Story UFC-185: Account Creation

The following document will go over the design of account creation, as outlined in Story ticket UFC-185.

Acceptance Criteria

For reference, the following are the acceptance criteria for this ticket:

- I can click a button to choose to create a new account that sends me to a form to fill out.
- I can fill out the form to have an account created.

Design Brainstorming

OK, because we're now using `react-hook-forms`, we're gonna need to really rethink how we structure the `AccountForm`. Right now, it's just a mess of handling inputs and whatnot, but I think most of this will get cleaned up from using `react-hook-forms`.

My *biggest* concern is still (what I consider hacky) use of `useResetStateFromProp` to switch the form from being for creating to being for editing. Obviously, this ticket shouldn't have to deal with that yet, but I *know* we're going to have to deal with it, so let's just deal with it upfront.

Then again, I'm just generally unhappy with how we've been doing these editing forms. It just doesn't feel right that, to trigger an edit form, we store an ID somewhere in the store, the form picks that up, and derives an *initial* state from the current state of the data. Then, if the stored ID changes, we have to re-initialize the whole form. Or if it clears, we just throw the form out and reset everything.

Like, I just want a way to 'invoke' a form with an initial value and then detach it from the store. Even though I can *feel* that that's just wrong.

Either way, since we're going to be using uncontrolled inputs with `react-hook-forms`, we're going to need a way to reset the `defaultValue` of the inputs when the account changes. This React - change input defaultValue by passing props post seems to have a good solution: add a key to the inputs with the account's ID and set the `defaultValue` to the account's value. Then, whenever the account changes, it'll bust the inputs and re-create them with new default values. That's actually pretty clever. And since we're not storing any internal state for the input values, there's nothing else that needs to be busted.

Actually, that might just solve all our problems right there. Here's the new flow:

- Set the ID in the store.

- Form picks up ID and makes itself visible.
- Inputs derive default values from the account.
- User can make changes.
- User can then do one of the following:
  - Close the form, which dispatches an action to clear the ID;
  - Submit the form, which grabs the input values and clears the ID;
  - Or edit another account, which changes the form's account, busts the key on the inputs, and updates them with new default values.

I think I'm really starting to come around on the uncontrolled inputs train. If only because `react-hook-forms` makes them so easy to work with.

Something of question is whether or not, if we're deriving whether or not to show the form based on whether or not the edit ID is set, we still need the `isVisible` state. And as much as I briefly questioned it, I also briefly realized that, yes, we do need it. Cause how else are we going to show the form when *creating* an account? So yeah, it's sticking around.

And I think with that, we've decided that we don't need to change anything from the redux side (hooray!) and just need to take advantage of the power of `react-hook-forms` to simplify the implementation of the `AccountForm`.

Navigating to the Form

A thought occurred to me... I've been thinking about how to best handle showing/hiding the forms, particularly now that we've added a wrinkle into the existing infrastructure with the `Make Another` button.

And it hit me: why don't we just register the forms on a route? That way, instead of controlling them with data in the store, they just show ..

And I just realized why this is a horrible idea: how would we retain the users current page with just the form overlaid on it if we completely change the URL?

Hmm... back to the drawing board.

BTW, I originally had this thought because I wanted to enable back-button-to-close-form functionality. I still want that.

Well, I just came across this: Building a modal module for React with React-Router - LogRocket Blog

- Basically, it's a way of tricking the react-router `Switch` component into using the *last* URL, so that we can change the URL and display the modal.
- Basically, exactly what we want.

OK, now that we've determined that this is possible, here's how it's going to work:

- `onClose` on the `Sidebar` actually just means going backwards in the history.
- The edit version of the form will pull its ID from the URL instead of from the store.
- Done.

Side Tangent

I think, in order to encapsulate the calculation of the whole 'previous location' thing, we could probably get away with just sticking it in a selector. We'd need it to be a two-tiered selector (like how the ID selectors work) because we'd need to save the previous location as a variable inside the closure, since we don't want to bother storing in the store.

Component Breakdown

Atoms

- OptionCard
  - Can be used as a radio button like in the AccountTypePicker or as a 'checkbox' button like in the pickers above the Account/Transaction lists.
- FormSectionHeader (e.g. 'What type of account is this?')
- Sidebar
- [modification] Dropdown (add the triangular tip)
  - Decided against this. Will just use plain Dropdowns.
- [modification] Input (need to add a symbol prefix option)

Molecules

- CollapsibleSection (LinkButton with down chevron + children)
- RadioGroup
- AccountTypeOption (OptionCard as a radio button)
- AccountTypePicker (AccountTypeCard x4)
- GlobalAddButton (ShadowButton + Dropdown)

Organisms

- AccountForm (OverlineHeading + IconButton + LabelledInputs + CollapsibleSection + FormSectionHeader + AccountTypePicker + SubmitButton + LinkButton)

- [modification] AppNavigation (add the GlobalAddButton -- small and large)

Scenes

- SidebarAccountForm (Sidebar + AccountForm)

Tasks

- Create and add the global action button to the AppHeader.
- Build the AccountForm.