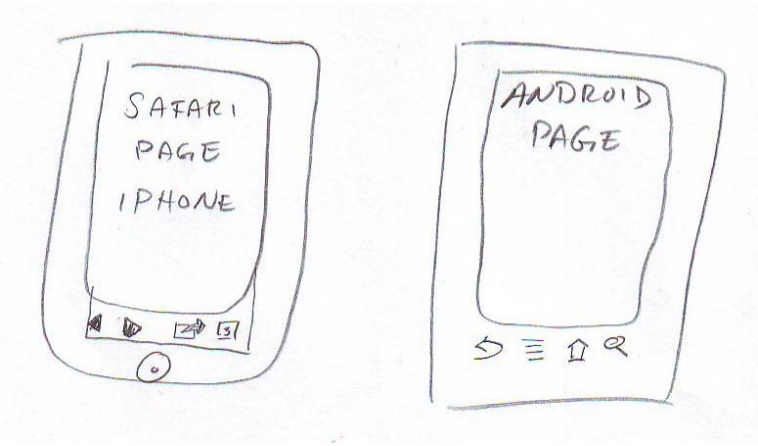
Fat Charles—Design

The first and foremost challenge is to make this feel like a good interface on a mobile phone, which means potentially dealing with limited functionality: having to adapt to a different environment than the default keyboard and mouse. Additionally, the lack of screen real estate is a major limiting factor, meaning we have to be smart and clean with whatever it is that we're using.

There are two methods with which to deal with Fat Charles on a mobile device. We can incorporate an app directly packed for native distribution and install for each device, or simply do mobile web. For now, the assumption is to go with a mobile web interface.

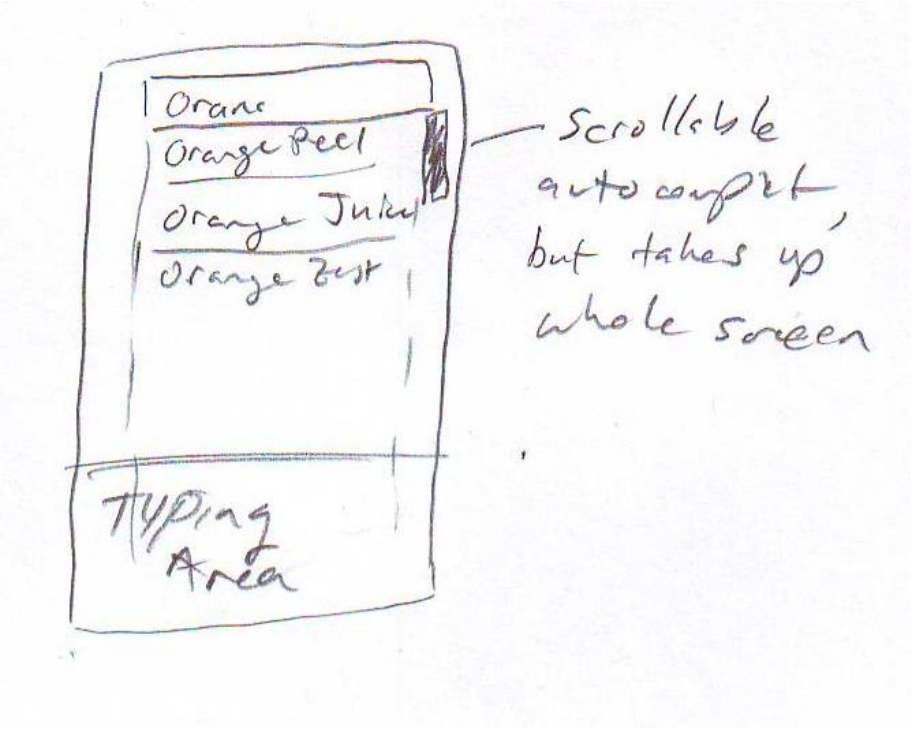


Designing for mobile web means we have certain built-in functionality, most important of which is a back button, which will allow the user to navigate back and forth. There is some additional functionality that either Android or Safari has, but we'll ignore the rest of that for now. Additionally, we note that both browsers are now Webkit based, so most JavaScript frameworks for mobile-web will work on them.

Home Page

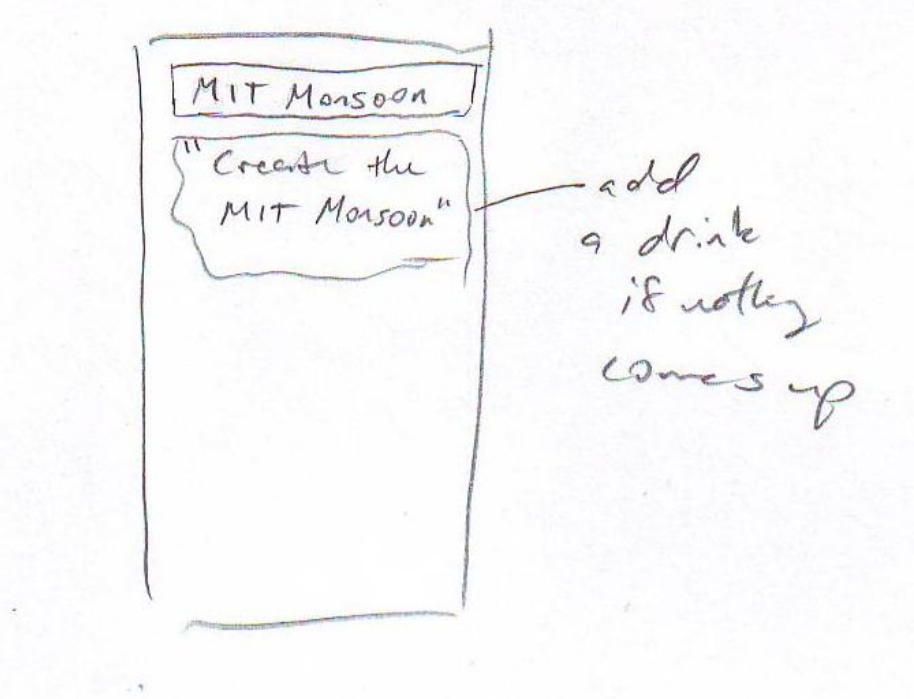
When coming to Fat Charles, we will seek to satisfy the primary goals of the user in as little screen real-estate as possible. Here are two possible home pages, with only a minor difference. They both involve the Fat Charles Logo at the top, with a text field that has helper text. Optionally, this can become just an external label. This says “Input ingredient or drink name”, and is a multi-purpose text-field because it will involve parsing for both drinks and ingredients. On the left, there’s a suggestion possibility for Fat Charles to give you something completely at random.

Typing



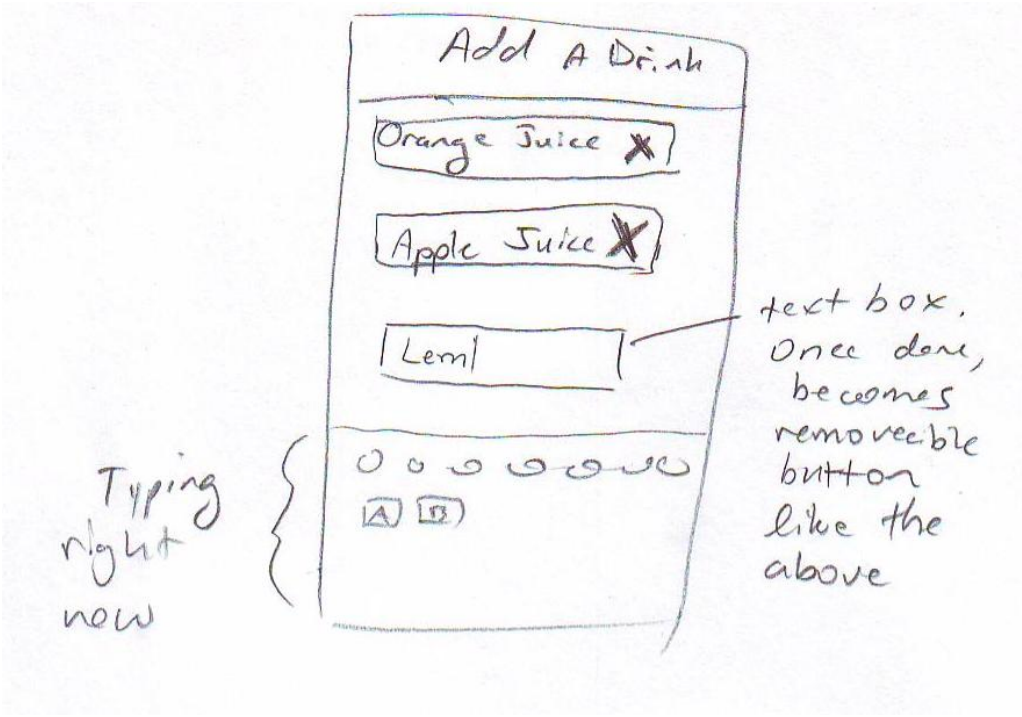
When the user types, some parsing happens, and the user is provided with a scroll-list of potential regex matches for what they have typed so far (3 or 4 characters will trigger this). As we can see, the logo disappears and more screen-space is utilized for the display of the choices. This scroll-able list has both drink names and ingredients that the user can select from, with the most popular at the top (indexed by our site statistics). Clicking away from the text-field will, by default in the browser, minimize the typing area and go directly towards the touch and select interface in the phone.

Task: Adding a Drink



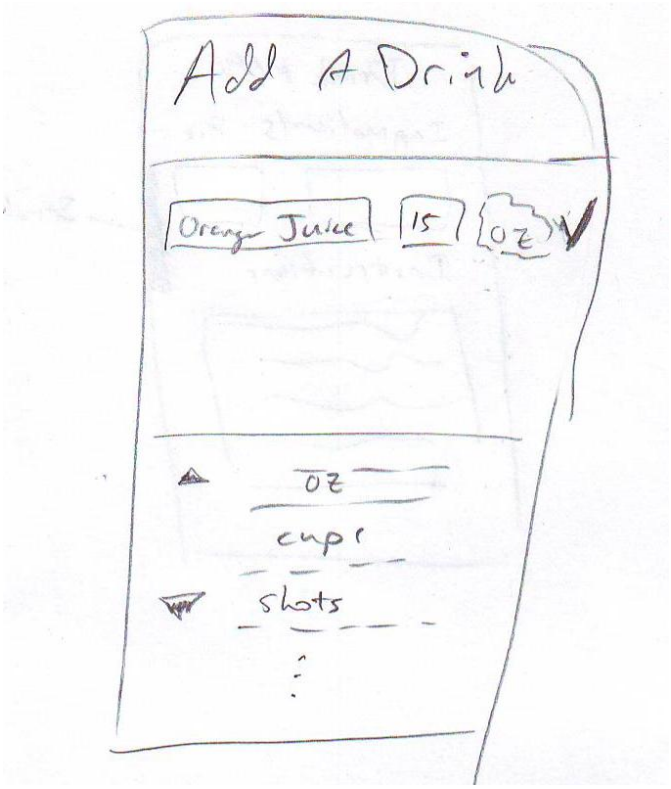
If the user continues typing, then the list continues to narrow based on the typed entries. The user does not have the option to create another page called “Rum & Coke” if it already exists. So if the user types in an already-existing name, they do not have the option to do anything other than click that drink. However, if the user continues to type to the point that they do not encounter any regex match whatsoever, there will be a Javascript generated button that allows them to “Create <DRINK NAME TYPED>”. This is how a user will go about adding a drink, simply by typing until they no longer have a regex match, because one way or another, they have to type in the entire name to add it to the database.

Task: Adding a Drink (cont'd)



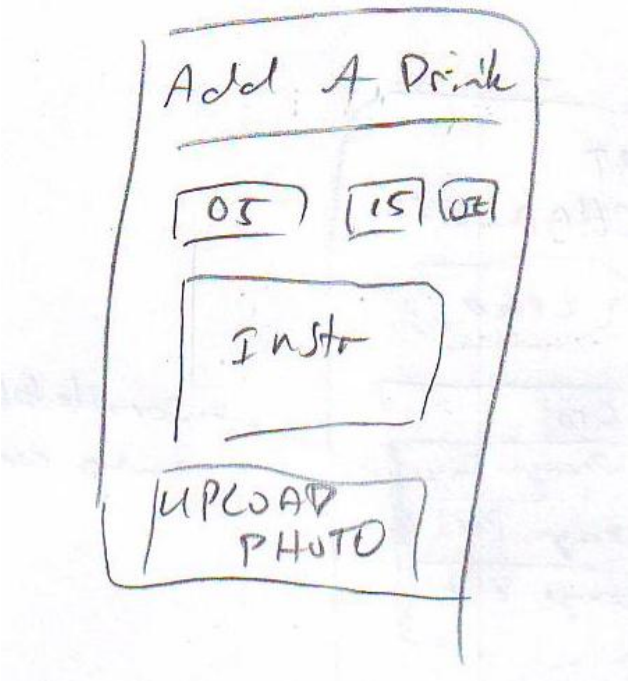
For the moment, I have put the header text of "Add a Drink", but the user will actually have the name of whatever they put previously in the top. This header will remain there for the duration of this "Add a Drink" mode. Additionally, it's a text-field, so the user can tell that they can touch and edit it if they deem they want another name. There are text-fields for the user to type in ingredients. In the same way that the home screen changed to give more space for choices, the same happens for typing in ingredients. Once an ingredient is added (user hits enter into the text field), it transforms into a button with an "X" on the right side so that the user can remove ingredients if they decide they no longer want them. There are more options for each drink in the next image.

Task: Adding a Drink (cont'd)



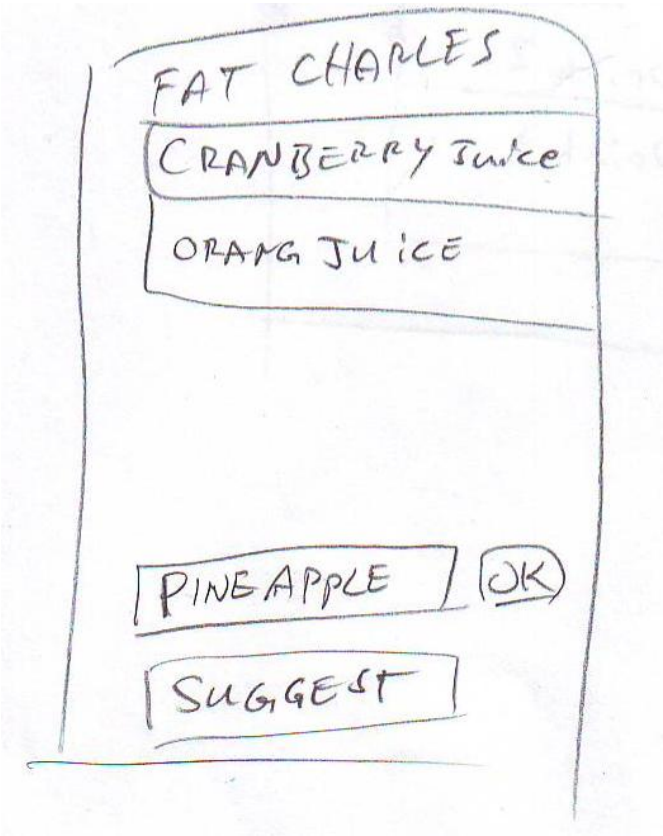
Additionally, there are fields to the right (not displayed in the previous image) where the user can select quantities. For something like the measurement, clicking it will give an iPhone-like scrollable list of limited options, "oz", "cups", "shots", etc. The user is in charge of typing in an amount such as 15 in front of the oz. Hitting either the "enter" key or clicking the check on the right at any point will automatically add the ingredient to your current list. There is no guarantee that the user must input an amount, so if they simply hit "enter" after typing "Orange Juice", then "Orange Juice" will be added to the list of ingredients without an amount. It is assumed that they will say something generic in the instructions about this ingredient, or it is allowed to vary according to each person if an amount isn't entered. After entering an ingredient, the user can select the "oz" or the "15"/blank amount, if they wish to change or add in an amount.

Task: Adding a Drink (cont'd)



The instructions field is just another textbox that's labeled where the user can type in the instructions. Then, at the end, the users also have the option of going into their phone to upload a photo of the current drink that they're adding if they choose. Because we assume that uploading a photo is the last step, they can hit a submit button below the upload photo button and submit their drink.

Task: Finding a Drink

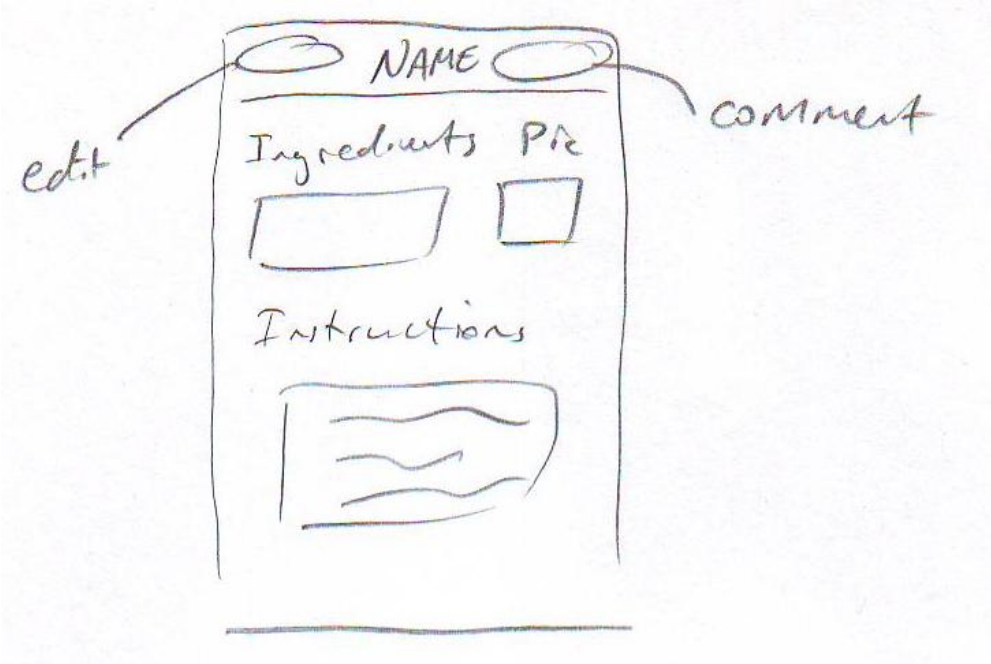


Once the user types in an ingredient, they will be brought to this screen. For this example, the user has added “cranberry Juice” and “Orange Juice”. These are buttons, and like in “Adding a Drink”, they can remove these with an “X” on the right side. The field for “Pineapple” is a text-field, and upon an “enter” key or hitting the “OK”, the ingredient will be added to the above list, with the potential to be removed. It is assumed that the user, upon looking for a drink, will not specify an amount. At any point in time, the user can hit the “Suggest” button and they will be taken to a listing of drinks that match the ingredients.

Task: Finding a Drink (cont'd)

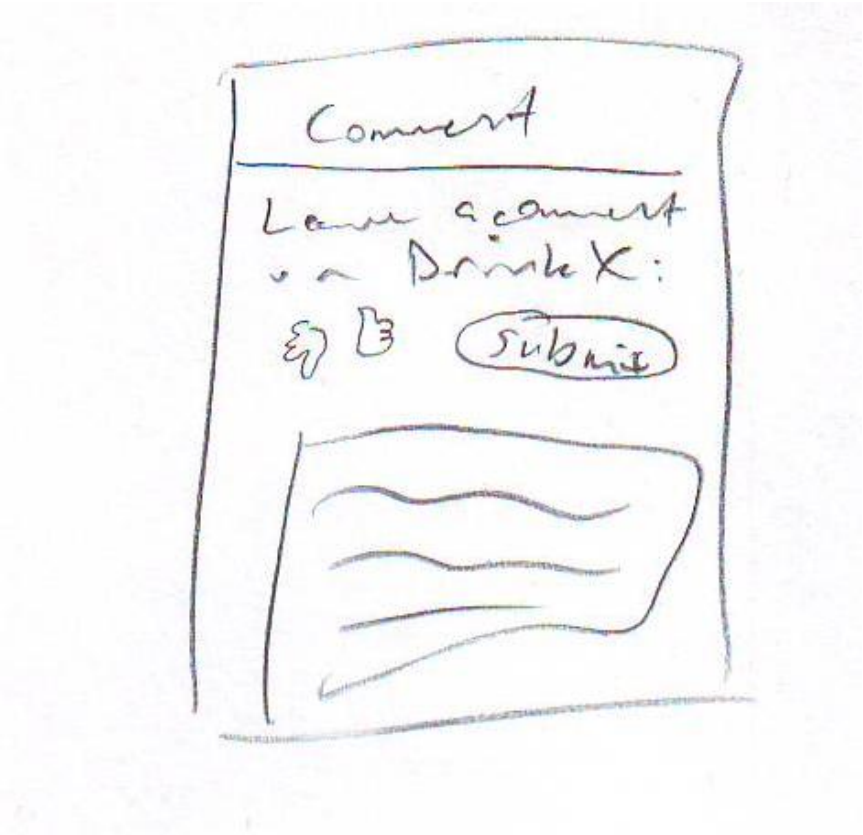
Upon hitting the suggestion button, the users will be brought to this screen. It's a simple listing of suggestions that Fat Charles presents to the user. It's scrollable vertically, but can detect a position change in the device and can change to a horizontal list if the device is horizontal or turned horizontally. These are ordered based on thumbsup/thumbsdown (see Task: Leaving a comment).

Task: Finding a Drink (cont'd)



When the user selects a drink, this is the screen they will be brought to. It's a typical format for a recipe, with ingredients first, a picture to the right, and instructions below. The user has some choices, tapping the picture will allow a pop-up of a bigger picture. Then they will also have the option to scroll vertically to browse the instructions/ingredients. At the top are a pair of buttons, allowing the user to edit or comment on this particular drink. This header bar is kept in-position, so even if the user scrolls downward, they will still see it. Editing a drink will basically lead to the "Add a Drink" format, but with the information pre-filled. At the bottom, the user can see the comments about the drink if the scroll past the instructions.

Task: Leaving a Comment



This is the comment forum. The main area is for a textbox where the user can leave their input. However, prominently at the top are a thumbsup/thumbsdown and a submit button. This is the most generic form of giving feedback, and is how we index the searches, using +1 for a thumbsup and -1 for a thumbsdown, and a simple aggregate sum to determine the "rank" of each drink. If the user hits "enter" or hits the "submit" button, both will submit this comment to the drink. Upon submission, this brings them back to the page of the drink, and if the user chooses, they can scroll back down to their comment and edit it.

Analysis

This interface allows the user to type, because it is the most specific way to say what a user wants and doesn't want, without having to navigate through a hierarchy of menus. We reduce the amount of typing necessary by doing things like auto-complete suggestions. We do realize that typing on a mobile screen, or even with a physical keyboard, is an expensive task on a mobile device, so we try to limit it as much as possible, but we do incorporate typing as a primary means of input because the user can be specific using this form of input. Much of this design is centered around making a good interface with reduced space. As such, the default is to put the items which need to be the most visible at the top, because input comes at the bottom.

Learnability

We expect this interface to be very learnable. With some helper text, we can direct the user to what we want in the most direct fashion. This disappears on input, so it acts as a perfect guide. Any user that is accustomed to inputting data into a form type interface online will get accustomed quite easily to this interface. The home screen is very Pandora-esque because it is multi-functional. Common themes are used, such as large "X" buttons on things that can be cancelled/removed. We basically think of this whole interface as a form, and you can remove the items you don't want. Additionally, we take large indicators from external sources, such as a header bar with two buttons on either side, very resemblant of iPhone. Additionally, the consistency with a recipe format should also help. Lastly, the forum-like comment style at the bottom, and the universally understood thumbs-up and thumbs-down are also very intuitive.

Efficiency

All items are only a few keystrokes away. We think that by being able to explicitly specify what they want, instead of navigating to something generic like "Sodas", we can remove the submenus and give the user a large amount of efficiency. We add in redirects to the end of the form submission to eliminate the need to navigate somewhere else. Again, we realize that typing is not always a convenient option, but we think that the trade off between specifically saying what you want will help increase efficiency overall.

Safety

By allowing users to remove selections or to backspace in their typing, we expect this to be a very safe interface. We avoid things like confirmation dialogs, but we think that by having typing safety, that users will not make as many mistakes (such as navigating into a wrong submenu by accident). Additionally, the back button built into the browsers is very convenient, and if we were to simply use query strings in our URL, would allow the user to fully go back without resubmitting to servers. The large X buttons and changeability of the form elements is also a very safe feature, so users can review and edit their items before final submission.