Foodwatch

Sean Miller; Group Manager
Brandon Zahl; Designer
Mike Silver; User Testing
Kyle Hornberger; Writer
CSE 440; Autumn 2009
http://cs.washington.edu/education/courses/cse440/09au/projects/foodwatch/documents/Interactive_Prototype.pdf

Problem and Solution Overview

Wasted food costs American households approximately $43 billion a year. To prevent this, we are developing a better way for people to be reminded and keep track of their food in terms of quantities and freshness, so that they may improve their food buying habits. Foodwatch will target these issues by giving the user an inventory of their food, reminding them when their food will spoil soon, tracking their food expenses, and informing them of ways they can store their food to increase its longevity.

Our goal is to provide tools for customers to police their own food storage habits, guide their purchases, and motivate them to save money and reduce food waste.

Tasks

In this phase of design, our tasks went through some revision. Our previous first and second tasks were very similar, and while Foodwatch would ideally include all of them to aid the customer in policing their food waste, we decided to mix things up a bit. Our new set of tasks shows more functionality in the customer's improvement of food management over time, and is better oriented towards long-term goals.

Task 1: Changing update settings [Easy]
To update you on what food you need to use soon, Foodwatch sends out a periodic email containing all items in your inventory that should be used promptly. In this task, you need to set the email address and frequency in the Settings screen, and then review the email sent. These settings would also dictate how much is contained in the email - a weekly email would contain all food that will spoil in the coming week, while a daily email would simply contain information about the next day. This was previously our second task.

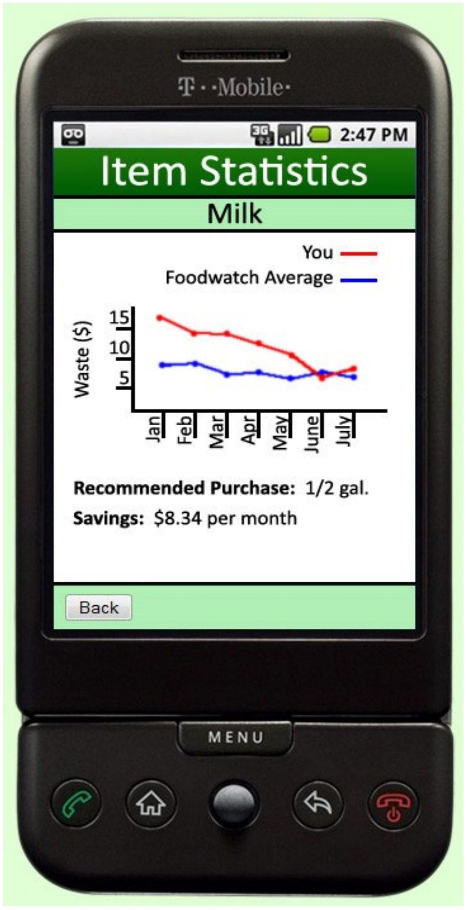
Task 2: Determining what quantity of milk to buy to minimize waste [Medium]
This, our new task, focuses around having the customer review their purchase and use history. You are at the store, about to purchase milk, and must decide how much to purchase. Foodwatch has two sections to address this: both a quick notification to you (as a popup dialog from your shopping list) and a full-page item stats screen that displays a 12 month purchase and use history compared to the average of all Foodwatch users.

Task 3: Finding a recipe for dinner with food you have at home [Hard]
As before, this task combines all three of Foodwatch's major screens to demonstrate how more information about your food can be used to make complex decisions. While at the store, you need to find something to make for dinner. Using Foodwatch, you can do this by looking up a recipe that contains items you already have at home. This will hopefully motivate you to use food you have previously purchased rather than buying more food by making these decisions easier and quicker.

Interface Revision

Obviously our biggest change from our low-fidelity prototype to the interactive prototype was the addition of our new second task, which required new screens (Figure 1) and new prototype functionality. We revised our shopping list layout to include a waste warning, and after user testing (Appendix A: Figures 2 and 3) our new task revised it even further to clean it up. After testing we decided to section the shopping list with headers to partition items the user has added, items that Foodwatch's AI has automatically suggested, and items whose purchase quantity should be reduced (our new warning feature) from each other. More good feedback came in the suggestion to give a weekly summary of wasted food, which we have decided to package into our email reminders.

Apart from the new sections of our interface, we have finally settled on a good representation of freshness (Figure 2). Blocks of color didn't work well for our initial design, and when we revised that to a system of freshness bars in our low-fidelity prototype, it was often confused for quantity of food remaining rather than freshness. Our new approach uses a colored stopwatch to symbolize time as well as freshness.



Recommended Purchase: 1/2 gal.

Savings: $8.34 per month

Figure 1: Item statistics

We reworked our approach with the settings screen, since the only necessary variables to set are email address and frequency (time of the email is mostly irrelevant), since our time dialog was confusing to some people.

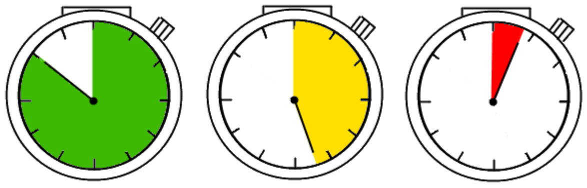
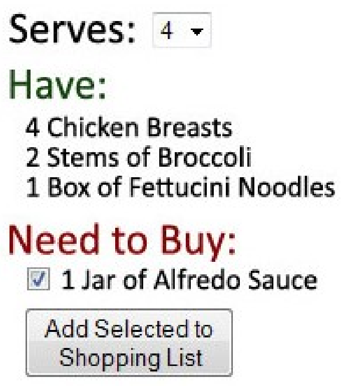


Figure 2: New freshness meter

In our low-fidelity prototype, we also had problems with users' interpretations of the recipe screen, so we clarified that by reorganizing it. The list of ingredients (Figure 3) is now displayed with a much clearer distinction between items you have at home and items you will need to buy at the store. We also added in the option to not push all of the required ingredients to the shopping list, as one of our user testing episodes suggested to us.
Some of our user testing feedback was addressed purely by the transition to a higher-fidelity prototype, which has more easily identifiable buttons and readable text.


Serves: 4

Have:

4 Chicken Breasts

2 Stems of Broccoli

1 Box of Fettucini Noodles

Need to Buy:

☑ 1 Jar of Alfredo Sauce

Figure 3: Recipe ingredients

Prototype Overview

Overview of Implementation
From the beginning, we designed Foodwatch with the g1/Android architecture in mind, so naturally our interactive prototype is represented within the g1 frame. This enacts the feel of really using it on a phone as opposed to a web interface. While the phone includes Back, Home, and Menu buttons, since our prototype's main interface is only one layer deep, we utilize the Menu button to bring up the standard Android menu and navigate between Foodwatch functions.

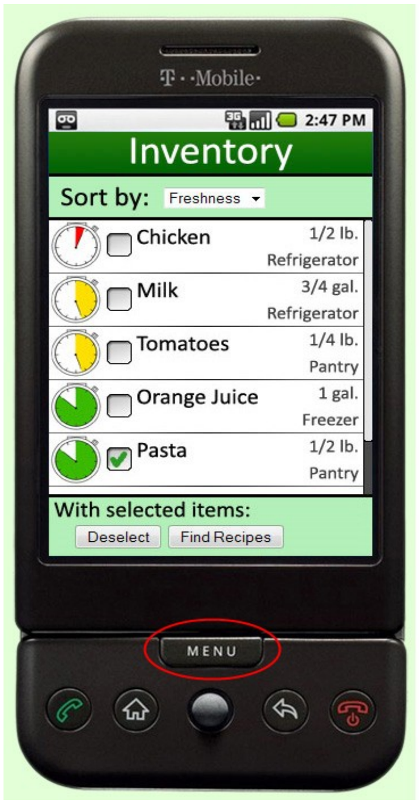
A couple of our screens do, however, have back buttons on the pane so that we did not have to rely on the g1's button. In addition, our prototype simulates using the g1 by starting the user on the Android desktop (Appendix A: Figure 1), showing that Foodwatch launches straight to the point with the inventory screen.

We used Javascript/JQuery and HTML to build the interface such that we could overlay images (for checkboxes and the menu screen) relatively easily while also being able to build the prototype quickly.

Scenarios

<u>Scenario 1: Set an appropriate Time/E-mail For the Status Report</u>

The G1's menu button can be pressed at to pull up a menu to jump to different pages (Figure 4 and 5).



Chicken	1/2 lb.	Refrigerator
Milk	3/4 gal.	Refrigerator
Tomatoes	1/4 lb.	Pantry
Orange Juice	1 gal.	Freezer
[✓ Pasta]	1/2 lb.	Pantry

Figure 4: Inventory

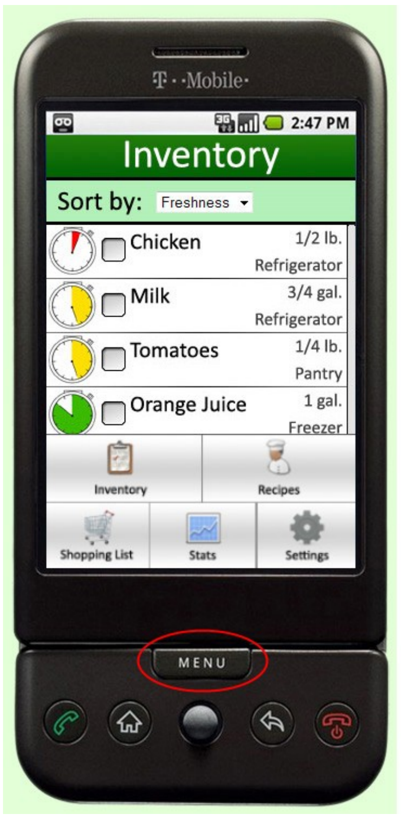
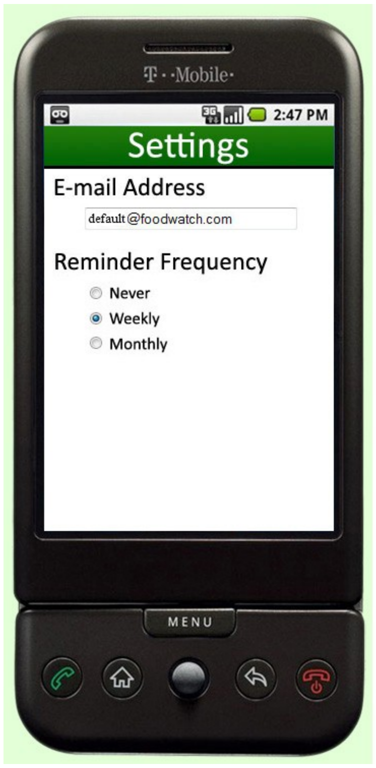


Figure 5: Inventory with menu

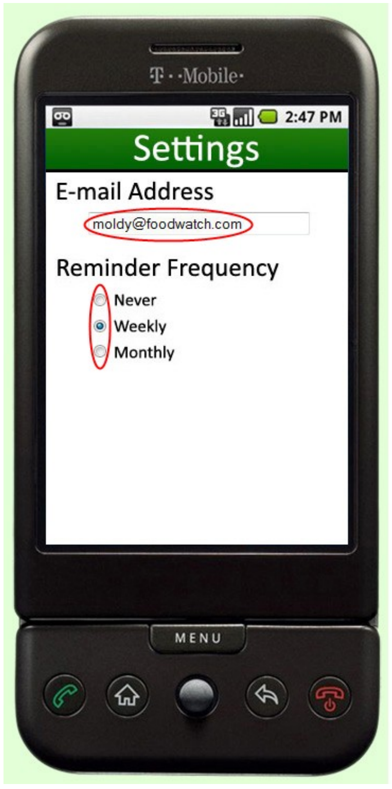
A user can then select Settings to jump to the Settings page. From the settings page, the user can change the default e-mail and how often the e-mail is sent (Figures 6 and 7).



E-mail Address: default@foodwatch.com

Reminder Frequency: Never [✓ Weekly] Monthly

Figure 6: Settings



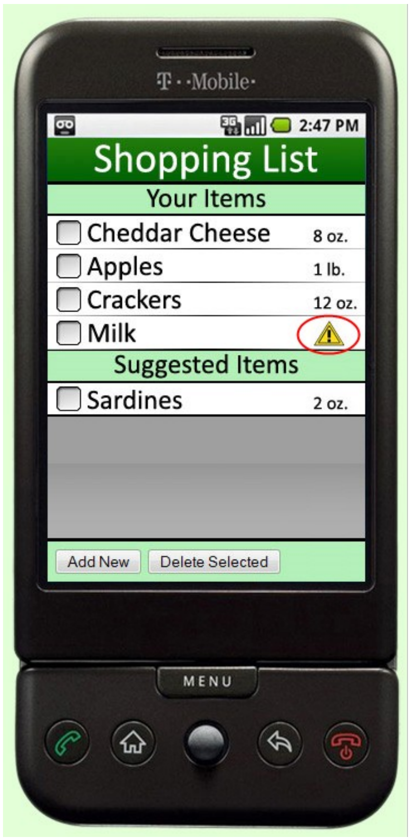
E-mail Address: moldy@foodwatch.com

Reminder Frequency: Never [✓ Weekly] Monthly

Figure 7: Highlighted settings

Scenario 2: Figuring Out Recommended Amounts of Food to Purchase

From the Shopping List, Foodwatch notifies you of often spoiled foods. Pressing on the warning symbol then pulls up a pop-up (Figures 8 and 9).



2:47 PM

Shopping List

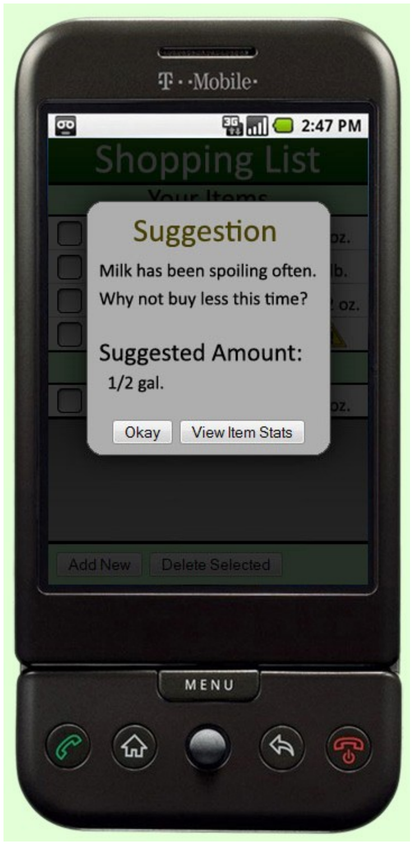
Your Items

Cheddar Cheese	8 oz.
Apples	1 lb.
Crackers	12 oz.
Milk	⚠

Suggested Items

Sardines	2 oz.

Figure 8: Shopping list



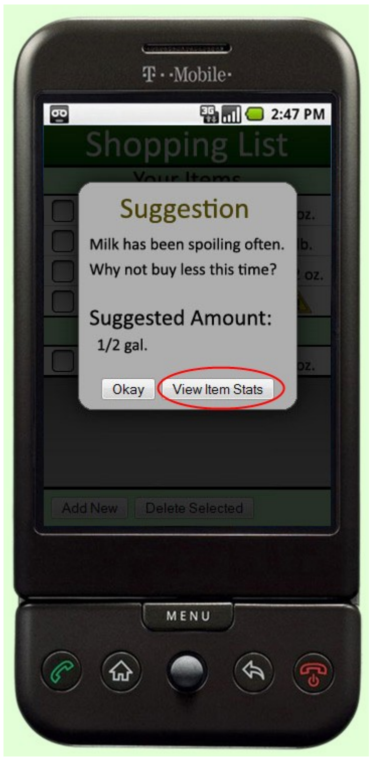
Suggestion

Milk has been spoiling often.
Why not buy less this time?

Suggested Amount:
1/2 gal.

Figure 9: Warning pop-up

The pop-up suggests a recommended amount of the product to buy based on your usage habits. Clicking on 'View Item Stats' takes you an individual stats page (Figures 10 and 11).

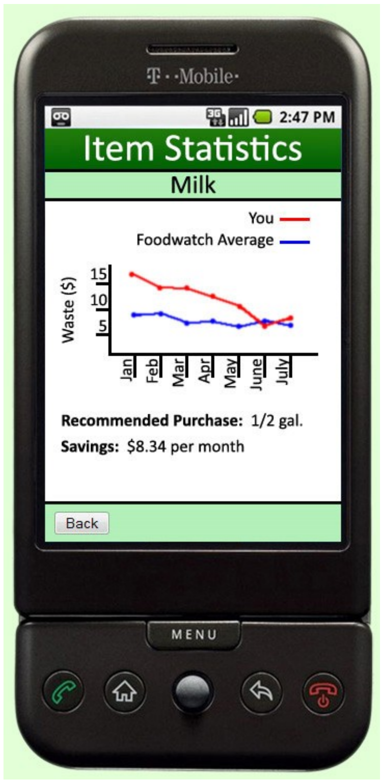


Suggestion

Milk has been spoiling often.
Why not buy less this time?

Suggested Amount:
1/2 gal.

Figure 10: Warning pop-up highlighted



Item Statistics

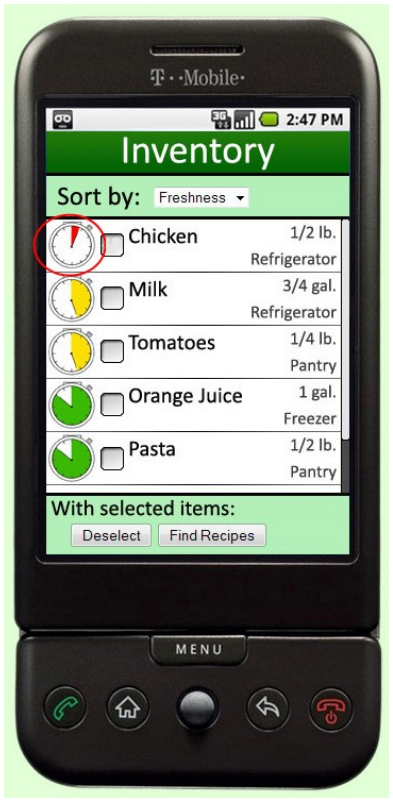
Milk

Recommended Purchase: 1/2 gal.
Savings: $8.34 per month

Figure 11: Item statistics

<u>Scenario 3: Generating new Shopping List Items Based off Your Inventory/Recipes</u>

A customer can check their inventory to see how fresh food is and select items to generate a recipe (Figure 12 and 13).



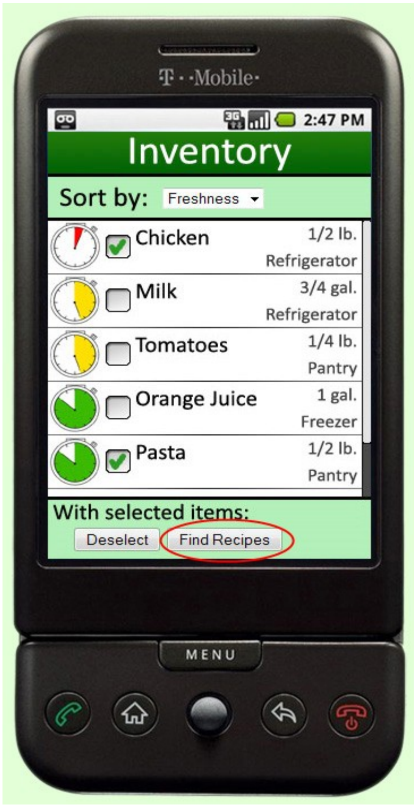
Sort by: Freshness

☐	Chicken	1/2 lb. Refrigerator
☐	Milk	3/4 gal. Refrigerator
☐	Tomatoes	1/4 lb. Pantry
☐	Orange Juice	1 gal. Freezer
☐	Pasta	1/2 lb. Pantry

With selected items:

Deselect | Find Recipes

Figure 12: Inventory



Sort by: Freshness

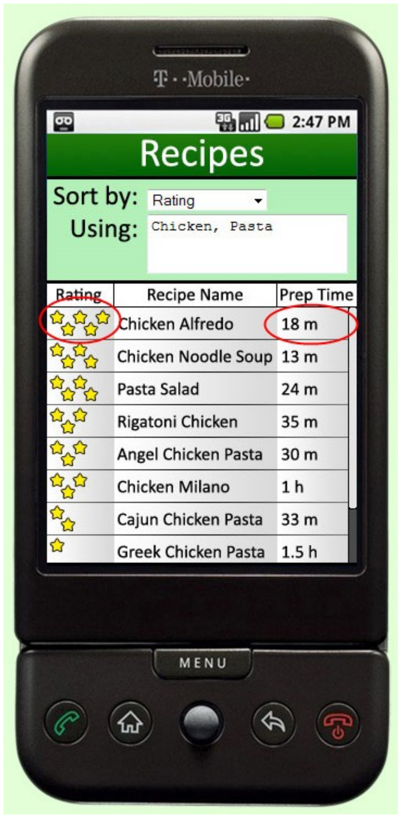
✓	Chicken	1/2 lb. Refrigerator
☐	Milk	3/4 gal. Refrigerator
☐	Tomatoes	1/4 lb. Pantry
☐	Orange Juice	1 gal. Freezer
✓	Pasta	1/2 lb. Pantry

With selected items:

Deselect | Find Recipes

Figure 13: Inventory items selected

The Recipes Screen provides such information as preparation time and the rating of the recipe. Pressing on a specific recipe will bring up a more detailed single recipe screen (Figure 14 and 15).

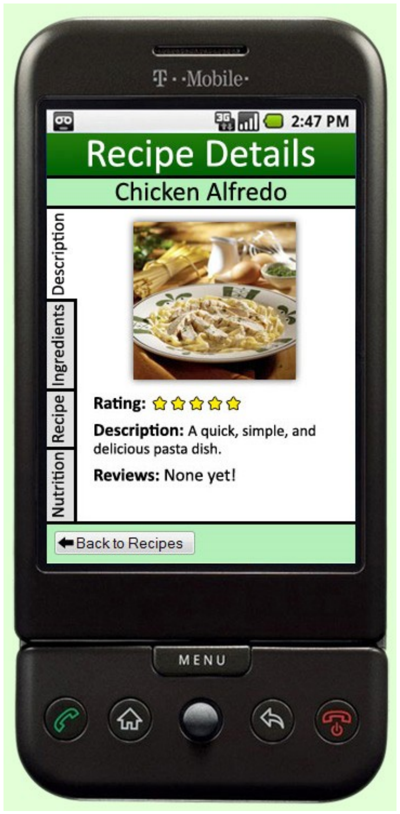


Sort by: Rating

Using: Chicken, Pasta

Rating	Recipe Name	Prep Time
★★★★★	Chicken Alfredo	18 m
★★★★	Chicken Noodle Soup	13 m
★★★★	Pasta Salad	24 m
★★★	Rigatoni Chicken	35 m
★★★	Angel Chicken Pasta	30 m
★★★	Chicken Milano	1 h
★★	Cajun Chicken Pasta	33 m
★	Greek Chicken Pasta	1.5 h

Figure 14: Recipes



Chicken Alfredo

Rating: ★★★★★

Description: A quick, simple, and delicious pasta dish.

Reviews: None yet!

Back to Recipes

Figure 15: Single recipe

From the single recipe screen you can find out additional information about the recipe, such as ingredients. From the ingredients screen you can see what additional items you need to make this recipe and add selected items to your shopping list (Figure 16, 17 and 18).

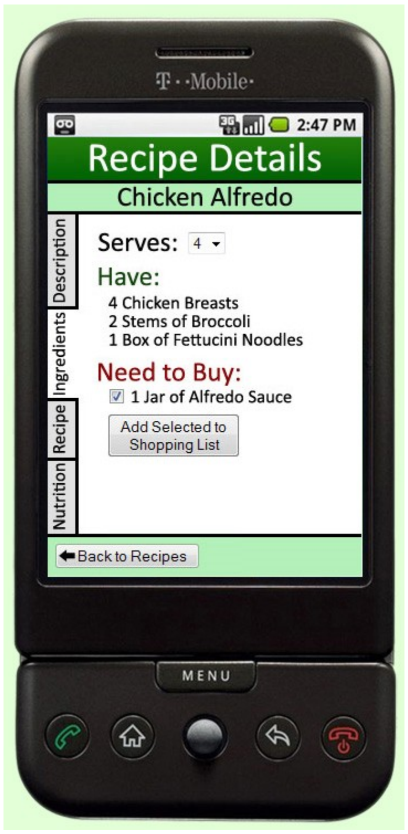


Figure 16: Ingredients

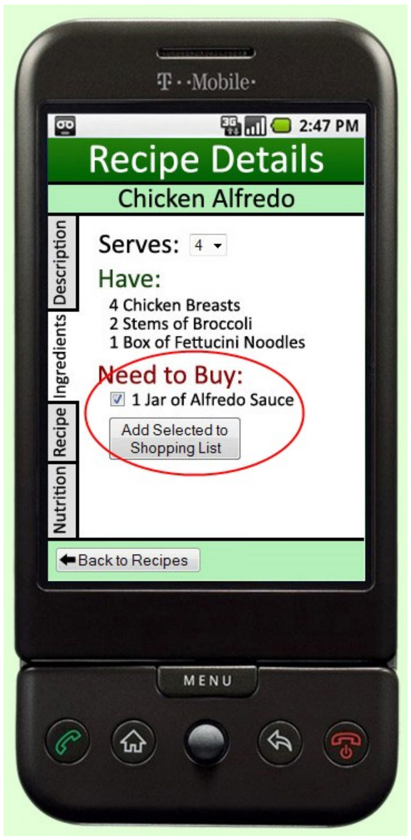


Figure 17: Adding ingredients

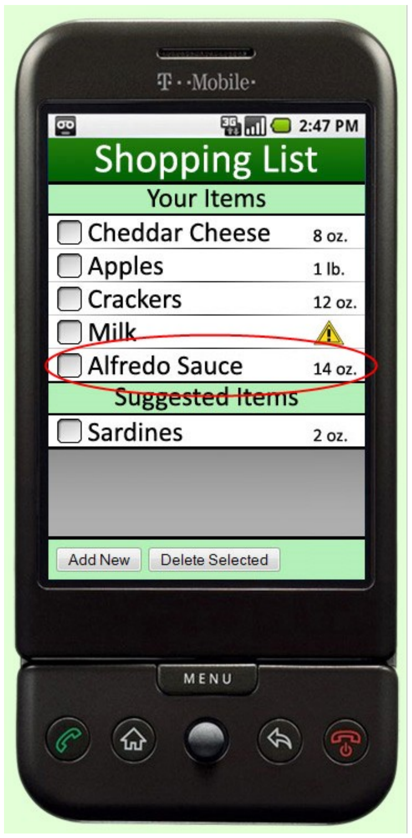


Figure 18: Shopping list

Tools

<u>Helpful:</u>

The primary tools used on the interactive prototype were Photoshop, the JQuery JavaScript library, and HTML/CSS. Photoshop helped in the basic graphic design of the screens. The ability to show/hide layers in Photoshop allowed us to break up the screens and generate configurations of smaller components. The grouping of layers also facilitated an ease of organizing and grouping by section and made it possible to work on the whole prototype within a single file. Each screen was sliced and exported to HTML. JQuery was used to detect clicks and do swapping of screens based on input.

<u>Not Helpful:</u>

On the other hand, the tools were obviously not designed for this specific situation. It felt a little like we were retrofitting them to our needs. A prototyping program for linking images based on where a user clicked would be better, but due to time constraints it was better to go with what we knew rather than research and learn completely new software. In the end, the results produced with our selected tools were definitely adequate and appropriate for this type of prototype.

Omitted Design Ideas
Since this is a 'golden path' prototype, cuts had to be made for the sake of the time constraints we had. Most relevant to our current design is the omission of the global statistics page that was part of our original concept (Figures 19 and 20). This is meant to give the user an aggregate of all their food waste data. Since we're already covering single item statistics, it's a natural and useful extension of that.

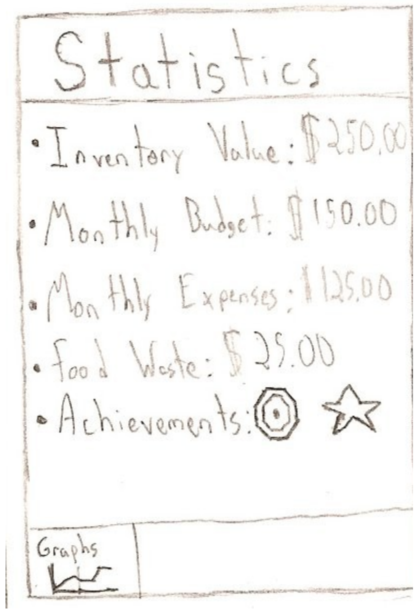

Statistics
- Inventory Value: $250.00
- Monthly Budget: $150.00
- Monthly Expenses: $125.00
- Food Waste: $25.00
- Achievements:

Figure 19: Global stats sketch

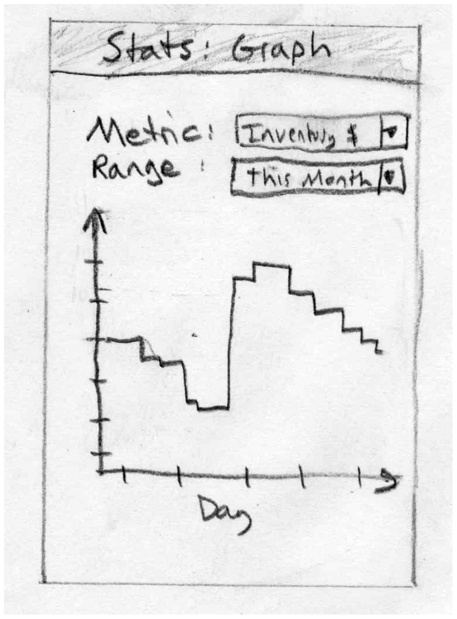

Stats: Graph
Metric: Inventory $
Range: This Month

Figure 20: Global stats graph

As stated previously, we ended up cutting one of our easier tasks in this step of the design process, after realizing we had two very similar tasks. Still very important to Foodwatch's mission are the notifications (Figure 21) when you store food improperly, however they don't form a detailed enough task with the interface and weren't nifty enough to warrant inclusion in our interactive prototype. A couple of the ideas we kicked around (but never really sketched out) were a disclaimer page, multiple custom shopping lists, and tagging of food items (so you could mark something as “Timmy's Snacks” if you wanted).

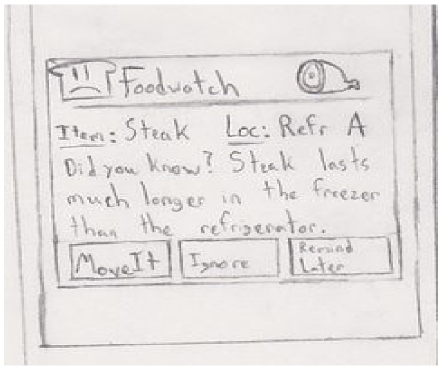

Foodwatch
Item: Steak Loc: Refr A
Did you know? Steak lasts much longer in the freezer than the refrigerator.

Figure 21: Notification

Wizard of Oz Trickery!
As stated from the beginning, Foodwatch depends on a 'magic refrigerator' that can detect what you put into it and how much/when. In our interactive prototype, we simply have a pre-generated inventory for demonstration purposes.

As well, the fridge would work in conjunction with the Foodwatch AI to determine when the customer ran out of something (especially something they regularly use, such as milk) and automatically add it to their shopping list under “Suggested Items”. Again, we don't yet have such an advanced refrigerator, so we have to use predetermined data.

Finally, on the same principle, we don't have a fully implemented search feature for recipes, or even for any combination of items in the inventory besides pasta and chicken. We have a premade list of recipes, and only one of those recipes will actually bring up a valid single recipe screen.

Appendix A: Prototype Screenshots

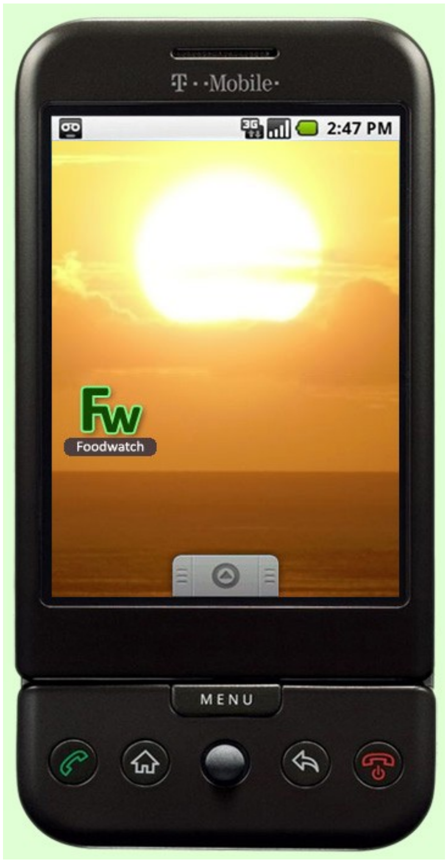
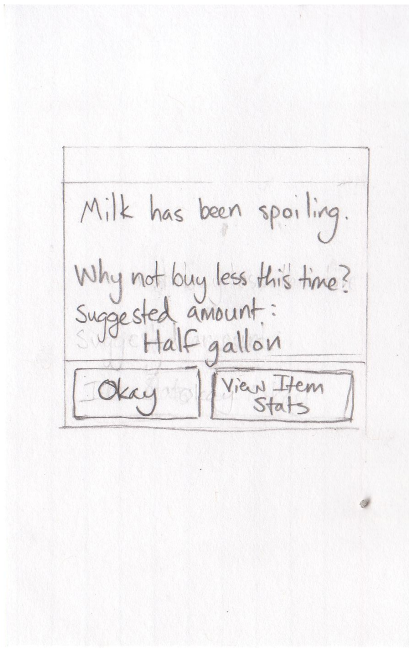


Figure 1: Opening screen

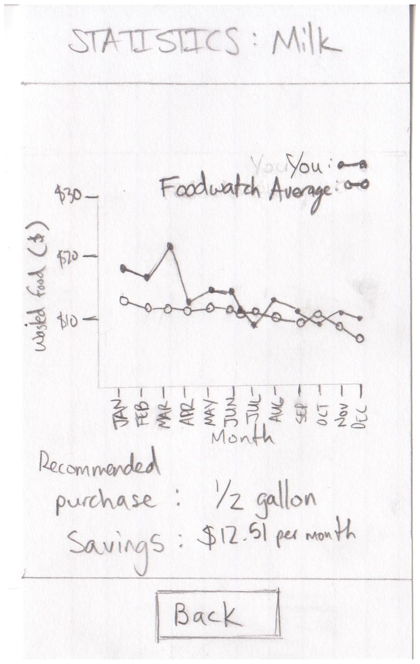


Milk has been spoiling.

Why not buy less this time?
Suggested amount:
Half gallon

Okay
View Item Stats

Figure 2: Tested Popup



STATISTICS : Milk

Recommended purchase : ½ gallon
Savings : $12.51 per month

Figure 3: Tested Stats Screen

Appendix B: Task 2 Testing Notes

Since we significantly changed our tasks around from the low-fidelity prototype to now, we went out and tested our ideas against a typical customer. Here are the results.

Lewis- Age 26

- completed the task with no problems

Suggestions:

- highlight the shopping list row containing the warning
- separate out the three shopping lists
- weekly summary of wasted food
- less than 3 taps to any part of the app
- swipe across an item on a list to see more info, such as stats
- color code rows of a list