
## Problem-Solution Overview

## Tasks & Final Interface Scenarios

The Three Tasks

## Design Evolution

The Five Phases of UI Design

## Major Usability Problems Addressed

Severity 3-4 Heuristics
Other Changes Based on the Heuristic Evaluation

## Prototype Implementation

Tools
Wizard of Oz
Hard-coded Data
Future Implementation

## Summary

# Problem-Solution Overview


Figure 1. The Conceptual Model for Problem|Solution Overview

During our investigation on user needs, we identified three main reasons for severe household food waste: forgetting the expiration dates, being unable to find uses for foods, and buying excessive amount of foods. AfrIdge is a digital solution to solve these problems, and in turn help users reduce household food waste. We envision an artificial intelligence behind AfrIdge that observe and learns the food usage of each user. With AfrIdge, you can monitor the foods' expiration status, optimize your cooking recipe by making the best use of the foods in the fridge, and shop easily with our auto-generated grocery list.

# Tasks & Final Interface Scenarios

## The Three Tasks

### Easy Task

Task Description  Navigate through the four main views via the bottom navigation bar

For the sake of simplicity, we tried to design an intuitive user interface for Afrldge instead of creating a tutorial or help view, therefore it is essential to have an introductory task that helps users get familiar with all views (the three solutions) in the app.

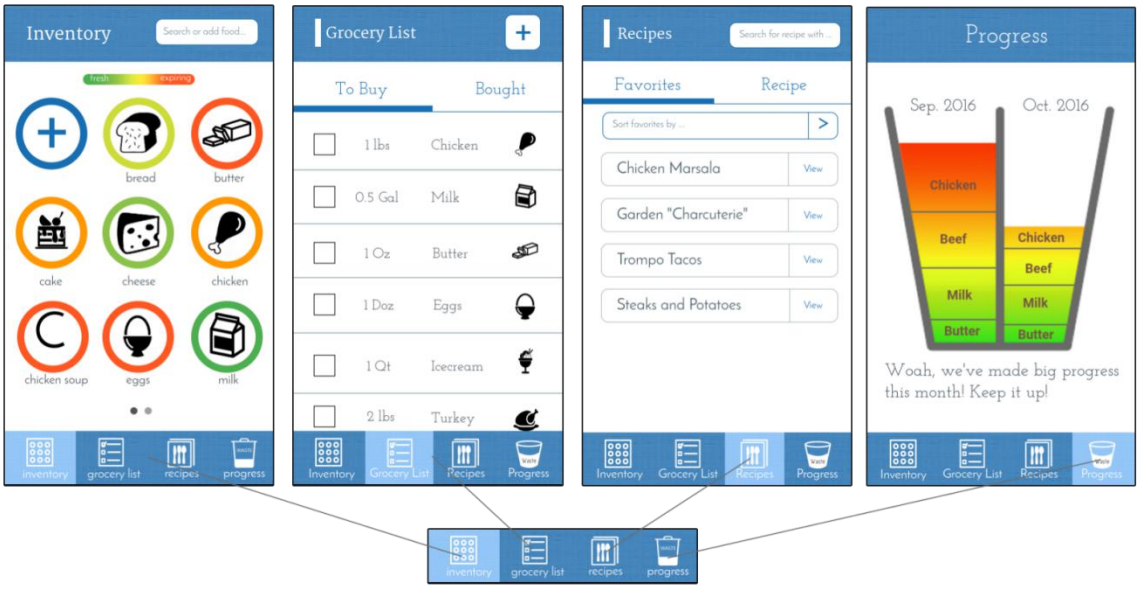



Figure 2. Storyboard flow for using the bottom navigation bar

## Medium Task

Task Description  Shop with the grocery list using the check boxes, adding, editing, and deleting on one food item

We designed this task because it reflects the basic operations on food items that AfrIdge mainly deals with. Users learn to add, edit, and delete food items, and use AfrIdge to assist with their shopping process from this task.

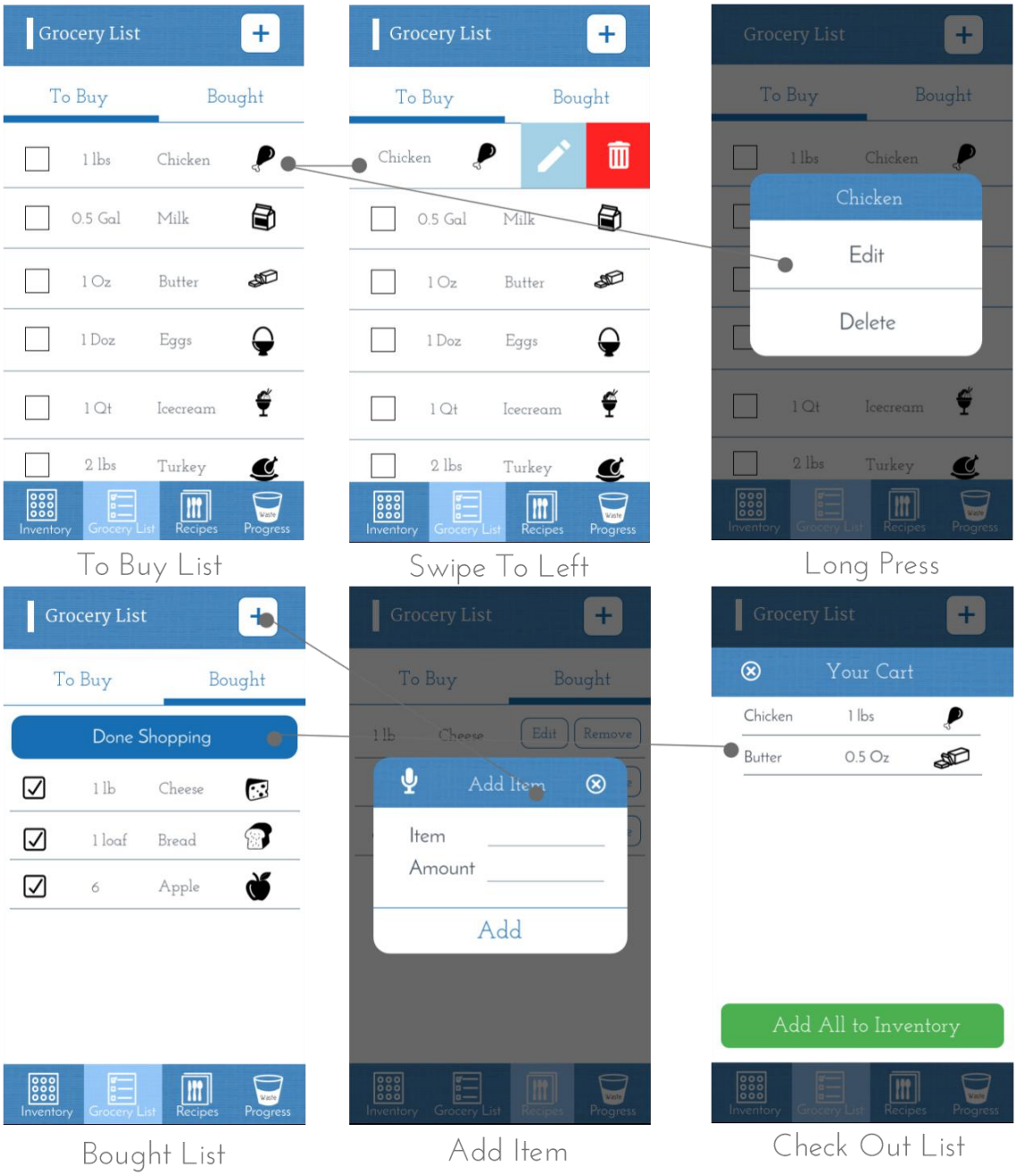



Figure 3. Storyboard Flow for Performing Basic Operations on a Food Item

## Complex Task

Task Description  Discover unlabeled modal windows that interconnect the inventory, grocery list and recipe views

This task not only is built on top of the learned operations from previous two tasks, but also extends the usage of AfrIdge to meet more needs for food management. This task aims to integrate the three solutions on one food item.

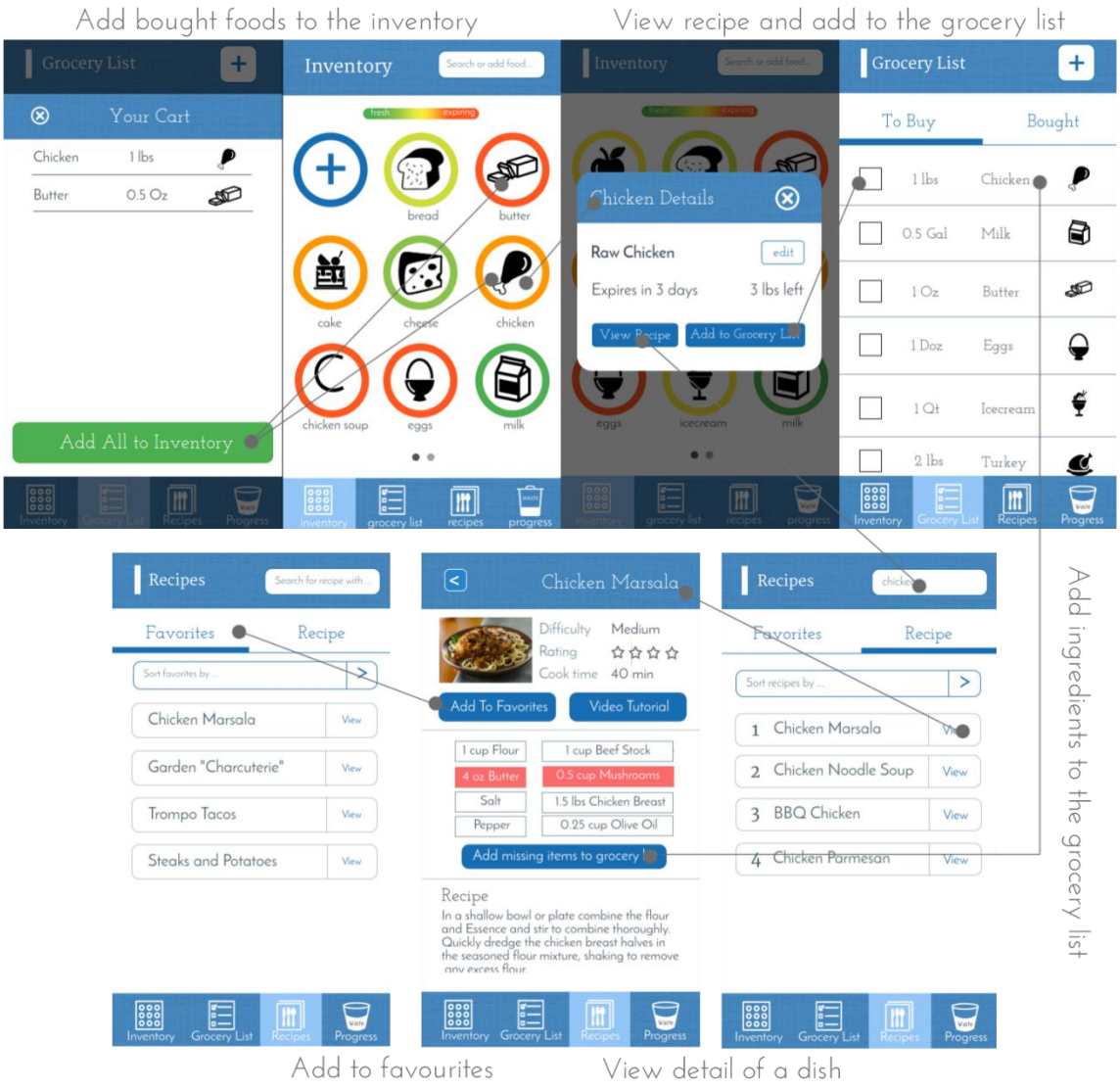



Figure 4. Storyboard flow for discovering modal windows that interconnect different views

# Design Evolution

## The Five Phases of UI Design

### Phase 1 Needfinding

We initially placed our target at graduate students, but interviewed a greater variety of people on their thoughts about their home. After analyzing our interview results, we spotted a general complaint on the food management at home. In our second round of interview, our participants have an age ranging from 20 to 50. Some rent their houses, others own their own ones. We revised our initial Point of Views (POV) for three new ones, and performed the flare-focus practice for several times. In the end we narrowed our solutions down to the best three, and created an experience prototype for the three solutions. Our three experience prototypes include an all-in-one App to communicate with landlord, an app that randomly brings the user to a new outdoor location by suggesting different themes, and a computer vision to monitor fridge inventory.


Figure 5. The Experience Prototype of the All-in-1 App for house renting

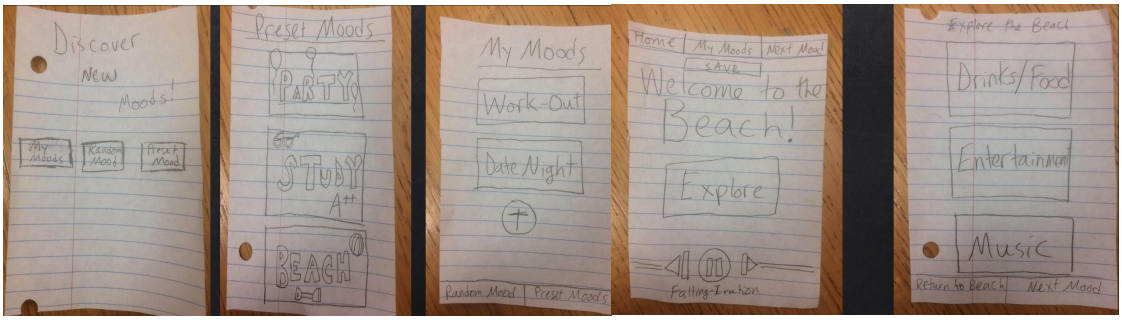



Figure 6. The Experience Prototype of the Theme-switching App


Figure 7. The Experience Prototype of the Computer Vision that Monitors Food Inventory

Figure 8. Testing One Eperience Prototype with a Participant

From testing our three experience prototypes with users, we found that our idea of building a computer vision to monitor fridge inventory is the most feasible. We arrived at this conclusion from a comprehensive consideration on the feedback of experience prototype tests, the influence of the problem that it solves over households, and the social impact of the idea.

## Phase II Sketch

During this stage, we each storyboard a set of UI sketch for the three tasks, from which we will select a preliminary design interface for the low-fidelity prototype.

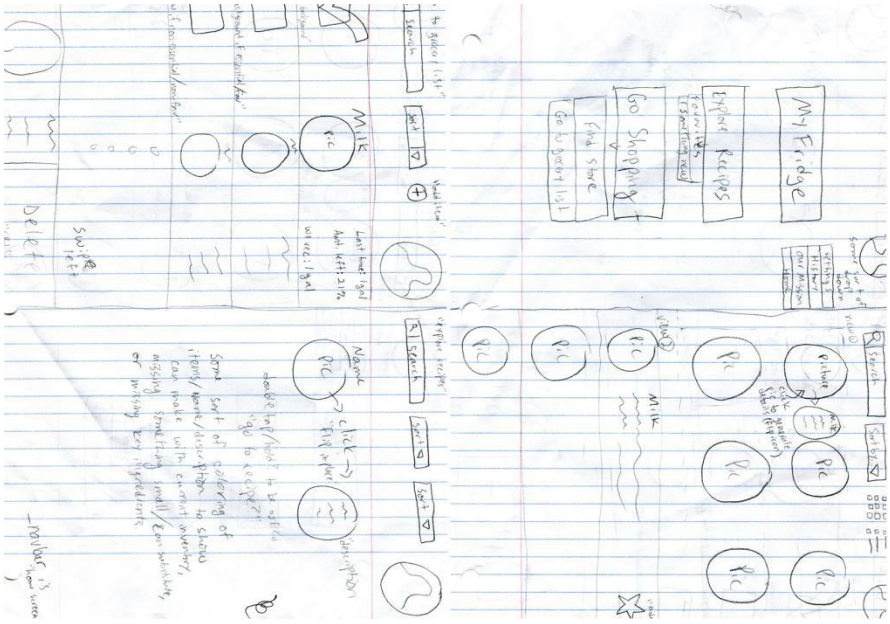


Figure 9. Concept Sketch From Brian

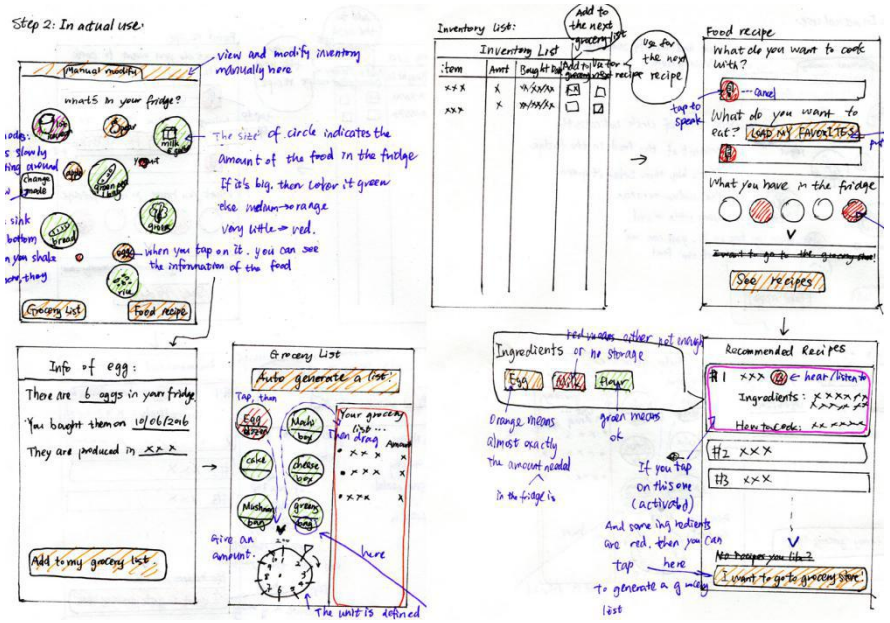



Figure 10. Concept Sketch From Jinglin

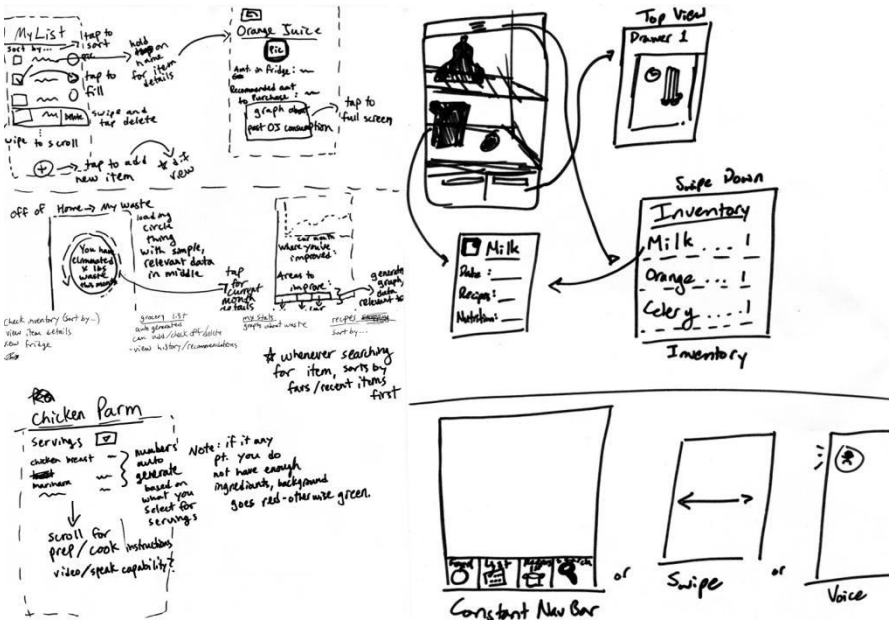



Figure 11. Concept Sketch From Tarek

We pooled the concept sketches, categorize them according to their functions, and did UI sketches in more detail.

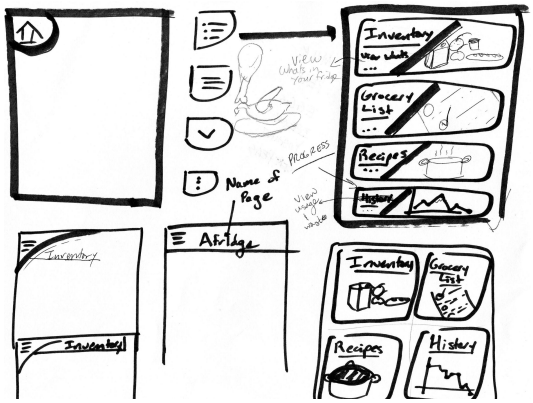



Figure 12. UI Sketches for the Hamburger Icon of the menu

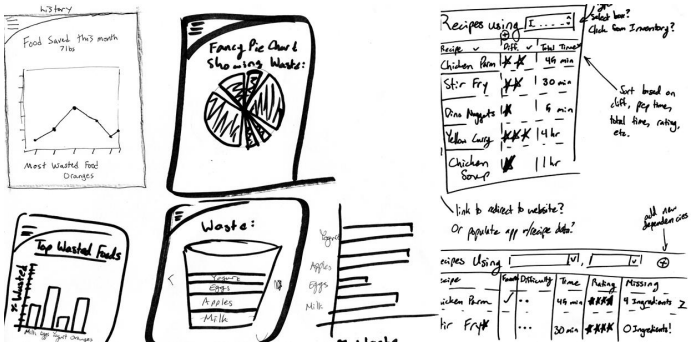



Figure 13. UI Sketches for the Progress and Recipe View

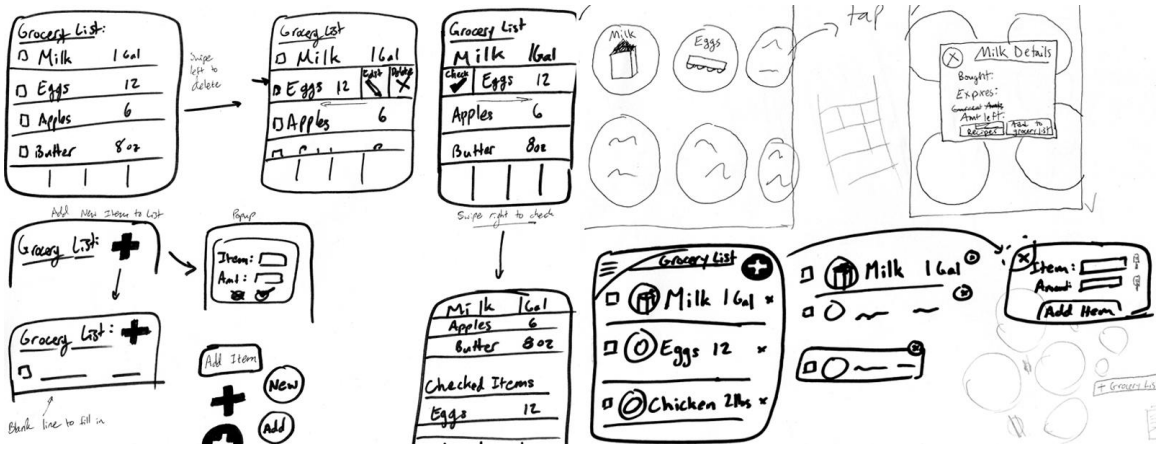



Figure 14. UI Sketch Pool for the Grocery List View

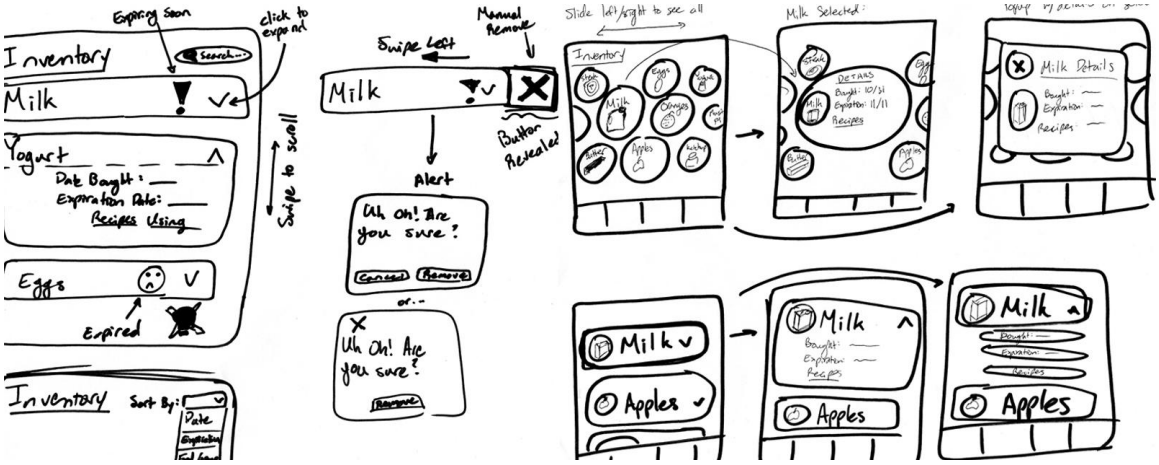



Figure 15. UI Sketch Pool for the Inventory View

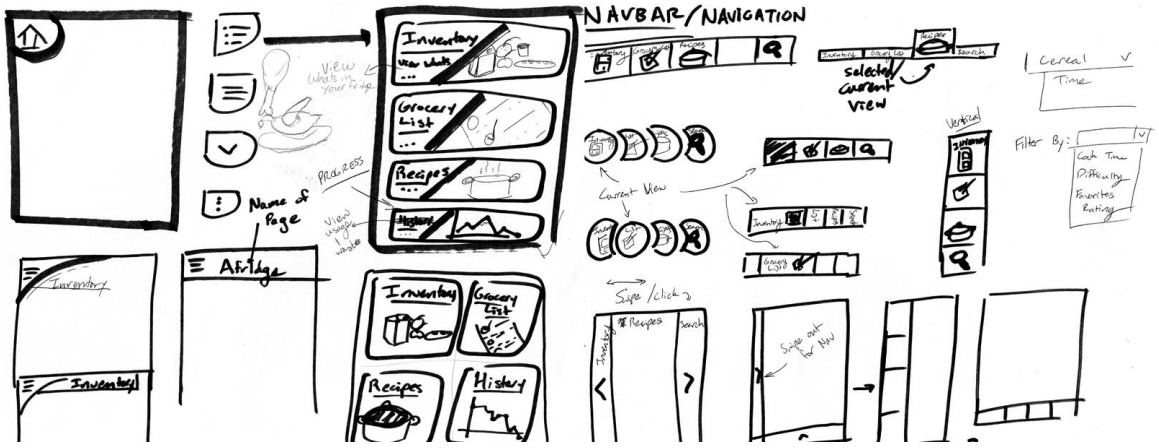



Figure 16. UI Sketch Pool for the Main Menu and Navigation Bar

## Phase III Low-Fidelity Prototype

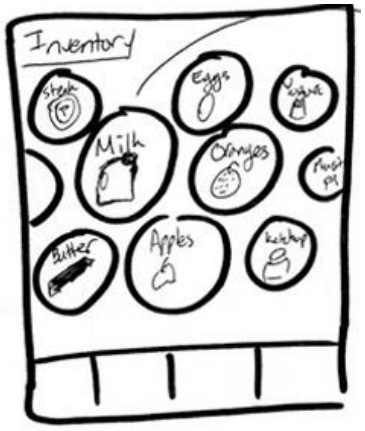
Our third phase of UI design is based on the sketch. We employed the strategy of reaching a consensus in the group on one best design in each sketch pool, and assembled chosen parts to be our low-fidelity prototype.

### The Best Grocery List View


- ✓ Users intuitively use "big plus" icon to add item.
- ✓ The design shows check boxes, pictures of food, name and quantity in a clean way.
- ✓ Swiping left to delete a list item is standard throughout mobile devices

**Con:** Normal checklist style is boring, although it is functional.

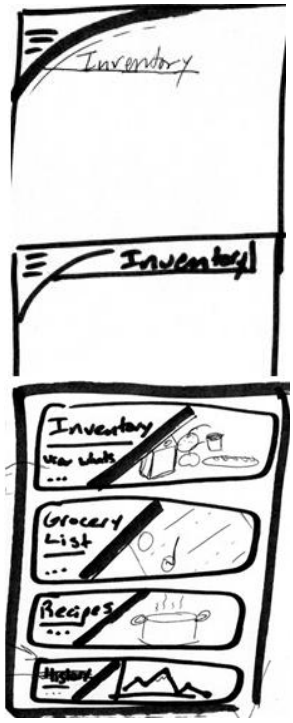
### The Best Inventory View




- ✓ UI is aesthetically appealing with circles compared a list or gridlocked items.
- ✓ Their ball-like shapes lure users to tap and play around. First-time users can easily explore food details without App tutorial.
- ✓ Thumbs tap big circles more easily.
- ✓ Food display looks neater by compactly wrapping items.
- ✓ Circles add potential for clustering, allowing more items to fit in a single page.

**Con:** Circles may not be intuitive compared to a standard list, and clustering could lead to cluttering

## The Best Menu




- ✓ A concave shape of navigation at the corner of the screen uses the minimal space, leaving more room for display of important information.
- ✓ Hidden navigation bar takes up less screen space.
- ✓ The curve adds an aesthetic element.
- ✓ The list items are arranged by their predicted usage frequency.

**Con:** Navigation bar that is hidden takes an extra click to access compared to a constant one that is on the bottom or side.

## The Best Progress View


- ✓ A trash can represents the waste well.
- ✓ A direct reflection on waste amount with drawing is easier to be understood than using a scientific chart.
- ✓ Virtualization of real item can have larger impact.

**Con:** User has to scroll through multiple pages to see all of the data.

## The Best Recipe View


- ✓ Utilize the space of screen well by only showing the name of dish, reducing cluttering.
- ✓ List design allows for easier sorting and filtering in the future

**Con:** Ingredients not included in recipe details, instead outside linking which could be inconvenient although easier to draw and test.

## The Best Add-Item / Food Detail Modal Window


- ✓ The additional popup window helps the display pages look clean and concise.
- ✓ Users can save effort on typing by talking to the program through the mic button.

**Con:** Popup model creates an extra few clicks to view information and then close out of the window.

With the selected designs for each separate part of the app, we created the storyboards for our three tasks.

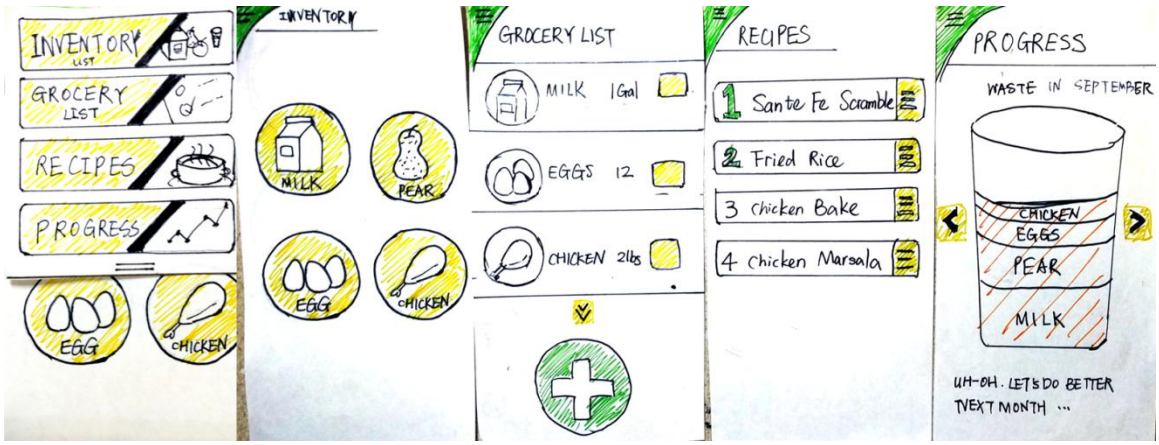



Figure 17. Simple Task Flow for Low-Fi Prototype
Navigate through the four main views from the main menu

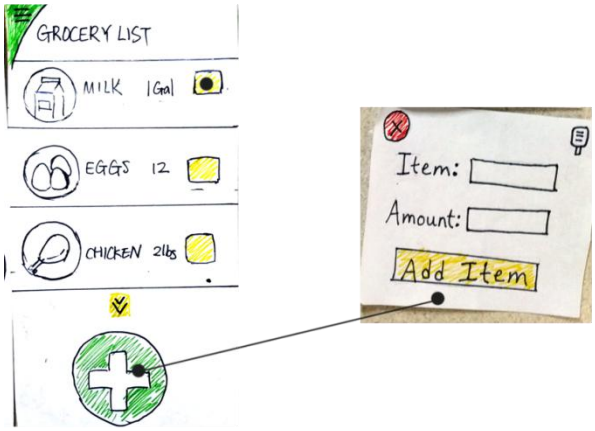



Figure 18. Medium Task Flow for Low-Fi Prototype
Edit the grocery list using the check boxes, adding button, and swiping deletion


Figure 19. Complex Task Flow for Low-Fi Prototype
Discover Popups via unlabeled items in inventory and navigation to other windows through these popups

## Phase IV Medium-Fidelity Prototype

According to the feedback we received on our low-fidelity prototype from three usability tests, we made the following changes to several views.


Figure 20. UI Change #1 for Medium-Fi Prototype
Change the hamburger icon of menu into a constant navigation bar at the bottom of the page

There was evidence in our usability test on the low-fidelity prototype that the hidden menu caused confusion. Therefore we dumped our hamburger icon and a drop-down main menu list design to prevent it from hiding important features. The revision reduces the click time for a user to navigate from one view to another.

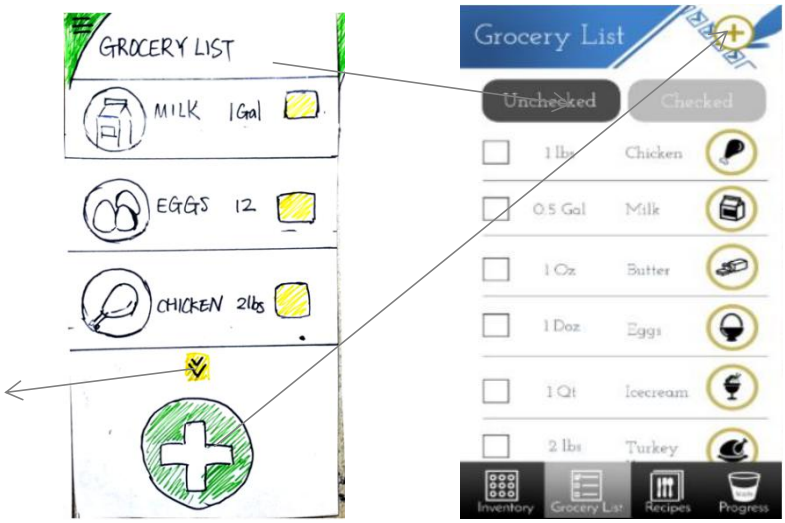



Figure 21. UI Change #2 for Medium-fi Prototype
Move the add button to the top right corner, split checked/unchecked list

Given that one participant mistook the meaning of the big add button at the center bottom of the view, we moved the "+" button to top right to avoid confusion with "submit".
In addition, we split our list into two sub-lists -- unchecked and checked, so that users can perform operations on the food that they still need.
We removed arrows for scrolling removed to avoid confusion of extra button presented in testing.


Figure 21. UI Change #3 for Medium-fi Prototype
Addition of "Ingredients" and "favorite" in the recipe view

In the recipe view, we removed "x" for closing because testers unconsciously clicked on background as they tried to close the window. We added ingredients in recipe and ability to add them to grocery list which allows user to see whether or not they can make the recipe and what they are missing.

Using the new set of UI design, we configured a new set of prototype.


Figure 22. Storyboard Flow for Inventory View in Medium-Fi Prototype


Figure 23. Storyboard Flow for Grocery List View in Medium-Fi Prototype

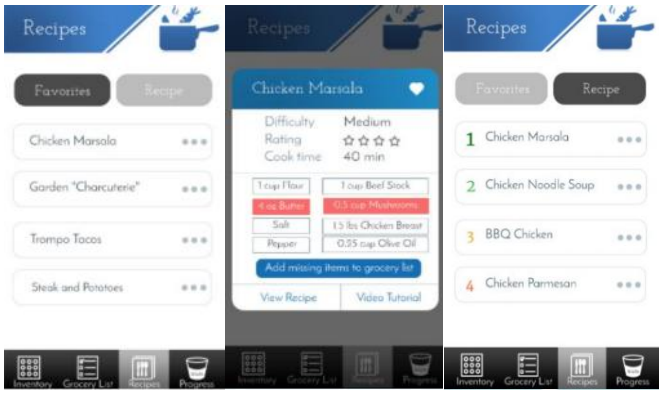



Figure 24. Storyboard Flow for Recipe View in Medium-Fi Prototype

# Phase V High-Fidelity Prototype

# Major Usability Problems Addressed

## Severity 3-4 Heuristics

**#1** [H2-2] Match between system and the real world
Severity 3 / Found by: B, C

**Heuristic** In the grocery list, we have the "checked" and "unchecked" sub-tabs. What do they mean? Does checked represent the items I have reviewed and am going to buy in my next grocery trip, or is it the list of items that I keep checking once I buy to denote that they are bought. Can I uncheck something from the checked list?

**Our Response** To clarify the meaning of two sub-list, we changed the name of the tabs to be "To buy" and "Bought". We added "done shopping checkout" function in "bought" tab to further indicate that it is the list of food items that the user just bought.

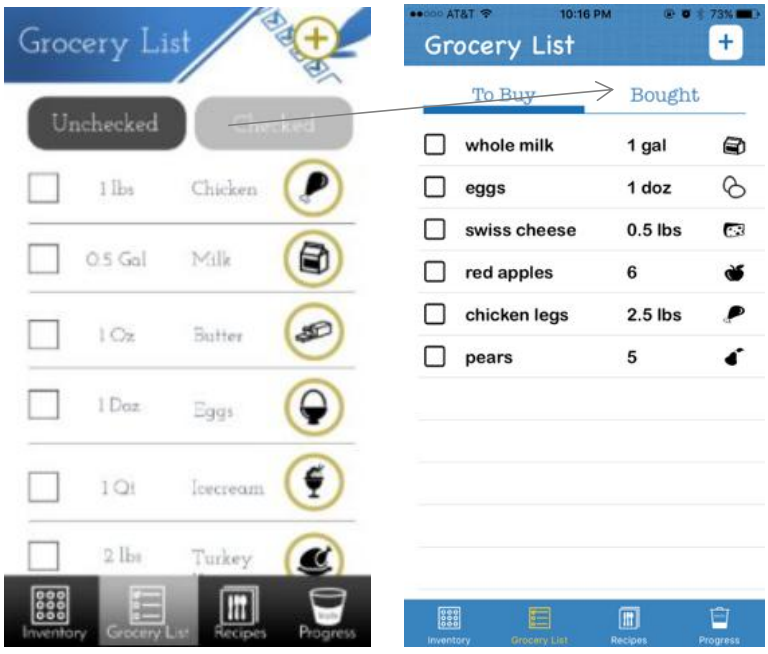



Figure 26. The change on the names of the sub-tabs

**#2** [H2-3] User Control & Freedom
Severity 3 / Found by: B, C

**Heuristic** There is no way to manually enter inventory into the app which were not automatically scanned by the application. User should have the control and freedom to add inventory.

**Our Response** We added a "plus" icon in front of all inventory circles. Users will be shown a pop-up window that allows adding a food item into the inventory by tapping on the icon.

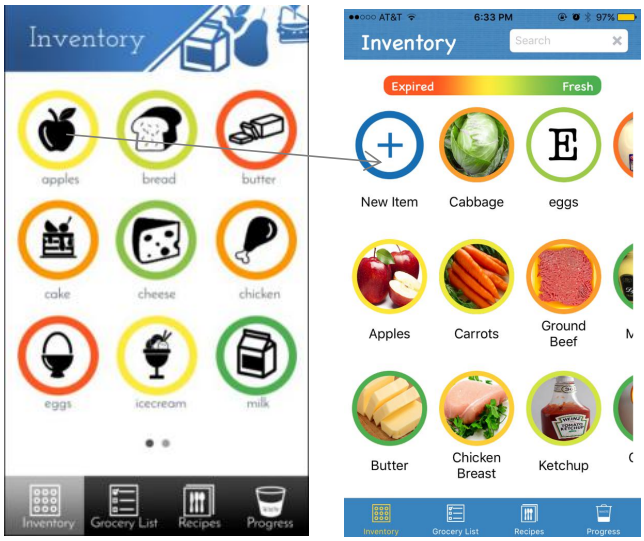



Figure 27. The addition of an "add" button in the inventory view

#3 [H2-7] Flexibility & Efficiency of Use
Severity: 4 / Found by: B

**Heuristic** The inventory list currently uses large icons to represent the food and only when the icon is pressed does the app show simple details of the food. This is not efficient as user wants to know the details of inventory efficiently just by browsing. Mark the details of each inventory on top of or next to the icon would make it more efficient to use.

**Our Response** We believe that it is a trade-off. We lay more emphasis on providing an overview of food in the fridge to users with only essential information. Too much detailed information on the same view will cause cluttering.

**#4** [H2-5] Error Prevention
Severity 3 / Found by: B, C, D

**Heuristic** In the inventory details page (for chicken), it shows the bought date and the expiry date. There is no way to change the expiry date if it was added wrongly. With wrong dates, the app would estimate stale foods wrongly and would send unreasonable notifications. The inventory details page should also have the option of editing.

**Our Response** Thanks to this heuristic that made us realize our missing of error prevention. We added the "edit" function for the food detail so that users can edit the amount and the expiration date.

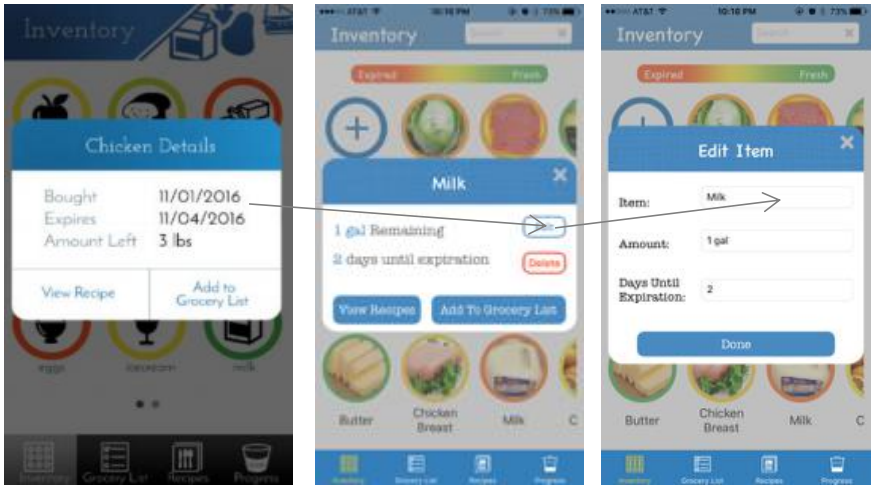



Figure 27. The addition of an "edit" function in the food detail pop-up window

**#5** [H2-2] Match Sys & World
Severity 3 / Found by: B

**Heuristic** The inventory list currently shows icons of food, which might be confusing to the user since it takes longer to tell which type of food that actually is from the black icons. The system representation of inventory doesn't match the real world. Changing icon representations to photo thumbnails of the real inventory photo would resolve this issue.

**Our Response** We understand that from the cognitive aspect of human, using real photos helps our users recognize the food better. Therefore we replaced the black icon with photos of food.


Figure 28. Changing icons to photos

**#6** [H2-2] Match Sys & World
Severity: 4 / Found by: B, D

**Heuristic** The recipe card shown has no image of the cooked food and the ingredients, adding to the difficulty for user to understand and doesn't match with the real world representation. Adding images of what the cooked food would look like would improve the usability for the user.

**Our Response** We identified the problem, and added a photo of the dish at the top left corner of the dish detail.

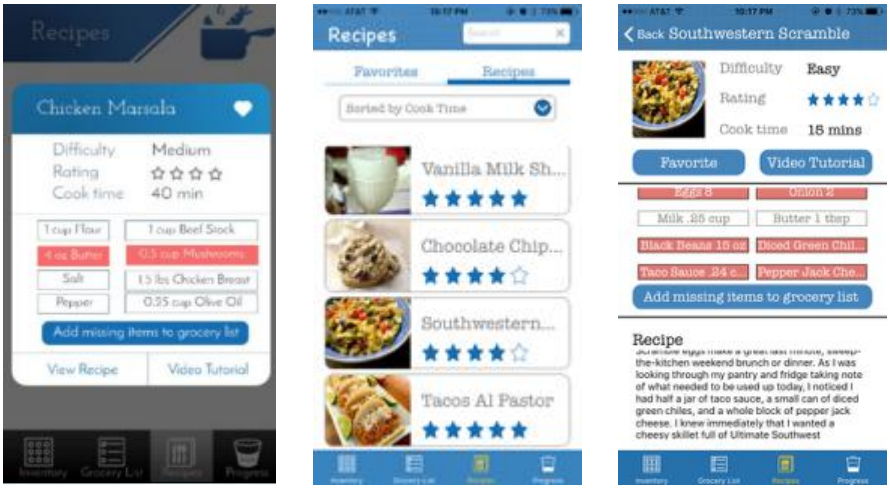



Figure 29. Adding photos of the dishes in recipe details

**#7** [H2-5] Error Prevention
Severity: 4 / Found by: D

**Heuristic** On the inventory screen, it seems like all the items are placed into very broad categories such as chicken. What happens if there is both raw chicken and cooked leftover chicken in the fridge? It is unclear if your app would create two different items in the inventory or combine them into one.

**Our Response** We understood the problem, but considering the possible cluttering of information caused by the "sub-inventory" idea in the suggestion, we decided to create two different items for different categories of food. We implemented an incremental search function to compensate the disadvantage of separating the foods that are under the same broad category (I.e. chicken soup and chicken breast) apart.


Figure 30. Splitting different sub-categories into individual food items

**#8** [H2-5] Error prevention
Severity: 3 / Found by: D

**Heuristic** In your grocery list, you can click on "chicken" which will lead to details about the item. From here, you can click on a button that leads you to the recipes section. Once there, you get a list of recommended recipes for that item ("chicken" in our example). However, from this page, you can click on the "Favorites" button and this will lead you to recipes that don't contain chicken. This is an error because the user was trying to find a recipe with chicken, but he stumbled upon recipes that didn't have chicken.

**Our Response** We appreciate the suggestion provided along this heuristic. We created a variable indicating the ingredient keyword that the user is using, so that AfrIdge filters out any recipes in the "Favorites" tab that don't have that item as an ingredient.

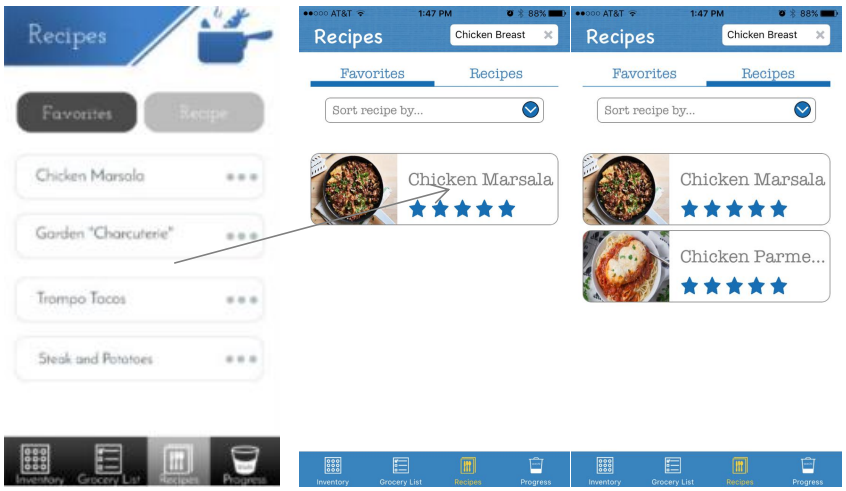



Figure 31. Filter out any dishes without the search keyword as an ingredient

We valued other heuristics of severity ranging from 0 to 2, and accordingly modified our UI as well.

## Other Changes Based on the Heuristic Evaluation

### Inventory View


Figure 32. Other UI changes on the inventory view

**Change #1**
Redesigned the background of the title bar and the bottom fixed navigation bar for better consistency in theme

**Change #2**
Added a simple but clear explanation on the colors of the rings

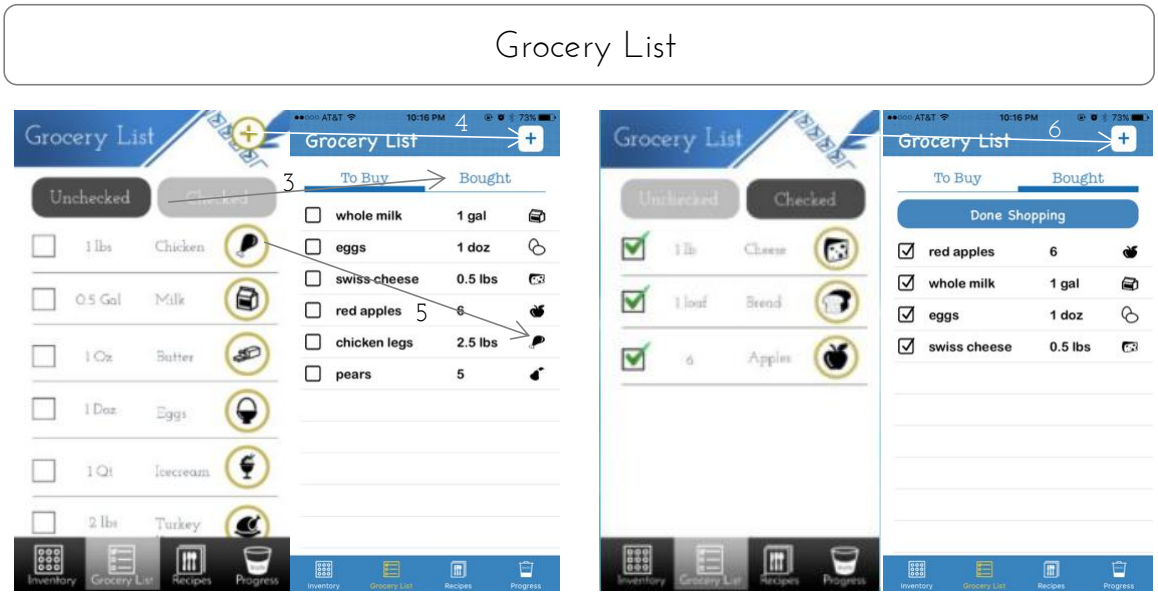



Figure 33. Other UI changes on the grocery list view

**Change #3**
Changed design of highlighting active tab to avoid confusion aroused by color

**Change #4**
Redesigned the color and shape of the "add" button to make it more noticeable

**Change #5**
Removed rings around the food icon to avoid confusion with the inventory

**Change #6** Users can add items to grocery list from "bought tab" as well

## Food Detail Pop-up Window

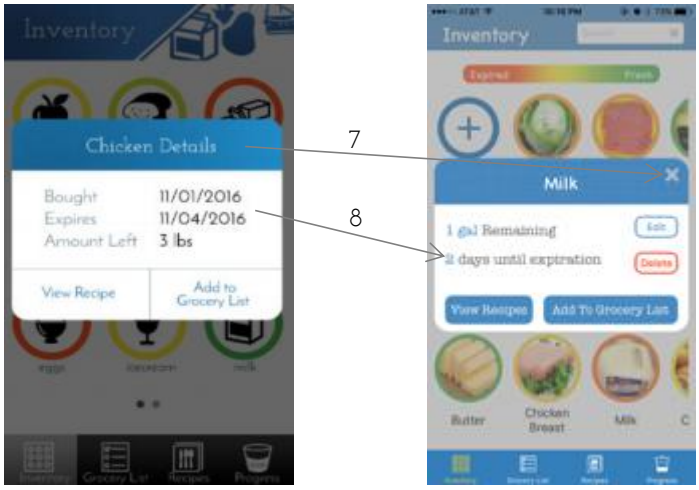



Figure 34. Other UI changes on the food detail pop-up window

**Change #7**

Offered users an alternative "close" (x) button for more flexible use

**Change #8**

Adopted a more humane way to show the expiration date so that users do not need to do subtraction on date in their mind

## Recipes

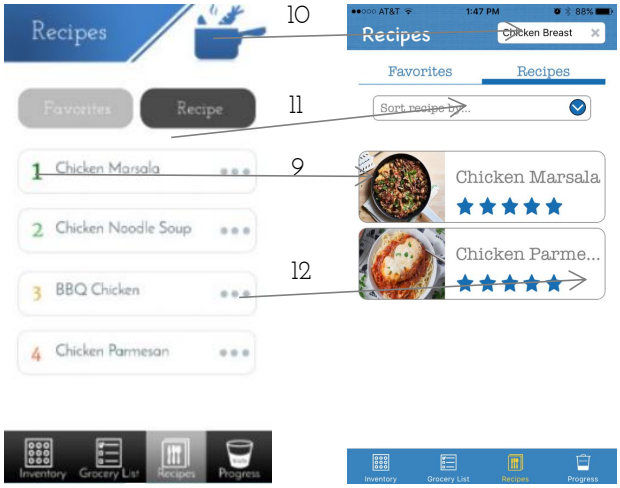



Figure 35. Other UI changes on the recipe view

**Change #9**
Removed color on numbers to avoid confusion. We replaced number ranking with dish photos instead.

**Change #10**
If users are led to recipe view from the food detail popup in the inventory view, the search bar will auto-fill the food name as a keyword.

**Change #11**
Added a sorting function for more complete functionality

**Change #12**
Removed three dots to avoid confusion on its meaning.

Recipe detail

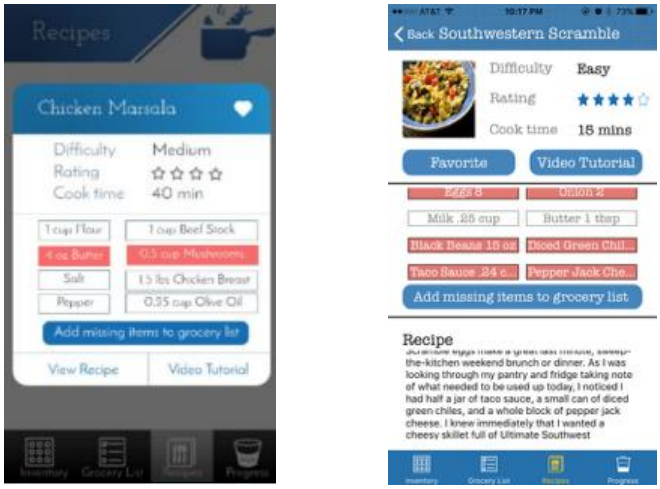



Figure 36. Other UI changes on the recipe detail

**Change #14**
Made the detail of a dish a new view with a "back" icon because the original popup window contained too much information.

**Change #15**
Included an image of the dish for better usability.

**Change #16**
Replaced heart icon with text label for clarity. If the food is already in favorite, then this button will show "unfavorite".

**Change #17**
Removed "View Recipe" button and instead include the recipe detail here to save users a click.

# Prototype Implementations

## Tools

We built our Hi-Fi prototype as an iOS application runnable on the iPhone 6/6s using **Xcode 8** and **Swift 3**, as well as **GitHub** for collaboration. This allowed us to create an interactive prototype in which users could view and manipulate data in multiple views, both within and across these views.

In Xcode, the storyboard portion made it extremely easy to see exactly what we were expecting to pop up on our screens, and also allowed us to simply map objects in particular views like buttons or tables to their respective variables in code. This made dynamic styling and responding to click events very intuitive. The storyboard also allowed us to visualize the task flows on screen, since we could organize and place the views as we liked, with arrows representing exact navigation back and forth between views. Built in functionality like tab bar controllers and table views proved to be very useful as well.

At first, we used one single storyboard in which we placed all four of our main flows (Inventory, Grocery List, Recipes, and Progress), organized by row to group subviews within each section. Although this layout showed us everything in one place, it made collaboration through GitHub a nightmare. Two people editing different parts of the storyboard on different branches in Git created unsolvable merge conflicts, forcing us to completely eliminate all but one of the branch's storyboard changes, leading to bugs in the code and wasted time, effort, and work. Xcode's storyboard functionality, as we learned, works very poorly with respect to collaboration, but we were able to find a workaround. After struggling through the first set of merge conflicts, we split our project into separate storyboards for each of the four task flows, and divided our work among them so that no two people were working on the same chunk. This not only helped us avoid merge conflicts, but also cleaned and decluttered our views, showing us only the specific views that we would work on at a time.

Constraints in Xcode were also often painful to work with. When setting constraints, objects would often disappear if only the position was set, or fly to the top of the screen if only the size was set. This proved to be a difficulty when attempting to style our views, with conflicting constraints often popping up and leading to bugs and ambiguous layouts on the views not directly translating to the visual presented on the device or simulator.

Swift 3 proved to be a very valuable tool, and was not super hard to pick up over time considering the nonexistent iOS experience anyone on our team had before entering this class. Swift allowed us to programmatically edit styling on the page, like colors and border radii, which made the realization of our design possible. The connection of buttons and tapping actions to specific functions in code also made reacting to user interaction very straightforward, although there were a couple times were tapping interactions would overlap, leading to bugs that needed to be fixed. Swift also allowed us to programmatically lay views on top of each other and adjust their positions, making pop ups and other modal views like dropdown menus and hidden buttons possible. This was especially important for our Inventory view, which used pop ups for all of the interactions with food items, including viewing details, editing, deleting, and creating. We also used built in animation capabilities, although confusing to learn, in simple ways to augment user interaction and show a response on certain button clicks and checking off items in the grocery list.

Swift, unfortunately, is not the only language used for creating iOS applications, with Objective C also seeming to be a popular option. When searching online to learn how to implement specific parts of our application, it was often hard to find examples that used Swift. This, along with Xcode's limited tab-bar editing capabilities through the storyboard, made it very difficult to figure out how to customize our tab-bar in ways other than simply setting colors. Xcode didn't allow for a tab-bar's background image to be set, only a background color, and didn't have any options for styling selected and unselected items other than changing colors. Time limitations and limited resources also prevented us from figuring out how to fully customize our tab-bar using Swift.

## Wizard of Oz

One of the main concepts behind our application, use of computer vision to sense what is in your refrigerator and what you add and remove, had to be worked around since it wasn't something that could be feasibly implemented for our prototype. This was especially apparent in the initial population of data, and the interactions between the Grocery List and Inventory. After going to the grocery store and buying items, the app should theoretically automatically update your inventory when you add the new items into your refrigerator, so in order to mimic this we added a way to pass data from the bought items grocery list back to our inventory. After filling out the grocery list in the application, users can click on an "Add To Inventory" button that automatically clears the grocery list and adds the bought items to their inventory, as if mimicking physically adding items and them being scanned into the system.

## Hard-Coded Data

All of the data initially shown in our application is hard-coded. This included all of the inventory items and their amounts and expiration dates, grocery list items and their amounts, and recipes and their cook times, ratings, ingredients list, details, description, difficulty, and video link. For expiration of inventory items, we used a set scale that did not represent individual items' shelf lives, such that an item that is fresh and supposed to be eaten the day it is bought but only lasts for a couple days will show as just as close to expiring as an item that can stay in your refrigerator, untouched, for months. Because our application is extremely data heavy, we only chose a select few recipes to display, since everything was hard-coded rather than programmatically generated. The progress view was just a static image with the goal of showing the presence of tracking waste and allowing users to compare results from this month and last month.

## Missing and Future Features

As our application is now, none of the data added, edited, or deleted in a single use is permanently saved. Once the app is completely closed, it will show the same initial settings upon reopening. For the future, migrating to a way of permanently stored data is necessary to make the app usable, even without the addition of computer vision and machine learning and just with manual entry. The AI aspects of the application are missing here of course, but were taken into account through hard-coded data and wizard of oz techniques mentioned above. Images for any user-added items in the grocery list and inventory are also not present, but in the future we could use an internet search query to find images based on user's entry, or ideally use the cameras for computer vision to capture these images. The recipe selection is also limited due to them all being hard-coded. These could be fetched and compiled from multiple sources like recipe websites and youtube videos in order to fully populate this section and allow users a wide variety of recipes to choose from. Waste-tracking capabilities are also missing from this implementation, but could be added in with the computer vision aspect of tracking expired items.

# Summary

The amount of food waste in the United States, and across the world, is appalling, and we believe that our product idea can help take a chip out of this problem by reducing users' food waste by tracking expirations and helping them tweak their shopping. We set out with the goal of creating an application that would allow users to monitor their food items and usage, and after many iterations and weeks of work we accomplished the task of making our interactive iOS hi-fi prototype for doing just that. From start to finish, the iterative design process taught in this class, along with feedback from interviews and classmates' heuristic evaluations, pushed us to refine our idea and design multiple times, eventually leading us to land on a final implementation that we were happy with. Over time, our tasks became more focused and streamlined, and our UI was cleaned up and made more intuitive. Although there are certain things missing from our prototype, we were able to pack a large amount of functionality into our application in a very short time with no prior iOS programming experience. The concept AfrIdge presents is something that will become a standard of the future, as people are moving more towards using machine learning to help optimize and augment daily activities and basic needs.