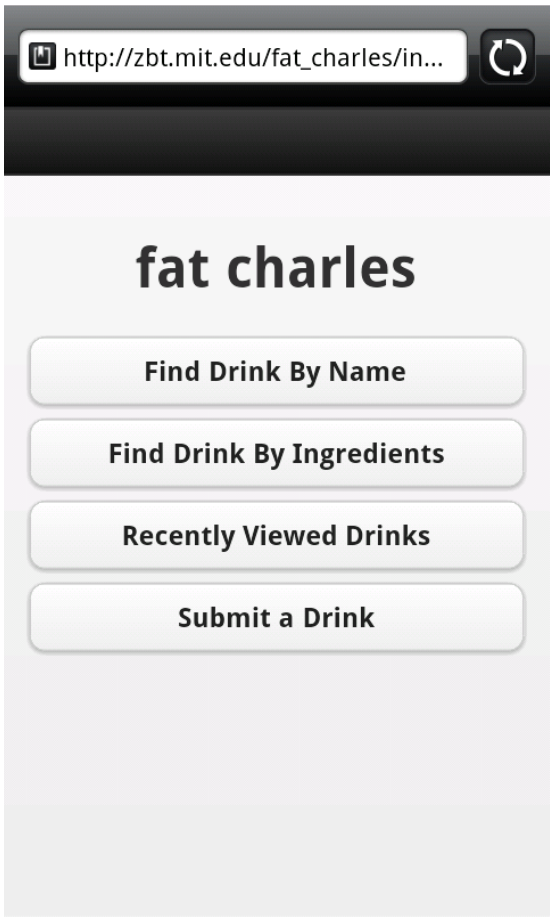
1. Design

Home page

When the user starts the program, he is presented with 4 options:

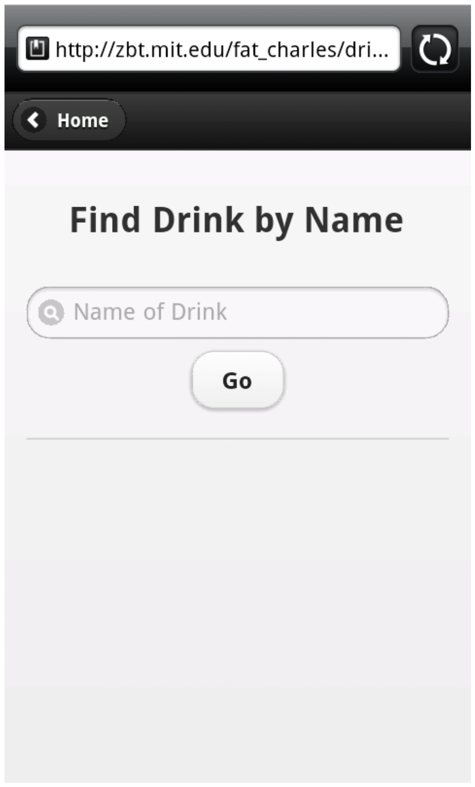
- Find Drink By Name
- Find Drink By Ingredients
- Recently Viewed Drinks
- Submit a Drink



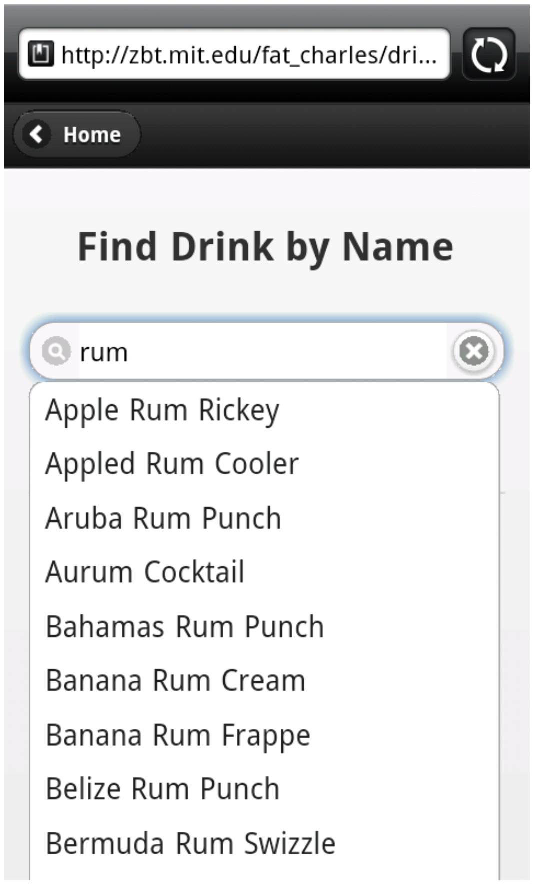
From the very beginning, our home page strived for simplicity. We wanted to clearly lay out what the user's options are for getting to a drink, and make it very self-explanatory. Originally, our design did not include the option of seeing recently viewed drinks. We implemented this feature during user testing when we saw users struggle to remember the names of drinks they had just seen. We ordered the four options based on their predicted "popularity".

Find drink by name

When searching by name the user simply needs to type in the name of the drink. The navigation is at the top left of the screen and is externally consistent.



We implemented autocomplete to minimize the amount users will have to type as we realize typing on a phone can be tedious. Autocomplete can also help the user remember the name of a drink if they only remember part of it, allowing the user to make use of recognition in addition to recall.


http://zbt.mit.edu/fat_charles/dri...

Home

Find Drink by Name

rum

Apple Rum Rickey
Appled Rum Cooler
Aruba Rum Punch
Aurum Cocktail
Bahamas Rum Punch
Banana Rum Cream
Banana Rum Frappe
Belize Rum Punch
Bermuda Rum Swizzle

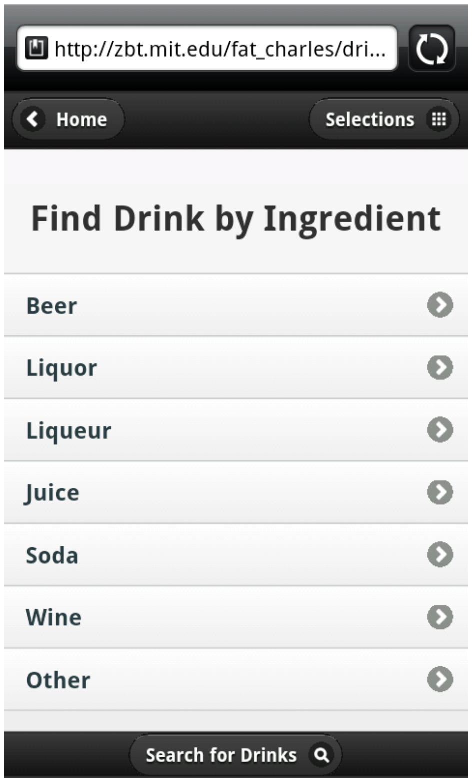
There wasn't much iteration over the design of this page. We knew from the beginning that we wanted it to be simple, that we wanted to implement autocomplete.

Find drink by ingredients

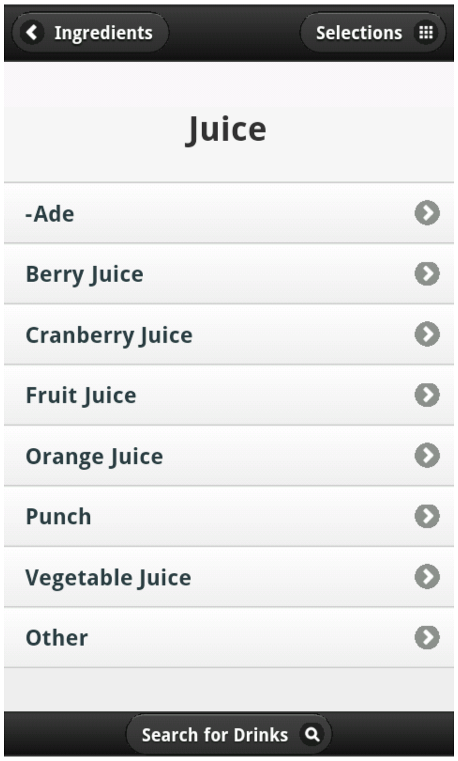
This feature allows users to find drinks based on the ingredients they have. The ingredients all categorized. To make things more consistent and easier to implement, we decided to make ingredient fall into a category and sub-category, so there are three levels of this feature:

- The 7 main categories: Beer, Liquor, Liqueur, Juice, Soda, Wine, and Other.
- The sub-categories of each of the main categories.
- The ingredients of a specific category and sub-category, which allows selection.

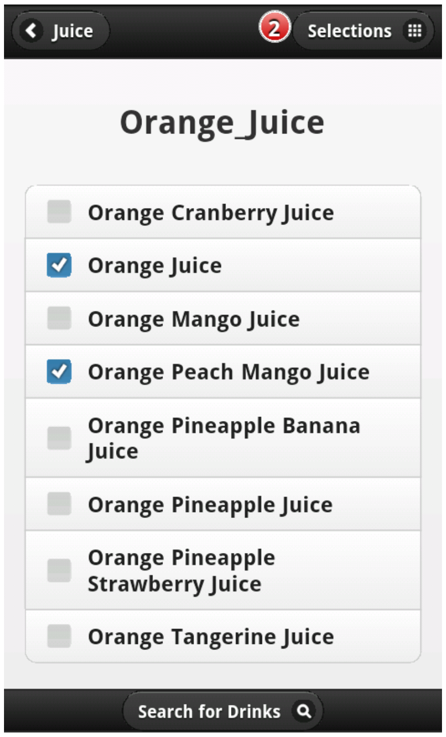
This page shows the top level of the 7 main categories:



This is an example of second level page, where the user has selected Juice:



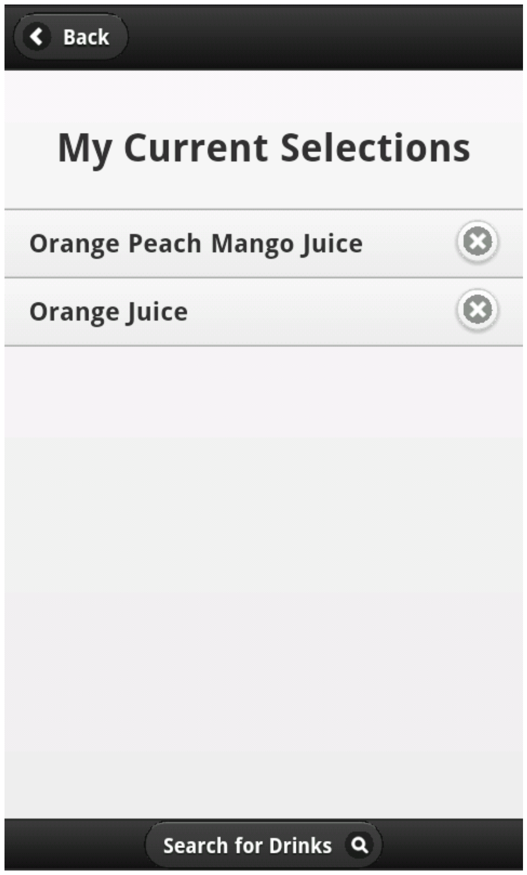
This is the lowest level of this feature and it shows all drinks that fall under the Juice->Orange Juice categorization.



Orange_Juice

- [ ] Orange Cranberry Juice
- [x] Orange Juice
- [ ] Orange Mango Juice
- [x] Orange Peach Mango Juice
- [ ] Orange Pineapple Banana Juice
- [ ] Orange Pineapple Juice
- [ ] Orange Pineapple Strawberry Juice
- [ ] Orange Tangerine Juice

There is also a Selections button that is persistently in the top right corner that allows the user to see which ingredients he has selected. This was an important design decision because we wanted to allow the user to see his current state but could not have it always present, like we would be able to do with a website. There is simply not enough room on a mobile screen to allow for that information to be continuously visible. So we decided having the selections easily available was the best solution. We also decided to put a number next to it to show the user that their selection is being added to the selections bin.

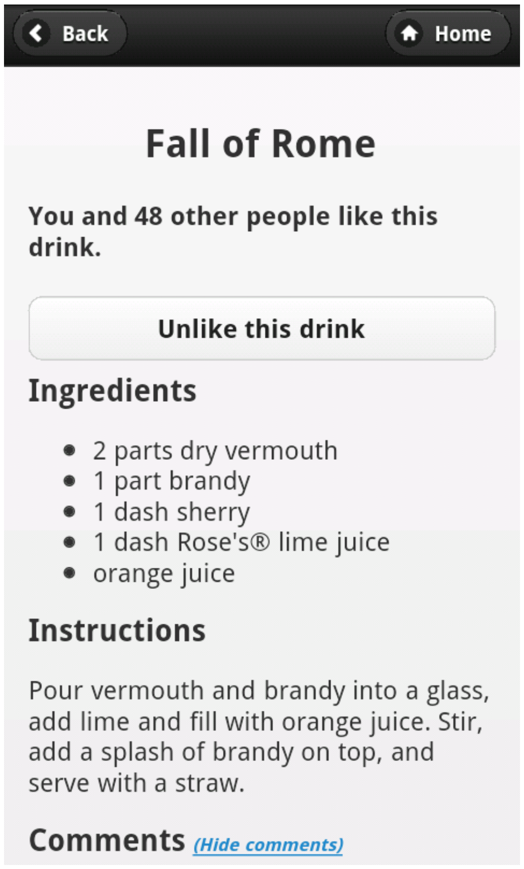


This feature was the most interesting and difficult usability challenge of this project. We discussed several very different designs for this page and even after all our user testing, we ended up changing our design in the middle of implementation. After paper prototype testing we decided to use icons to display each selection because we thought it would be more space efficient and that the images would be a nice touch. We realized during implementation that the text for each item does not always fit inside of an icon, and so we decided to go with a simple list design.

Search results and the drink page

After searching by name or by ingredients, the user will be shown the results of his search. Generally, the user is shown higher rated drinks first. For search by ingredient, the search is first ordered in order of number of matched ingredients. We attached a number to the search result, and later also added the word “likes” to clarify the meaning.

Every drink has a drink page associated with it that displays the name, ingredients, instructions, likes, and comments for that drink.


Back
Home

Fall of Rome

You and 48 other people like this drink.

Unlike this drink

Ingredients

- 2 parts dry vermouth
- 1 part brandy
- 1 dash sherry
- 1 dash Rose's® lime juice
- orange juice

Instructions

Pour vermouth and brandy into a glass, add lime and fill with orange juice. Stir, add a splash of brandy on top, and serve with a straw.

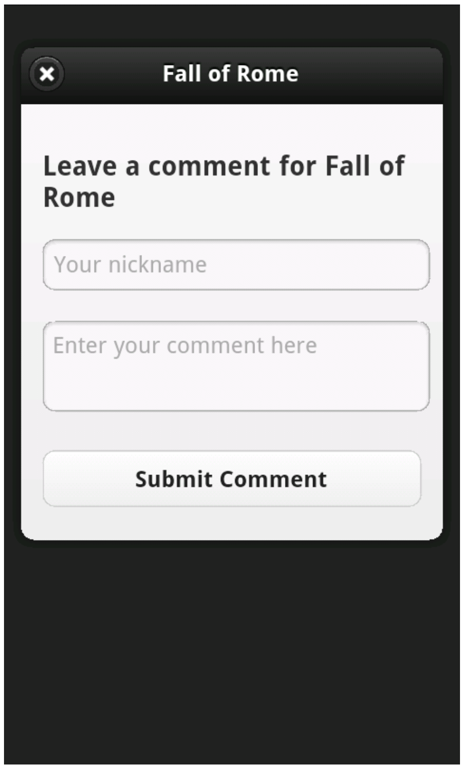
Comments (Hide comments)

The drink page contains the name of the drink, the number of likes, the ingredients and instructions, and finally comments. While the overall structure of the page has remained the same, different components have changed in different ways.

The original way that ratings were done used “upvotes” and “downvotes”, similar to Reddit’s system. There were issues about how to implement “clearing” a vote after it has been upvoted or downvoted, and users generally didn’t seem to need the downvote capability. The second time around (after the computer prototype), we switched to “likes”, which communicates the idea of a ranking of drinks with a lot more simplicity.

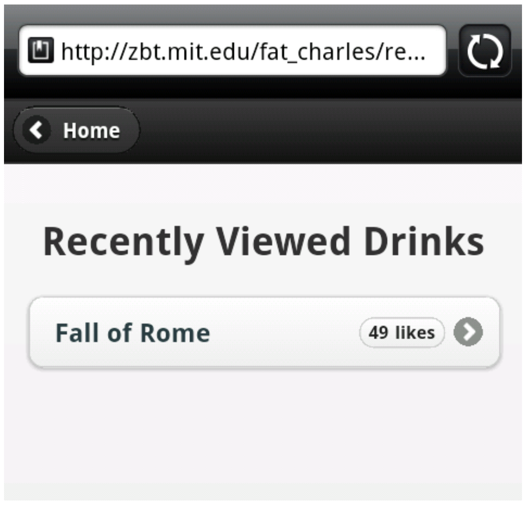
The original design used in the computer implementation hid comments until a button was clicked to show them. User feedback indicated that users wanted just to see the comments without an extra tap. In the final iteration, comments are shown by default. A small link next to the “Comments” header shows and hides comments.

Finally, comments are implemented using a popup so that the user feels that they are still “on the same page”, but the majority of the entire screen can be used for the comment form. In paper and computer prototyping, the comment page was completely separate, which required text fields identifying the drink. That was made unnecessary with the popup.

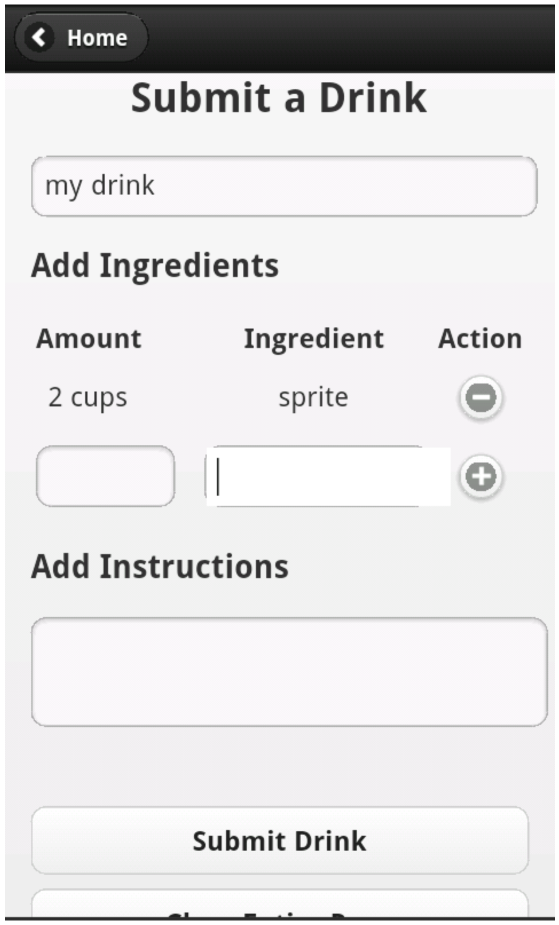


Recently viewed drinks

The recently viewed drinks page was created in response to user input during both paper and computer prototyping that they wanted to easily look up drinks that they had seen already, even if they might have forgotten the name. It is a simple page consisting of a single list of links to drinks.



Submit a drink



Submit a Drink

my drink

Add Ingredients

Amount	Ingredient	Action
2 cups	sprite	⊖
		⊕

Add Instructions

The “submit a drink” page includes fields for the name, ingredients, and instructions for creating a drink. While the basic idea of this page stayed the same throughout the project (find ways for the user to input everything and display them as they are inserted), details in the implementation changed between prototypes and until the final implementation.

The ingredients text field started off as a single text field. However, due to back-end constraints where the amount and ingredient name are stored separately, the text fields were split as well. This meant that additional screen width was needed. A button that contained the word “Add” to add an ingredient was scrapped for a simple plus button, and the “clear text box” button became a simple “X” that was moved into the text field.

2. Implementation

Front-end implementation

The front-end of our application is done in HTML/CSS/Javascript with jQuery added to make the coding process easier.

We used the jQuery Mobile framework, which packages CSS and Javascript that makes the page elements look like they are part of a native app. As with all frameworks, there are pros and cons to this decision. While jQuery Mobile has been hailed as one of the best “rapid prototyping” techniques for mobile, we found that the external consistency with native applications that users are already familiar with (iPhone/iPod touch in particular) and its ease of use was worth the additional cost of learning the framework and working around some of its rough edges. It allowed us to worry less about the details of the code and more time to think about the user experience.

The jQuery Mobile framework implements page transition effects by using asynchronous AJAX page loading. Unfortunately, because our pages often contained multiple sub-pages and pop-ups, we could not take advantage of this in all of our pages. Thus, we disabled AJAX page loading to and from all of the main pages to maintain consistency.

We have one HTML or PHP page for each “main” feature in the application. The different pages correspond to the home page, search by name, search by ingredients, recent drinks, submit a drink, and drink viewing functions. The search by ingredients page contains multiple sub-pages, each of which represents the categories or ingredients that show up on the screen at any given time. The drink viewing page also contains a sub-page which is the popup dialog for commenting.

The styling of the front-end that is not taken care of by jQuery Mobile is contained within a single CSS file which is loaded with every page contains selectors for the id of each different page. Javascript code provides the functionality of all buttons and links, including making the appropriate AJAX calls to load additional data into the page.

Back-end implementation

The back-end of our application is based on a standard setup of the MySQL database. Server-side scripts written in PHP serve as the bridge between the client-side code and the database.

Our database contains five tables:

1. ingredients - contains one entry per ingredient of a drink. It is indexed by the drink’s ID.
2. instructions - contains one entry per drink. The single table entry contains all the text of the instructions.
3. ratings - contains the number of likes per drink.

4. comments - contains one entry per comment; multiple entries exist per drink if the drink has more than one comment.
5. categories - this is our most complex table. Many of our recipes were obtained from crawling existing sites on the Internet. This table takes the specific ingredients listed in those recipes and maps them to the general ingredient categories (the ones used in “find drink by ingredient”) so that they may be looked up in our application.

Most pages consist of static HTML and Javascript, and AJAX calls are used to load and push data from and to the server, such as comments and likes. Two exceptions to this are the “find drink by ingredient” page and the drink page, in which server-side logic is used to dynamically generate the page’s contents.

In addition to server-side storage in the database, client-side session storage is used for things that need to be stored temporarily. Session storage is used to store the ingredients that are currently selected by the user while searching by ingredient as well as the drink IDs for recently viewed drinks.

Session storage is also used to remember if a user liked a drink when determining whether the user should be allowed to like it again and deciding whether the phase “you and … other people like this drink” should be used. This is not a complete solution, but for simplicity and so that a login is not necessary for such a simple application, this is a fine simplification. In reality, a login would be necessary to keep track of users and their likes on the server side.

<u>3. Evaluation</u>

Users

For our user tests, we simply asked a three of our friends, who each have differing levels of experience with drink mixing, to test our application. User 1 has never tried alcohol, user 2 only drinks during parties, and the user 3 has had lots of experience with mixing drinks. The users vary across the spectrum in terms of their familiarity with mixed drinks, both in making them and drinking them. They are, however, all college students who are familiar with an environment in which the application may be used.

We began each user test by informing the user that the point of the app is to help people find mixed drinks that they enjoy. Then we jumped straight into describing the tasks, only revealing the next task after the current one was completed.

Tasks

1. You have several different drinks that you would like to mix together, but you don't know what to make. Using the application, find a drink that contains the following ingredients: Cranberry Juice, Vodka, Lime, and Pepsi.
2. Re-find the drink that you found in task 1 any way you can. Once found, like it and leave a comment.
3. You realize that switching out Pepsi for Dr. Pepper makes the drink much better. Submit the new mixed drink you have just created.

Usability problems

- **Information Scent**
    - <u>Description:</u> The user could not find Pepsi, which was under the Soda->Cola categorization. Another user thought it was weird that clicking on Vodka took him to more options, and then that to just select Vodka he had to go to Vodka and then select Vodka.
    - <u>Severity:</u> Major
    - <u>Solutions:</u>
        - Re-categorization: Find some other way of categorizing the ingredients in such a way that the users know where everything is based solely on the categorization.
        - Selection of categories: Allow users to select category headings such as Vodka instead of forcing them to go deeper into the hierarchy of ingredients to select Vodka.
        - More information scent: Show a few of the subcategories next to the category headers, so even if the user doesn't realize right away that Pepsi is in Cola, they may see Pepsi in the list of two or three things inside the category.

- **Feedback**
    - Description: One user didn't realize they had selected one wrong ingredient and did not check the selections page to see if they selected everything correctly.
    - Severity: Minor
    - Solution: On the drink page, we could have some sort of feedback that highlights which ingredients that were searched for are present in the drink.
- **Safety**
    - Description: One user unintentionally deleted an item from his selections page and could easily retrieve his selection. He was forced to go back through the categories to reselect it.
    - Severity: Minor
    - Solution: On the selections page, we could have checkboxes instead of just "delete" buttons. That way, users could easily get back ingredients that they did not mean to delete.
- **Visibility**
    - Description: When looking for the drink a second time, one user searched by reselecting all the ingredients, instead of simply pressing the "most recently viewed" button.
    - Severity: Cosmetic
    - Solution: We could not think of a way to make it more apparent that "most recently viewed" was an option.
- **Efficiency**
    - Description: The users found the submission of a new drink to be tedious and annoying.
    - Severity: Major
    - Solution: Since it is likely common that most drinks will be only a slight alteration to an already existing drink, we can add another button on the drink pages to fix this issue. The button will take them to the submission page with the information of the drink they were viewing already entered into all the fields, so all the user will have to do is make a few changes and they'll have a new drink.
- **Feedback**
    - Description: The users didn't realize they had to press the + button to add the ingredient. In their mind, the + just added another field for ingredients.
    - Severity: Major
    - Solution: We can simply change the interface to match the mental model of the users.

4. Reflection

Iterative design

The biggest lesson from the iterative design which we took away was to make several easy to evaluate designs early on, and then decide which one to proceed with after evaluations and user testing.

We really had three primary designs to go with, but some extra brainstorming might have led to a different or more elegant design. Our design is a very traditional level design, and as aptly pointed out by Ted during our GR5 meeting, “good UI design is often boring” because traditional agrees with the user and the outside experiences that users would be accustomed to. However, if our group were to redo one thing, brainstorming would most likely be the thing to improve upon. While we believe our app is efficient, having more options would allow better comparison and a wider array of options.

The “find drink by name” feature of our app was mostly fleshed out by the beginning, because there is no other way to input the name of the drink other than by typing. There was essentially no iteration involved in this feature, and the only change to this was the addition of autocomplete once our database was populated.

The “find drink by ingredients” feature was the one which received the most iterations and contemplation about how to flesh out. Firstly, we had the list view and the icon view conflict, where we had to decide between the display of ingredients as text or as icons. While the list view clearly allowed alphabetical ordering and easy display of checkboxes to tell you when you have put an item into your selections, there was must wasted space if the text did not occupy the entire width of the screen. The icon view allowed display of more ingredients before the need to scroll, but it would be harder to display the name of each ingredient or category. Additionally, displaying the “has been selected” status of an ingredient is less intuitive than the checkboxes in a list. Lastly, there is the need to reconcile the difference between iPhone style scrolling of pages (horizontal) and Android style scrolling of pages (vertical). Then we had to reconcile how to display the vast array of ingredients which were in our database. A hierarchy style of display would require the user to navigate through more pages before reaching final ingredients, and could require users to navigate through more pages when selecting ingredients from different categories. The alternative was to display all the ingredients in a “contacts” style manner where all of them would be listed alphabetically and they can be quick-scrolled by first letter. The final decision of how to maintain your selections and display them to you was also something which we weren’t sure of how to tackle.

The final feature, the “submit a drink” feature was thought to be very easy, because you simply add in ingredients, a name, and some instructions. This turned out later to be not so easy because of the way that our database of drinks was structured. Putting this with the mobile constraint actually made this feature quite a difficult one to tackle, because you didn’t have all the space of a desktop browser to type.

Overall, our fleshing out of these features and our decision on which to go with came down to the user testing which we performed. This was the place in the project where we think we excelled the most. We took feedback from users after the paper prototyping and computer prototyping phases and heavily weighed them when making any design decisions. They provided the most useful feedback as to what things in our prototypes worked and which didn't. Because we feel we did a thorough job of going through user feedback, identifying problems, and trying to pinpoint their sources, this made making design decisions easier.

We used user feedback to determine the risk of all our features. Though we didn't think we had many risky features, users provided valuable feedback for those few features we were unsure about. In particular, maintaining your selections and keeping state between different pages was important, and users showed that it was indeed not effective to simply have checkboxes. It turns out that we couldn't quite figure out an animation that we wanted, but that the flashing red bubble really helped to tell users what they needed to know. Their feedback on the usage of icons versus the usage of lists to represent the different categories also really helped to tell us that icons were not really helpful except for showing more icons per page, but they didn't provide users with the labels as well.

Iterative design also helped because we found out what things were infeasible in our app. Our decision to use jquery mobile at the beginning helped expedite things along and really boosted the rate at which we could initially prototype. This was exactly what we wanted, even though we knew that our app had bugs in its interactions with jquery mobile. As we dug deeper and went through more iterations, we fixed those bugs that weren't central to the main functionality of our app. However, during the first iterations of our app, we quickly found the limitations that jquery mobile provided. There were some errors in ajaxing and keeping state, and we quickly found the limitations of a mobile browser and the processing speed which it provides. Discovering this early helped to show us that loading our entire database of ingredients onto a page would either take forever, or reduce a mobile browser to its knees in responsiveness. Also, because we started with a strong prototyping tool and a target focus on mobile browsers, testing early and at different stages helped us to determine which features worked and which ones needed polishing at any stage of the design.

Our decision to prototype everything early on really helped us make the best decisions. We used paper prototypes for almost all the designs we had. Doing this really helped us not waste too much effort in expensive prototyping techniques like computer prototyping. Creating the icons through an image program would really have been time intensive, and our decision to draw and get user feedback at the beginning really helped narrow our focus while optimizing the time spent.