Phase 3: Letting the User Know

This phase pertains to the following acceptance criteria: 4, 6, 7, 8, 9, 13

The third step is enabling the user to know everything about the syncing process. This will give the user some control over when things are going to be synced, and how to deal with failures.

Design Brainstorming

Old Notes

In particular regarding failures, instead of automatically rolling back a failed change, the user should be given the option to know which particular change failed, why it failed (actual error message), and be given the ability to manually retry the change. This could manifest as a modal dialog or an indicator on the form for the, e.g., the transaction or the account.

The "Sync" Indicator

[May 7, 2021]. It's been a while since we last revisited this epic. Specifically this phase. This phase is the last phase to implement. Yeah that's right, we did Phase 4 before Phase 3, sue me.

Having now looked back on the UI design, the acceptance criteria, and where we are in the overall life of the app and what I want, I'm thinking that we need to make some changes to the original design.

First of all, I guess the original design was made assuming we'd have two-way sync? I don't remember, and Navigation doesn't detail the reasoning behind why the "Synced" indicator is as it is.

As such, the first change I want to make is to change it from being "Synced" to "Saved". This is more strictly accurate for how we currently handle things, since we still don't have two-way sync.

Secondly, the whole design of the dropdown with "Recent Activity" is kind of nowhere close to being applicable. That would require that we have a full CRDT implementation so that we could track all changes, but we do not yet.

As such, that kind of 'Activity' dropdown would have to be repurposed more so for showing either reasons for the current state of the system (e.g. error messages for why the system is offline) and/or effects that have yet to be saved.

If we go the route of showing unsaved effects, then that opens the can of worms for manual retries, showing all error messages, having to handle rollbacks more gracefully, etc. So... I'm hesitant about it.

What's the bare minimum we could do? Just show the state of the system. Is it online? Is it offline? Are there any unsaved effects?

I *think* we'd be good with the four states we've already designed (Saved, Saving, Offline, Error). Here's how they'd work:

- Saved:
  - The system is online.
  - There are no effects in the queue.
- Saving:
  - The system is online.
  - There are effects currently being processed in the queue.
    - If the user clicks on the indicator, a dialogue showing the current effects is shown, along with each effect's current status (pending, processing, errored).
- Offline:
  - The system is offline.
    - This might be because the user's network is done, they aren't connected, or our Backend is offline. Additionally, encountering a Network Error during the processing of an effect (e.g. the database is done) should also count as offline.
    - The reason for being offline should be shown to the user if they click on the indicator.
  - There may or may not be effects in the queue.
    - If there are effects in the queue, they should be shown in a dialogue when the user clicks on the indicator, with all the statuses as just 'pending'.
- Error:
  - The system can be online or offline.
  - There is at least one effect in the queue and it has been attempted and failed at least once.

    - Clicking on the indicator should open a dialogue showing the queue of effects, along with the error status for those that have failed, along with error messages.

Additionally, in the dialogue that opens when clicking the indicator, there should be a "Sync" button that does the following:

- Queues all of the fetch effects (accounts + import profiles) that would normally happen as part of app boot. This way, the user can use this button as an alternative to refreshing the page.

- Triggers a processing of the queue. This way, the user can manually force processing the queue after, e.g., coming online again without refreshing the page.

Note that the fetch effects should always be displayed last in the dialogue, since they are always processed last. This is only relevant to note since they are in a separate queue.

In this version of the implementation, I don't think we should have per-effect retries, advanced rollbacks, or anything else. It's taking a piece of internal state and making it visible.

Now, there is a very valid discussion to be had about whether or not this is even worth implementing. That is, do users even want this? I would be more inclined to think they wouldn't and that they'd much rather have me working on more finance-related features, or even the dedicated desktop/mobile apps. And even if I look to myself, only *I* only kinda want the feature; it would just make my debugging process easier.

So I think it's fine to hold off on this feature until people want it. The best idea would be to have a public "upcoming features" board of some sort where people can vote on what they want.

Design (Finalized)

TODO

Stories

- TODO