Platelist

Manager: Wen Sun
Design: Gene Oetomo
Development: Omar R.
User Testing: Kyle Qian
Documentation: Gene Oetomo

"Cooking is not an easy task. Our goal is to alleviate the barriers to cooking to make it easier for everyone."

Problem & Solution Overview

The problem our team is trying to address has to do with the enormous amount of time and energy required for meal planning. Many people who otherwise would prefer cooking at home are kept from trying it out due to how much of a chore it is. They would have to keep track of their favorite recipes (many either resort to remembering it or looking it up every time), sit down and plan out their week's groceries, and go out and buy all of it from a store. This is all before even getting to the part where they cook at home. The solution we provide with PlateList is a direct response to all of these time-consuming inconveniences. That way, our users won't feel the weight of all that time invested in cooking at home, a task that should not feel like the annoying chore it is today.

To accomplish this, PlateList provides three services that complement and play off each other: recipe searching/storing, weekly meal planning, and grocery ordering. The synergy of these three services effectively reduces the amount of time and effort it takes to meal plan and prep for cooking at home. Users would search for foods/recipes to be stored in their collection of PlateLists (an analog to music playlists), which can be anything from a collections of breakfast foods to a mish-mash of anything the user likes to eat. Then, the user can 'shuffle' these PlateLists into their weekly schedule, so that the user does not need to personally decide what to eat on a given day. Finally, all the ingredients/foods needed in this shuffled schedule can be put into a cart, from which users can place an order for delivery.

Tasks and Final Interface Scenarios

Simple Task: Platelists

Creating a PlateList is simply making lists of foods grouped however the user wants. Much like playlists in Spotify/iTunes, these PlateLists can be organized by themes like occasion, meal times, food types, or they could just be large aggregated collections of foods the user likes. You can keep track of your favorite foods, and over time create a profile that generalizes your

taste (pun intended) in foods. We chose this as our simple task because during our needfinding phase, we discovered that people who cook don't really have a way of remembering their recipes. If they have a favorite, they usually just write it on a post-it or a piece of paper, which usually gets lost. This function allows a much easier and organized way of keeping track of your favorite recipes. In addition, this simple task serves the overall complex task of recommending you foods and creating your weekly meal plan, which we will talk about next. Platelist is like a music playlist, except instead of songs we have recipes for meals. Each item on the platelist is a recipe.

Moderate Task: Search

Our app allows users to search for foods under a variety of different parameters, namely prep time and ingredients. In addition to just having a normal keyword search, we offer a more advanced search that's actually geared toward what the cook wants, rather than a general Google search. Many times, people don't want to go out and buy groceries, and just want to make a meal based off the ingredients they already have at home, so this task allows the user to input the desired ingredients, and it shows the user what he can make with what he has. In addition, people are always trying to cook for a certain occasion, whether it be a quick meal or maybe a fancier dinner. This function allows people to narrow their search even more so they get the results that they want, so if they're trying to make a quick lunch, they don't have to sift through all the results of really complex recipes to find one!

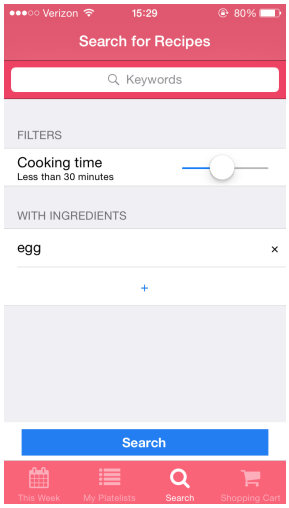
Complex Task: Weekly Meal Plan and Buying Ingredients

Finally, the user should be able to take meals from their platelist and construct a meal plan for the week. We envision the user sitting down on Sunday and planning out their whole week, figuring out what they're having for breakfast, lunch, and dinner for the next seven days. Once the user has planned that out, Platelist has all the information it needs to order groceries with one click, so we include that in the task. Although these tasks are not particularly difficult in the app, we decided that it should be our complex task for two reasons:

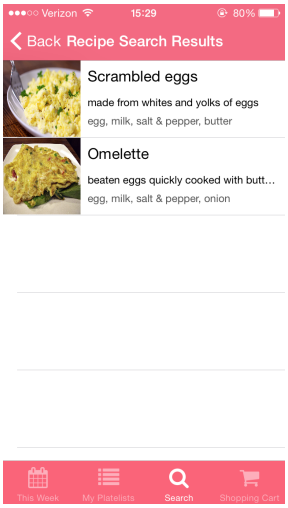
1. The manual alternative is very time-consuming and tiresome. We found that going to the grocery store is a huge deterrent for people to cook. People also struggle with figuring out what to cook in the first place.

2. 'Shuffling' meals as a mechanic is very abstract and its function may not be obvious without further explanation from us to the prototype users. It also requires a back-end algorithm that has not yet been developed and is aimed to do something which people normally can't figure out - to discover new foods similar to their tastes and have a whole week of meals planned out

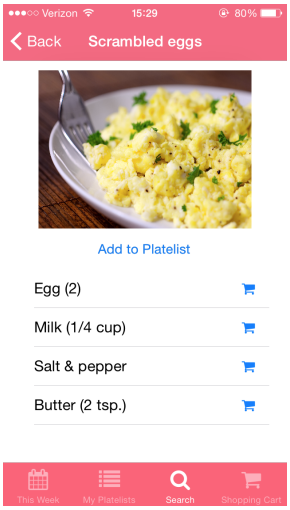
1. Search for a food



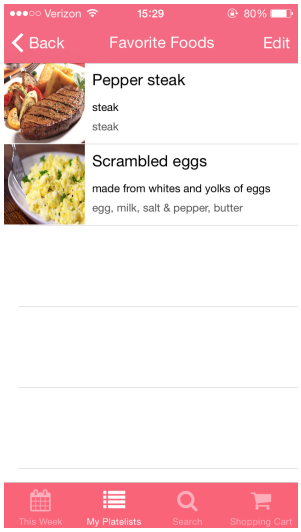
2. Search results



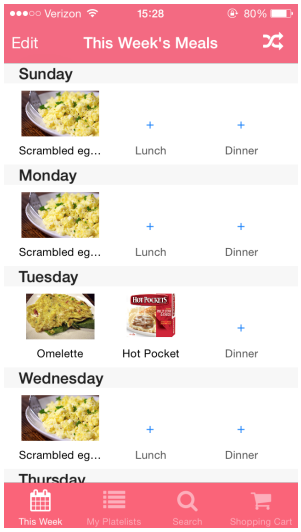
3. Add recipe to Platelist



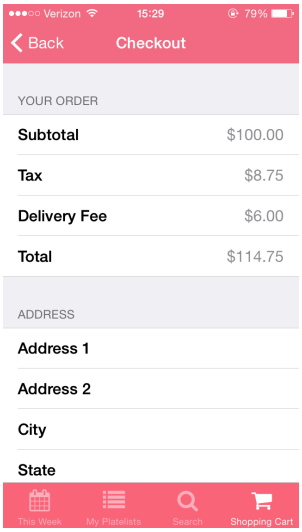
4. View "Favorites" Platelist



5. Shuffle meal plan



6. Order!



Major Usability Problems Addressed

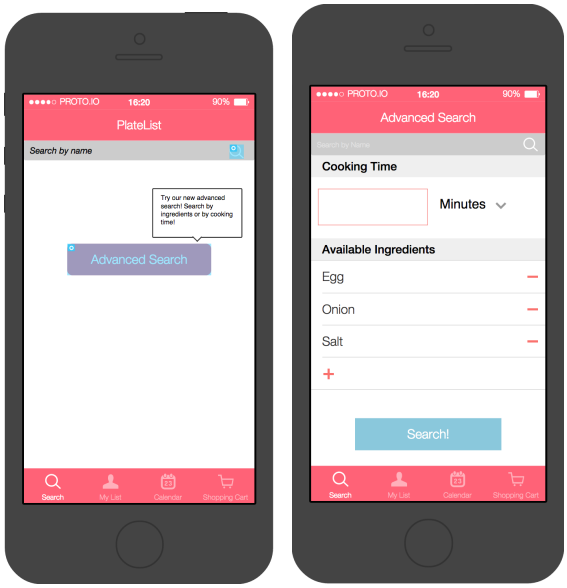
Please note: we no longer have access to our med-fi prototype, so we can't get the screenshots for those.

[H2-4] Swipe to start. Severity: 4

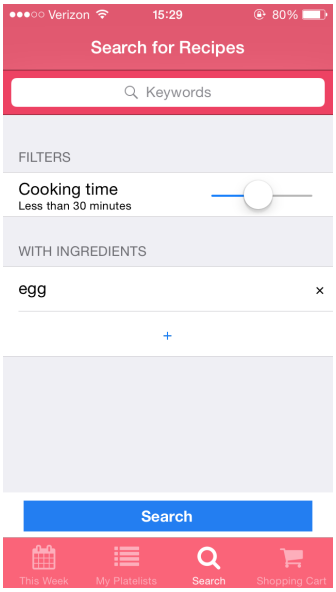
We used to have a start screen which we'd have to swipe to start, much like what the iPhone lock screen looks like. We completely removed this start screen because it was too confusing since people are used to the function of it being a lock screen. Now it just opens on the weekly meal plan page, as seen above

[H2-6] Search Interface. Severity: 3

Old interface:



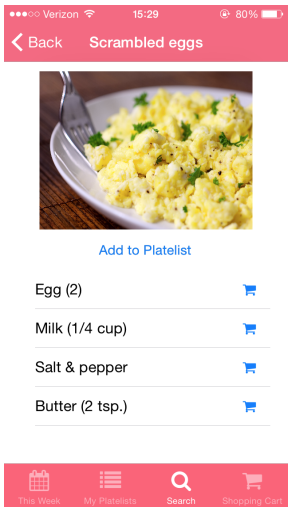
New interface



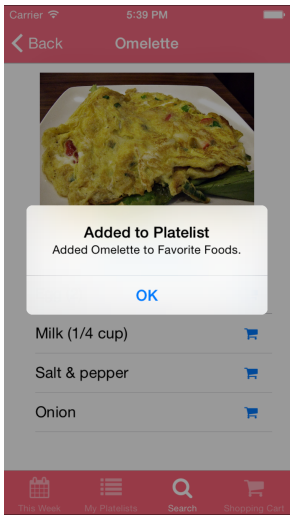
The old interface put too much emphasis on the "Advanced Search" and users had trouble recognizing the normal keyword search. In our new interface, we completely got rid of the old interface screen because we thought it was an unnecessary step to use a search. We also made the normal search much easier to recall its function, so the user doesn't have trouble figuring out exactly how to search for a recipe. The cooking time is also on a scale, which is much easier to adjust.

[H2-1] Acknowledgement of Adding. Severity: 3

Old Interface:



New Interface:

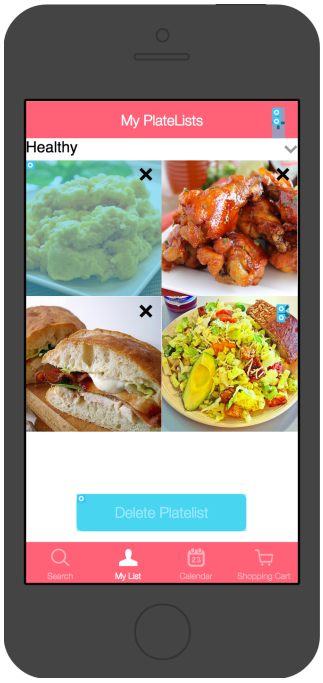


In our old interface, once the user added a recipe to the platelist, it didn't show any confirmation that it was added. On the new one, it shows which Platelist it was in, so it allows

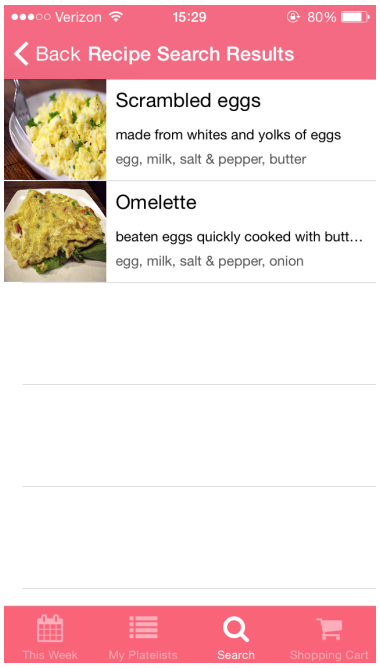
the user to know the system status, and you can then add it to a different platelist (since you know not to add it to the same one).

[H2-6] Captions. Severity: 3

Old Interface:



New Interface:

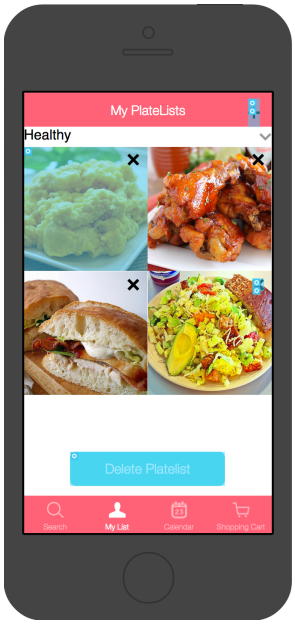


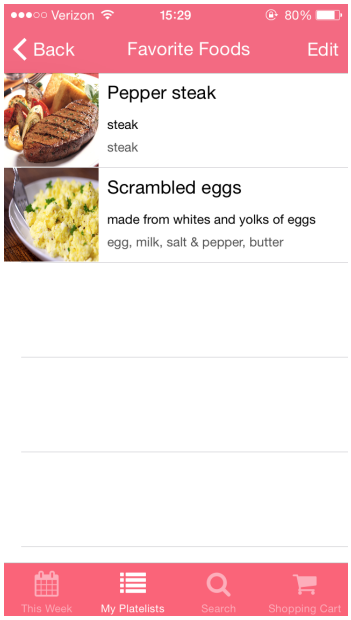
The old interface had no captions under its pictures, and forced you to recall rather than recognize what each item is. Our new interface not only has captions to remind you what it is, it also has a preview of the details so you know exactly what the food you're looking at is.

[H2-4] Delete Platelist. Severity: 4

Old Interface

New Interface

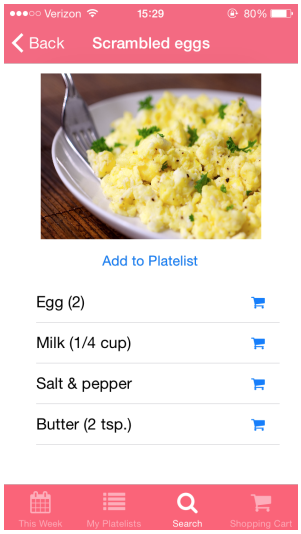




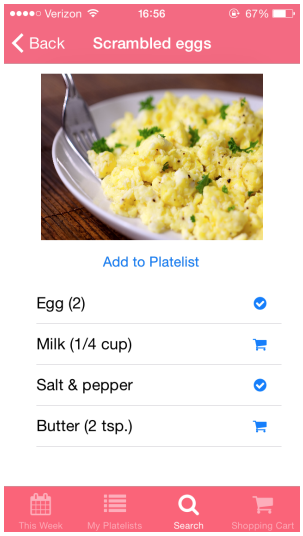
On the old interface, the delete function monopolized the page. There was no confirmation, and it was the big button drew a lot of attention to being deleted. The new one instead forces you to go into an edit mode, which you can see in the top right corner, before you're allowed to delete anything. This prevents the user from making errors, and also allows the user to know which state he is in (delete mode or normal mode)

[H2-1] Adding to Shopping Cart. Severity: 3

Old Interface:



New Interface:



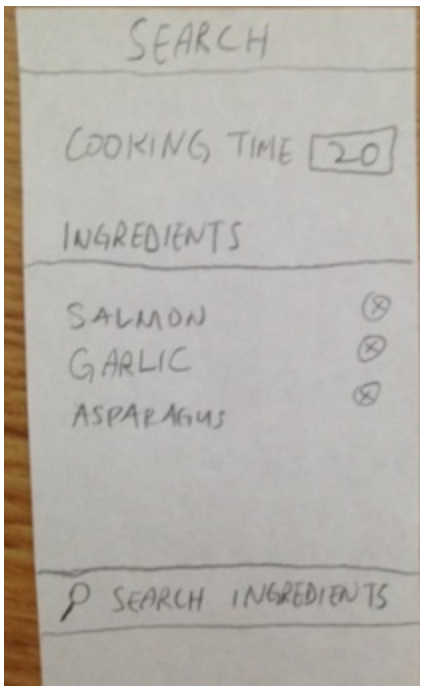
Previously, tapping the shopping cart didn't let the user know if the item was actually added or not. The icon would remain the same and there was no way of notifying the user that it was added. To fix this, we simply changed the icon to a check icon if it was successfully added to the shopping cart.

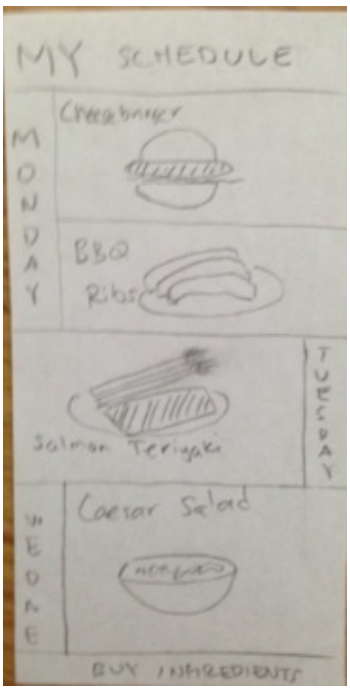
[H2-3] Undo Shuffle and Delete Actions. Severity: 3
Though we did not have time to implement undo -- the implementation would be complicated -- we would plan on implementing undo through iOS's native shake-to-undo functionality (https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/UndoRedo.html), so there would be no visible UI to carry out undo and the interface would look the same as it does in our prototype. You would be able to undo a shuffle and go back to your old meal plan, and undo deleting recipes or PlateLists or shopping cart items.

[H2-10] Help Dialogues. Severity: 3
We disagree with the evaluators that a separate help screen, or help popups, are an idiomatic part of an iOS application. If you look at an app like Yelp, there are no help buttons -- apps are expected to follow an idiomatic grammar of standard iOS applications, with things like Edit buttons for lists. Needing prominently placed help is an indicator of nonstandard app design.

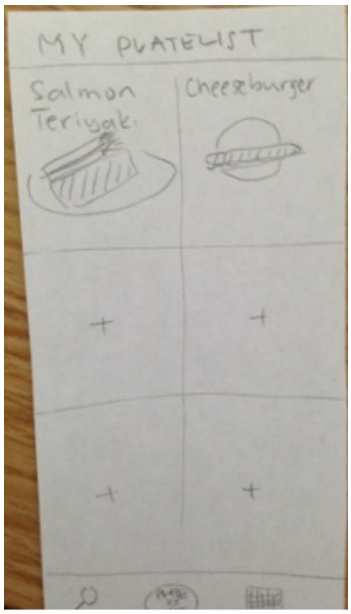
Design Evolution

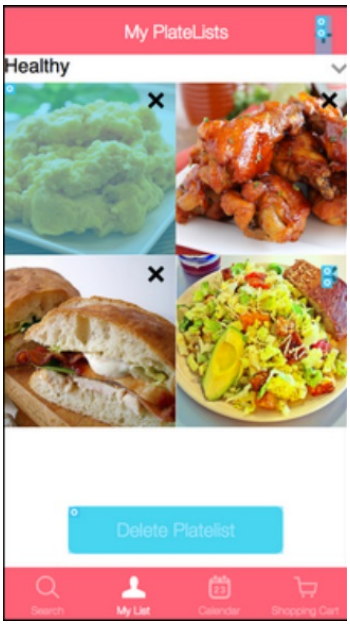
Our first lo-fi prototype started off with very high-level representations of our underlying concept--mainly PlateLists and shuffling. Functions like searching and ordering were still in their infancies. In particular, our search function was based strictly on parameterized recipe/ingredient searching that was not split into simple/advanced. Also, we noticed in our initial user testing that a popular request was the inclusion of pre-made foods such as snacks and Hot Pockets. Furthermore, we had to decide whether to make the weekly schedule page more of an ordered list of food or an actual timetable for cooking. A lot of these issues fell under the umbrella of 'user autonomy,' which is something we kept in mind moving forward.



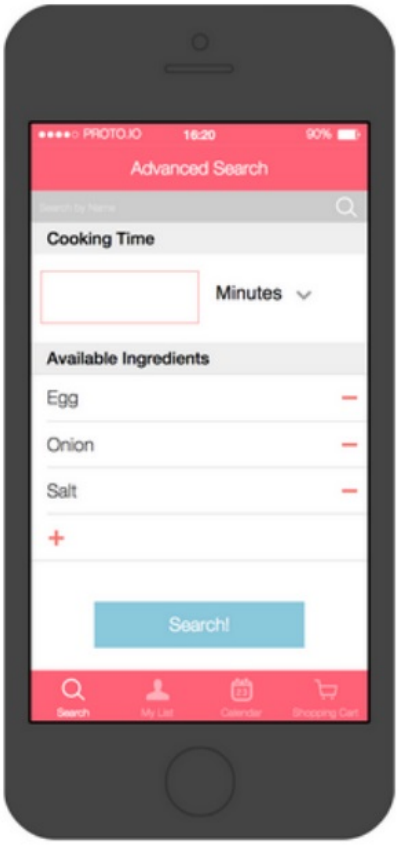


Moving onto the med-fi prototype, another key change we made was the ability to have multiple PlateLists. We realized that users probably wouldn't want to include everything into one gigantic list, and it makes sense to have different selections for different occasions or availabilities.We also included standard iOS elements such as a tab bar, so that users could easily access each part of the app without having to navigate a maze of screens.

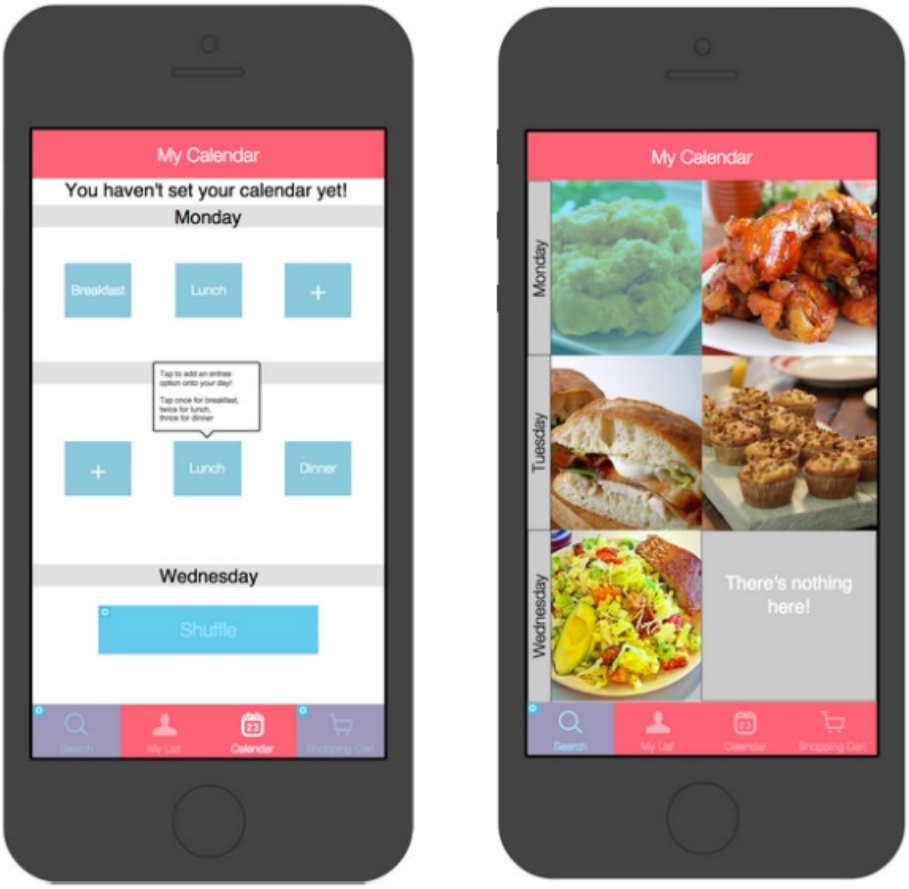




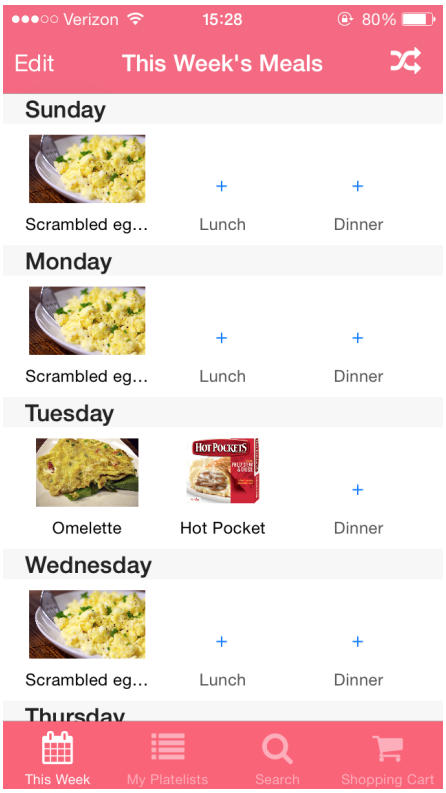
We also implemented a simple/advanced search function that afforded users the autonomy to select what kind of food they would like to put on a PlateList instead of focusing strictly on recipes and ingredients. We felt that this was the most appropriate and catch-all response to our testers asking both for more autonomy and the option to select from simpler choices.



As for the weekly schedule, we decided to keep the list-style format as opposed to a timetable, simply because it would overstep the stated functionality of our app--it's not meant to be used as a calendar to keep track of your time, just a simple interface telling you which meals you have planned. Again, reaching back to the theme of 'user autonomy' that we deduced from our user testing, we decided to add the ability to format daily meal structures, so that each day can have a varying number of meals from different PlateLists.



As we move to our hi-fi prototype, as seen by the screenshots above, our general aesthetic design remains intact--most people liked our choice of color palette and general feel, albeit proto.io itself felt a bit clunky. In general, the biggest design change we made for the hi-fi was better usage of buttons and native iOS functionality. In particular several of our button designs in the med-fi design felt somewhat obtrusive, as noted in our med-fi user test as well as the heuristic evaluation. Buttons like 'Shuffle' and 'Advanced Search' were not put in optimal locations to ensure conveyance of information without distracting the user. Another decision that users liked from the med-fi was the abundance usage of pictures. We decided to take this



even further by combining the calendar settings and calendar view pages, so that users can immediately see pictures of their week without needing to click through an extra options page.

In general, other than functionality improvements, our design changes from the med-fi were mostly aesthetic--we kept and amplified what users already liked seeing and we cleaned up parts where proto.io itself lacked aesthetic flexibility.

Prototype Implementation

Tools

We used Xcode and built a native iPhone app for the prototype. At first, we were developing a mobile Web app using Facebook React and Twitter's Ratchet framework for mobile-like styles, but we found that the experience on a phone browser was too slow, unresponsive, and unrealistic.

Then we switched to iOS. We were impressed by Xcode's storyboard and graphical layout features, which gave us a useful high-level visual representation of the app design. It took some time to get used to the iPhone programming model, but Swift made it much easier. Xcode also encourages you to use iPhone-native UI features; for example, it's much more natural to put a button to edit a list of items in the left of right side of the title bar than to put it somewhere else on the screen.

That attraction to common iOS idioms was both positive and negative. It made it easier to express many UI ideas, and encouraged us to stay close to platform norms, but it also forced us to trade away some novel concepts in the interest of ease of implementation. For example, we wanted to have a feature to zoom into plans for days of the week, but it's much easier to use iOS's table or collection view to just represent that data; by doing a custom implementation, you have to handle a lot of updating and control logic on your own.

Learning a new language and environment was also a constraint we faced with iOS; we often had to give up on clean code and build things inefficiently or with copy-and-paste to make them work quickly for the prototype, since we didn't know the best ways to do code reuse. Still, overall, we found iOS development produced aesthetically great-looking results, including smooth animation, and it was visual and intuitive, especially compared to mobile web development.

Wizard of Oz

During some early demonstrations, we ran into bugs in the prototype -- "clear all" would clear out the meal plan at first, but upon switching to another tab and switching back, you'd see the meals return. There were other issues with data views not updating after changes were made. When demonstrating the app, we'd carefully only show slices of the features that worked while explaining the app verbally.

Hard-coded data

We hard-coded the collection of test recipes which you can search and add to your PlateLists. Realistically, you'd want to make those recipes addable by users, or draw them from some larger online database, or solicit them from chefs as a promotion, so you'd have a big collection. But recipe entry was not one of our 3 tasks; it's not primary to the app, and as long as you have at least some recipes to search for and plan for your week, you can test the app accurately.

We also hard-coded the ingredients in each recipe -- numbers like "2 eggs" and "¼ cup milk" are just strings in the dataset associated with the name of each ingredient. When you add ingredients to the shopping cart, we just store the cart as a list of ingredient name-ingredient amount string pairs. We made this choice because testing the shopping cart UI doesn't really require a complicated, realistic understanding of ingredients.

Finally, the price data in the shopping cart checkout screen is hard-coded, since we don't have a realistic mapping from ingredients to things you could buy from a store. Again, testing the screen doesn't require you to know the exact price of what you're checking out, so we decided it wasn't worth implementing.

What's missing and future directions

First, the app doesn't persist any data. Each app session loses its PlateLists and week plan and shopping cart as soon as it's shut down. We could add automatic save/load like most mobile apps have.

Second, recipes and ingredient data are hard-coded. We could connect it to an external database or recipe site, or add all our own recipes, or even let users manage their recipes from a new screen.

The shopping cart UI is bare, not allowing you to edit quantities of items -- you can only add and delete to the list, which may even contain duplicates, and there's no actual "Checkout" button to tap. We could connect the app to a service like Instacart or Google Shopping Express, and we'd have to add a mapping from recipe ingredients to grocery items. We could also add a button to buy all the ingredients you need in one click, rather than having to do it from each recipe.

Finally, we have a few functions which have places in the interface but aren't implemented: adding custom meals for slots in the weekly view (you can choose the meal, but it won't be added) and undoing actions (shuffle and remove meal -- in iOS, it's done by shaking, so we didn't need to add any visual cues).