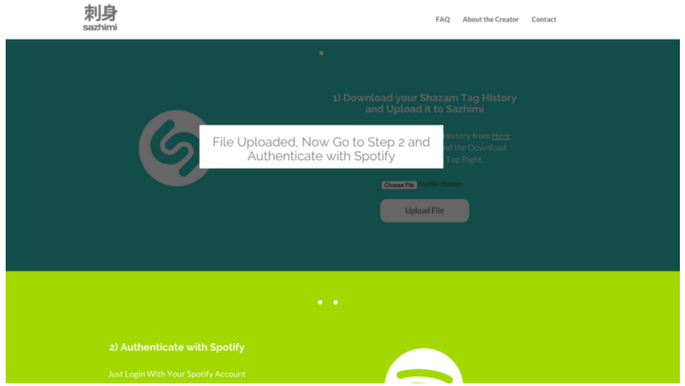
The Implementation

The final implementation of Sazhimi is a streamlined funnel that guides the user through the different tasks while trying to preserve the relevance, simplicity and context.
The main features of the UX pattern implemented in Sazhimi are:

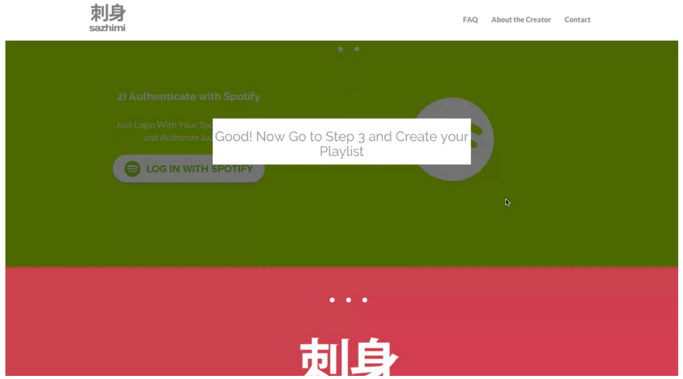
One page: A one page website with three sections / three steps process featuring instructional information and call to actions to complete the tasks. The website auto-scrolls from one section to the next one as users move through the funnel.

Sazhimi - Funnel Optimization

Blocking: This is my favorite part of the pattern. The page auto scrolls to indicating the users there was progress/ advancement. However, the users might be unsure of what they did or maybe they might be unsure about continuing. In this case they would likely scroll-back to the upper section. In Sazhimi they will see an overlay obfuscating and blocking the previous section, as well as a dialogue that encourages them to continue with the process.



Empowerment: Technically there's no need for a step #3 since I could run this function when getting the callback from the Spotify Authentication. However, by adding that extra step we are creating the notion of control and decision making. Of course, as I already mentioned before, there's also a cultural meaning behind the concept of "1,2,3" which helps to create a compelling message.



[whoisjuan]

Sazhimi - Funnel Optimization

Some Numbers

Since I implemented and launched Sazhimi I have received a decent amount of traffic that has been helpful to discover the effectiveness of the pattern and the funnel completion rates. I'm not going to lie, the bounce rate of the website is quite high. Around ***65%*** of the organic-real traffic goes away after landing on the website.

For those who stay around and dive into the funnel there's an average completion rate above ***60%*** which is amazingly good if you have in mind that they need to leave the website to download a file, comeback and then upload that file. Even better, for those sessions from specific promoted sources (Facebook, Twitter, Reddit, My own portfolio), the completion rate jumps to above **80%.**

Of course, there's a small bias because users who consciously land on Sazhimi are actively looking for a solution. This is an extra motivation to complete the required tasks and proceed through the funnel. Nonetheless, the numbers remain solid for such a fragmented funnel, proving that this pattern can be valuable for other cumbersome processes.

Where to use this Pattern?

Many of the findings I did through the creation of Sazhimi are very local and self-contained to what Sazhimi provides. However, I believe that there are some valuable concepts that can be implemented in other funnels with positive results.

Some funnels that I personally think would benefit from this pattern are:

Surveys

Wizards, Setting-Up Experiences, On Boarding Mechanics.

Checkouts that require extra/external information, like for example uploading a prescription when buying lenses online.

... or in general anything that can benefit from quickly providing users context about the length of a task and can be optimized through a waterfall of procedures (each task require a result from the previous one to proceed).

If you feel there's value in any of the discoveries that I did while implementing Sazhimi, I encourage you to use those discoveries. Although the pattern itself has some unique implementations, in the practice is just a mix of multiple UX concepts that you can use if you feel they adjust to your particular situation.

Try Sazhimi in this URL:: http://sazhimi.herokuapp.com

[whoisjuan]

AHN - Savings Heart Project

Year: 2015

Client / Company: Allegheny Health Network

Role(s): User Experience Designer

Contributions: General App Concept, Main User Flow, Contextual Inquiry, Wireframes and Mid-Fi Mockups, Clickable Prototypes, Animation Narrative.

Final Result / Deliverable: iPad App with a Full Experience for Patients and Healthcare Providers.

Problem

Saving Hearts was a semester long project at the Carnegie Mellon Entertainment Technology Center sponsored by Allegheny Health Network. AHN was looking to develop a technology based solution to decrease the readmission of patients diagnosed with Congestive Heart Failure, a chronic heart disease that is incurable but that can be stabilized and controlled with the right medication and an improvement in life habits.

AHN asked us to develop an iPad based experience that would teach their patients about the importance of taking medication, having a good diet and exercising regularly. The biggest constraint for this project was the target demographic: Diagnosed Seniors, likely under-educated and financially challenged.

How to design a habit changing experience for third age users with little knowledge of technology, who live on a tight budget and that don't have the intellectual tools to understand the cause-effect between their condition and their life-habits?

Process

As the UX Designer assigned to this project I lead the initiative of creating a disrupting experience that would appeal our audience and achieve the results we were aiming for.

In order to understand the details of the condition and the problem, we met with a client team composed by Cardiologist, a Cardiac Nurse Practitioner, a Pharmacist, a Nutritionist and a Physical Therapist.

From this meetings we got a lot of useful information about the patients, their habits and their ongoing treatment. It was easy to determine that we needed a engaging experience with a strong feedback loop that would help the patients to understand the implications of having a chronic heart disease.

One particular area that I started addressing within my design analysis, was the idea of creating a dynamic experience that could serve as a resource for both the patients and the healthcare professionals. My first design approach to the problem was to determine which was the process used by the physician to explain the condition to his patient and detect why it was ineffective.

The process looked like this:

1) Appointment with the Doctor, in which the patients are informed about the diagnosis and current state of the condition.

2) Patients receive Informational Booklets. A new appointment to start the treatment is arranged.

3) Before starting the treatment, the patients meet with other healthcare professionals (dietitians, pharmacists, etc.). They provide the patient with further information about their condition and treatment.

After analyzing the flow and the diagnosis process I determined that there were three main problems:

- The process was extremely disjointed. The patients were receiving the same information but with different angles that didn't necessarily connect.
- The printed booklets were very dry and they didn't seem to have a connection to the main diagnosis thread.
- The patients didn't have a time to built and mental relationship between their condition, their life-style and the upcoming treatment. They were only seeing the value of the medication since this is the most critical component of the treatment.

AHN - Savings Heart Project

Here is an example sheet of a classic medication booklet given to patients with heart conditions:

Class of Medication	What the Medication Does	Things to Know about the Medication
Angiotensin-Converting Enzyme (ACE) Inhibitors Common names: • Lisinopril (Prinivil®, Zestril®) • Captopril (Capoten®) • Quinapril (Accupril®) • Ramipril (Altace®) • Enalapril (Vasotec®)	These medications work by in 2 different ways: 1) They reduce the pressure in your blood vessels to make it easier for your heart to pump. This way your heart doesn't have to pump against such high resistance (Garden hose analogy). 2) We all have a chemical in our body called angiotensin. If you have too much of this chemical your heart may change shape and become more weak. These medicines block that harmful chemical.	• **Call your healthcare provider** if you have swelling in your face, tongue, or lips. • May cause dizziness or weakness • If you develop a dry cough, contact your healthcare provider. This may be a side effect of the medication. • Your doctor may recommend blood tests to check potassium levels and kidney function
Angiotensin II Receptor Blockers (ARBs) Common names: • Candesartan (Atacand®) • Telmisartan (Micardis®) • Losartan (Cozaar®) • Valsartan (Diovan®) • Olmesartan (Benicar®)		• **Call your healthcare provider** if you have swelling in your face, tongue, or lips. • May cause dizziness or weakness • Your doctor may recommend blood tests to check potassium and kidney function
Beta Blockers Common names: • Bisoprolol (Zebeta®) • Metoprolol succinate (Toprol XL®) • Cavedilol (Coreg®)	These medicines work in 3 different ways: 1) These medications block a stress hormone that can be harmful to the heart. 2) They slow the heart rate to allow your heart time to fill before pumping. 3) Protects the heart from going into dangerously fast heart rhythms. Not all beta-blockers are used in heart failure – only the ones listed in the first column have been shown to have benefit.	• May cause drowsiness, dizziness, fatigue
Diuretics (Water Pills) *Common names:* Loop diuretics: • Bumetanide (Bumex®) • Torsemide (Demedex®) • Furosemide (Lasix) Thiazide or thiazide-like diuretics: • Chlorothiazide (Diuril®) • Chlorthalidone	In heart failure, because your heart isn't pumping blood as well, your body thinks it doesn't have enough fluid so your body wants to hang on to as much fluid as possible. This can cause swelling and fluid in your lungs. These medications can reduce swelling. They decrease the build-up of fluid in the lungs and other parts of the body by causing increased urination. Loop diuretics are the main diuretic used to decrease swelling. Thiazide or thiazide-like diuretics may be used as "booster" diuretics to make the loop diuretics work better.	• May cause dizziness, weakness, muscle cramps, dry mouth, and increased thirst • May be prescribed with a potassium supplement.

It was pretty clear that the main problem was the lack of empathy with the patients context reflected on a cold and disconnected experience. Based on these findings we decided to create an educational app to replace the old and dry booklets. We also decided that we wanted an app with a very insightful experience that would include engaging mini-games and a healthcare provider mode to allow the healthcare providers to use it as a resource to explain their health recommendations and treatments.

A a UX Designer I was in charge of creating the interaction flows, the content experience and the general graphical interface composition. Part of my work was also to create a seamless transition to the games that were being conceptualized by another team member (game designer).

By testing with early iterations of a user flow and performing discovery interviews with the stakeholders and certain patients, I was able to define the required experience and its components.

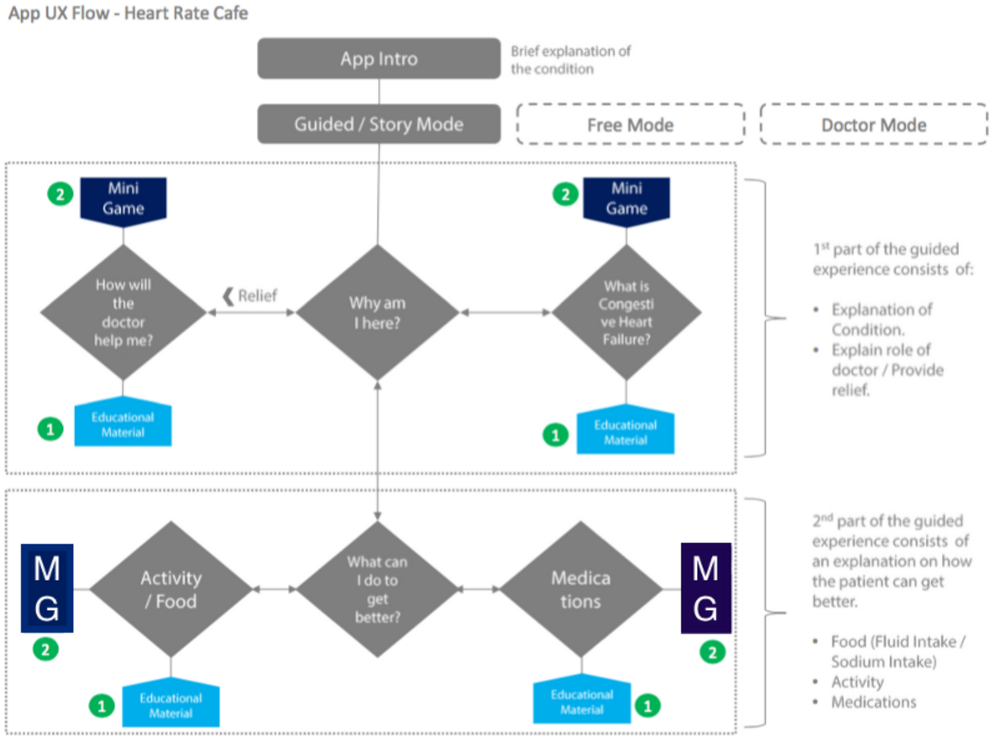
Here is a photo of one of the user interviews that we performed to learn about out target demographic:

In these interviews we used different design research techniques like AEOUT Observations combined with Participatory Activities to encourage users to share more insights. After finalizing the first set of interviews and analyzing the available data I came up with some conclusions and I suggested the team to rule our experience by four main design heuristics:

- **Visibility:** Everything in the app should be visually relevant, high contrast and readable.
- **Redundancy:** In order to tackle any issue with the experience progress I suggested the team to create different interactions and mechanics to move through the app.
- **Feedback Loop:** The app should provide constant feedback of its current state and upcoming states as well as giving a clear indication of progress.
- **Symmetry:** The flow of the app should be symmetrical. This means that the app navigation should hold a pattern and build familiar routes to progress and review the available content.

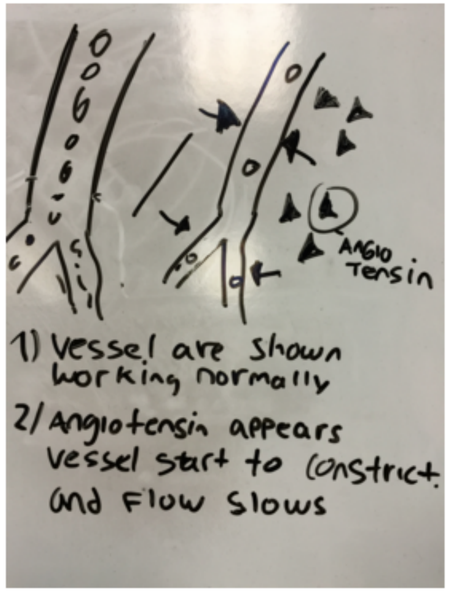
AHN - Savings Heart Project

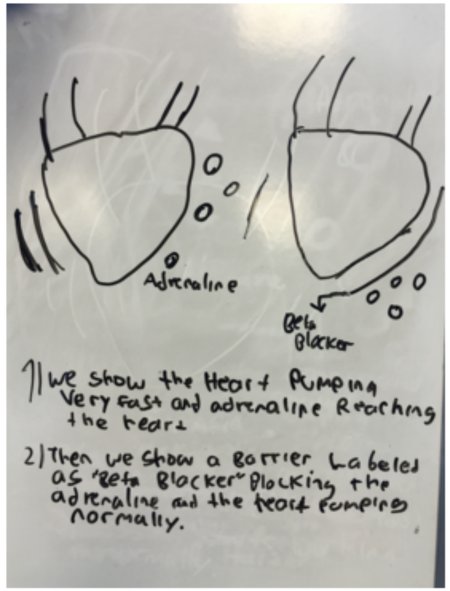
Here is an example of an early user flow that was used to determine the main heuristics:



I also started working on the concept and the tone for the content. The idea was to create very minimalist and consumable content, without being childish and favoring engagement.

I proposed a series of animations that would play next to a small snippets of information. Here are some early concepts for those animations:





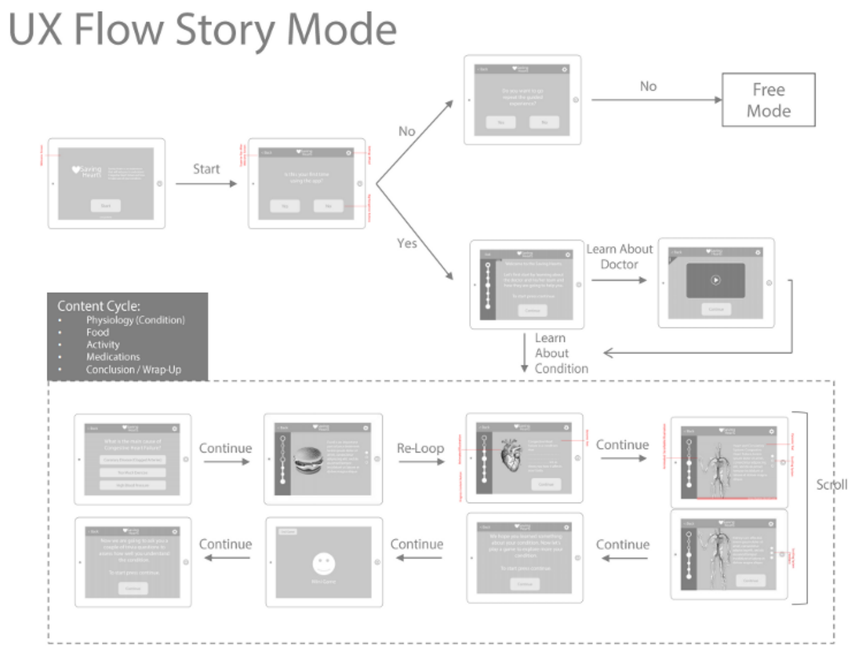
To arrive to the final flow and content proposal we performed several interviews with patients and individuals with similar demographic characteristics. We showed the patients our wireframes and tested for specific issues like readability and cognitive load. This also helped me to define the navigation tree and the user interface components for the app.
It also helped to create a consistent set of patterns that were implemented in the final design.

[whoisjuan]

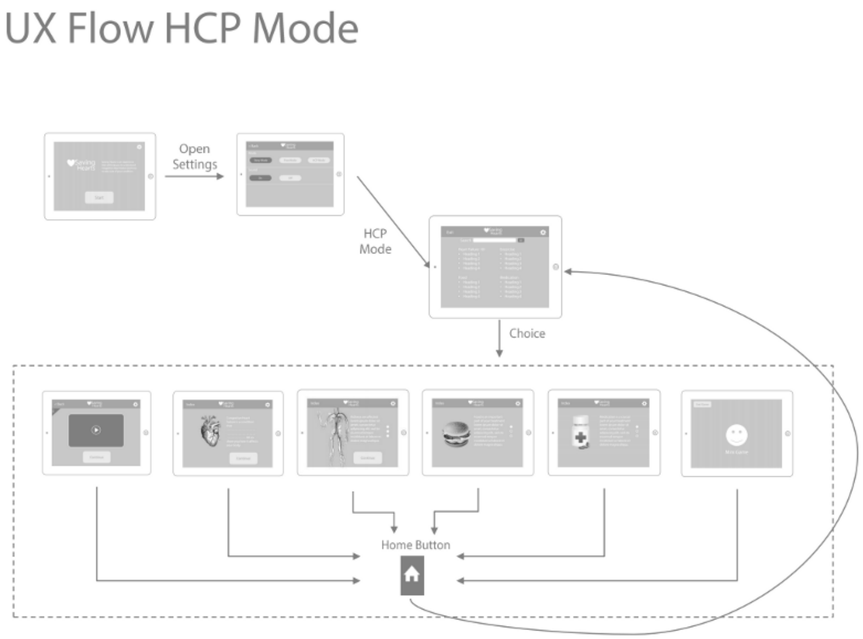
AHN - Savings Heart Project

Here is the final proposed user flow with final low fidelity wireframes:

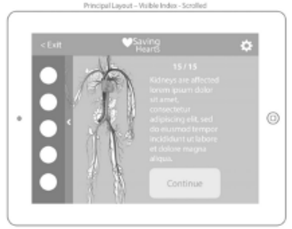
Patient / Story Mode

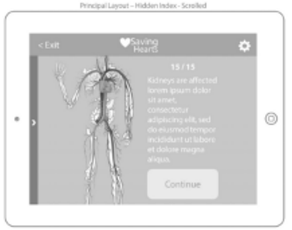


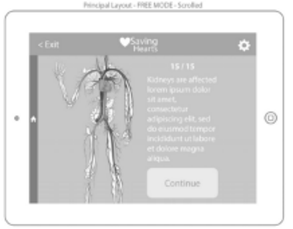
Healthcare Provider (HCP) Mode



Wireframes Examples







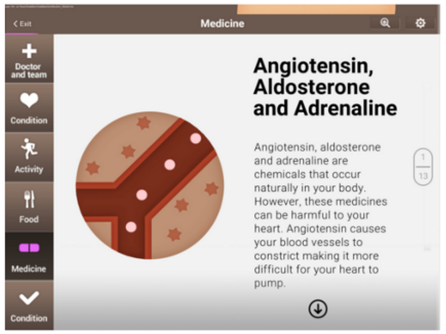
[whoisjuan]

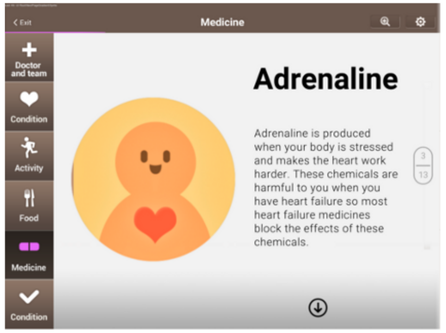
AHN - Savings Heart Project

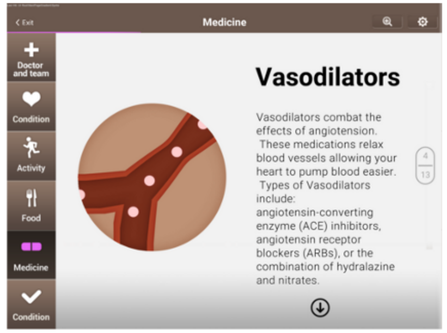
Solution

The final app is a friendly and encouraging experience that allows patients to comprehend their situation. More importantly this app enables a framework of dialogue between the patients and the health care providers. Patients who learn about their disease through this experience are in a better position to understand their condition and assume the responsibility of being compliant with their treatment and self-aware of the impact their life style has on their well-being.

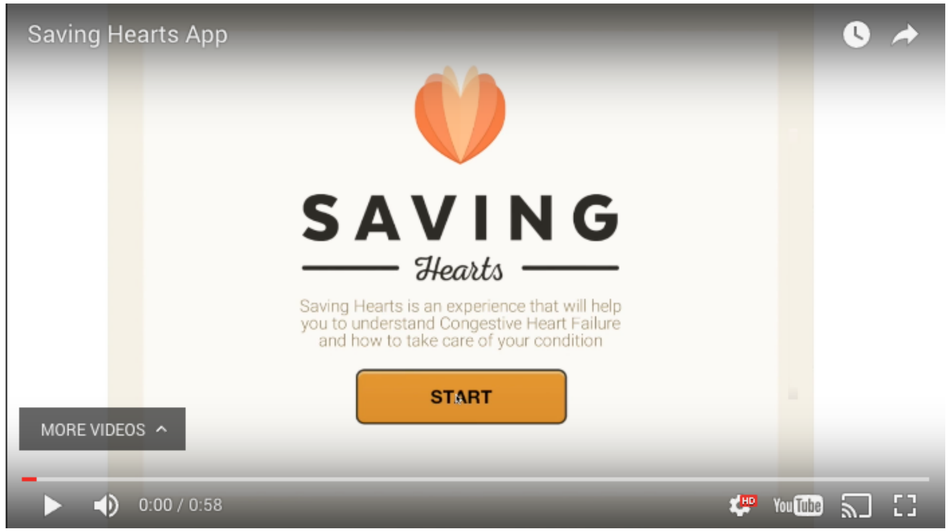
Here is a great example of the clarity and simplicity that we were able to deliver with the final app. The dry medicine guide shown before is now transformed into an engaging narrative :







Finally, here is a video demo that shows the fluidness of the interactions and the use of different navigational elements and feedback loops to increase comprehension and effectiveness:



View Video on YouTube: https://www.youtube.com/watch?v=n2itfdaT8rg

[whoisjuan]

Thanks for Reading :)

For more content please visit:
www.whoisjuan.me