Design Decisions

1. Users can only see the actions they are allowed to do.

When designing a new layout for Fritter, one question that came up is what actions can a user see? In A1, users can see every action at all times. Moving forward, however, I decided to only have the actions a person is allowed to do to be visible. This really just means I create two versions of the interface, one where a user is signed in, and one where the user is not. This means that non signed in users get specific actions like create account and sign in, while signed in users get specific actions such as change username or password, delete account, log out, create freet, and view my freets. Both interfaces has a 'view all freets' and 'view freets by author' function. This ultimately leads to better design in my opinion because users are not having to question what things they can and cannot do when interacting with my interface, I handle all that decision making for them.

Alternatives Considered:
The user can see all actions (login, logout, sign in, sign out): I quickly found this alternative and decided not to implement it, as I felt that it would lead to poor design. I wouldn't want a user who is not signed in to think that they can sign out or change their username and have to return some message to them. These type of interactions are why I did not end up choosing this alternative, as I felt the one I chose was more intuitive for users to understand.

2. A deleted Users Freets are not deleted

One thing I had to decide when designing fritter was to figure out what to do with the freets of a user who just deleted their account. Does the account's freets go away? Do the freets stay there? I ended up deciding to keep the freets existing, simply because I thought that it gives the user more freedom in their choices. If a user wants to get rid of their freets and delete their account, they can delete them both separately, if they want their account deleted but the freets to stay on the server, they also have that ability.

Alternatives Considered:
User's freets are deleted when they delete their account: This was really the only other alternative I was considering. Once again, I thought both approaches would be valid, however, I think sticking to not deleting the freets simply give the users more options, and leads to better design that way.

3. Editing and Deleting Freets is done through the freet list

This design decision is really prefaced around one question: how do users select the freet they want to edit and/or delete? In the past a1 assignment, users would type the freet id to indicate which freet they were referring to, but I felt this system was tedious and not very good design for the user. The design I ended up choosing was to simply have every freet written by a signed in user to have an edit and delete button associated with it. This was whenever a user clicked on "view all freets", "view my freets", and "view freets by author" option. Anytime a freet was displayed in the freet lister and the user's account was the creator, the user can edit and/or delete the freet. The reasons why I chose this decision was because I thought it was very easy to select what freet you were trying to edit, and made it a very interactive process: you see a freet with a mistake, you want to edit or delete it, you have the option to do it right there in the app by simply clicking on the button of that freet. I think this ease of use was the main reason for picking this decision this way.

Alternatives Considered:
Entering a freet id to edit and delete freets:
This alternative is similar to what was presented in A1. I personally don't think this is good for design, as the id's that the freets have are very long and can be hard to copy/paste or transfer over. A user is not going to have an easy time then to select their freets that they want to edit or delete. Because of these reasons, I felt not to consider this alternative over the one I went with.

Having a separate list of freets for you to edit / delete:
Another alternative I thought of was that when a user said they wanted to edit or delete a freet, and app would show them all their freets then and then have the user select their freet and edit/delete it. I felt this approach would have been fine, but since the users are already going to be looking at freets in the freet list, and also have a 'view my freets' option, I thought that having this extra step was not necessary from a design standpoint. It isn't that this alternative is bad, rather I don't find it better than what I ended up doing, as I felt this extra step for the user was not needed in the process of editing a freet, so I chose the other process.

4. Creating an Account Automatically Signs you in

One design decision I had to figure out was whether creating an account would log you into the account or simply create the account and require the user to log in afterward. Going back and forth, I decided to have the app log you into the account after you create it. My rationale for this decision is that it seemed the practical step forward: a user creates an account because they want to get into the account and put freets, so I simply connect those steps for them. I furthermore looked at other apps with logins, such as facebook, twitter, etc, and I also find that these platforms log you in oftentimes after you create an account. These were all the factors that went into my decision.

Alternatives Considered:
Creating an account does not automatically log you in: This was really the only other alternative I was considering. I feel this alternative would have also been a valid approach to the design decision I had, however, I simply felt that it was more intuitive to have the app log you in afterward for the reasons I listed above.