Redesign Plan

This document will outline the plan for implementing the redesign. At this point, the high-fidelity Figma designs have already been completed, so now it's just a matter of implementing everything, from all the old features to all the new stuff I want.

The plan will be split into phases that more-or-less correspond with epics.

Phase 1: TypeScript Migration and New Tooling

Epic: UFC-159 - New Redesign Service

The first phase of the redesign is to not work on the redesign at all. Instead, I want to migrate the Frontend to TypeScript and introduce some new tooling that I've been meaning to put into practice.

The migration to TypeScript will consist of creating a new Kubails service (henceforth called `redesign`) and migrating all of the existing *non-UI* code to TypeScript.

- Obviously, since all of the components (i.e. UI code) is going to get re-implemented as part of the redesign, there's no point in migrating the old UI code to TypeScript.
- Additionally, we're making a new Kubails service so that we can keep the old uFincs running while we implement the redesign. A nice benefit to using Kubails.

Additionally, there's a good bit of tooling that I want to introduce that I didn't feel like enforcing on other people. Now that it's all me, I can be as strict as I want! (on myself!)

This tooling includes the following:

- Storybook (+ a whole whack ton of addons)
- StyleLint (for Sass linting)
  - Actually, thinking on this some more, now that we have Prettier to enforce consistent styling, we don't *need* a Sass linter.
  - In fact, we technically don't need ESLint either, except for the fact that it lints for much more than just styling issues.
  - The only reason we'd need StyleLint is if it had useful checks beyond just styling. Although personally, I just want a check to make sure rules are in alphabetical order.
  - Having now just looked into the StyleLint rules, there *is* a plugin that can enforce alphabetical ordering. So... worth it!

- Prettier (for enforcing consistent styling)
- React Hook Form (not that this really needs to be *setup* or anything)

As such, we'll need to perform the following tasks:

- Create the new `redesign` service with TypeScript and all of the above tooling installed and configured (along with all of our existing tooling like ESLint) using a fresh install of `create-react-app`.
- Migrate the `models` and `dataObjects` (and give them a top-level folder, but rename `dataObjects` to `structures`).
- Migrate the `services` (and give it a top-level folder).
    - Also, convert some of the helper files into services (e.g. `dateHelpers` should really be something like a `date` service).
- Migrate the constants (and give them a top-level folder).
- Migrate the `hooks` (and give them a top-level folder).
- Migrate the rest of the `utils`.
- Migrate store `utils`.
- Migrate the store `slices`.
- Migrate the store `sagas`.
- Migrate the rest of the `store`.

Retrospective

Having now completed the goal of this first phase (migrating the frontend to a new redesign service that uses TypeScript), I just want to say that I kinda went way above and beyond the initial scope. Now the Backend has *also* been migrated to TypeScript. On top of being upgraded to the latest Feathers v4.

And on top of *all* of that, I also did the *laborious* (not really) task of renaming the stupid `account1/2` fields of the Transaction model to `debitAccount` and `creditAccount`. Yes, that's right, we're using proper double entry accounting now! (well, *more* proper)

Sure, it took double the time that was it supposed to take -- 2 weeks instead of 1. But to be fair, I *did* have the frontend migration done in one week; I just figured that since I'm in this mode, I might as well migrate the backend while I'm at it.

Of course, I've also gotten all of the tooling setup too. Prettier? Just fantucking tastic. I swear, I will never be able to work on a team that *doesn't* use Prettier; it'll just save so much bikeshedding. I can see that *while only being a team of one*.

Storybook? OMG, amazing. All of the addons it has, and just the overall workflow it enforces, makes it seem like that I'll be able to get the work of many more people done all at once. Not to mention the whole documentation aspect of it that doubles as an easier way of performing manual testing (but also enables an easier transition to automated testing). Although some of that feeling certainly comes from TypeScript. Speaking of which...

My god. I was so right about TypeScript. Yes, it's a bit more tedious to use than Python's type checker/system, but it's a lot stronger. It's already caught a *bunch* of things just from migrating the old frontend's business logic. It's so good, in fact, that I've started using VS Code more, because the intellisense is just that good. I mean, I could have spent a couple hours trying to rangle the intellisense into vim, but I've gotten my vim bindings and other shortcuts setup well enough in VS Code that I could reasonably think that I could use VS Code full time and live pretty well with it.

Overall, yes, I spent an extra week on this phase, but I think it was more than full worth it. These extra steps will more than make up for themselves in the future.

Phase 2: Re-Implementing the Existing Features

This second phase (which will be made of several sub phases according to each epic) will be for focusing on re-implementing all of the features that already exist in the current uFincs, more-or-less. I say more-or-less because it's inevitable that I'll stray into implementing some new stuff or removing some stuff (or just postponing some stuff totally not indefinitely... *cough* charts *cough*).

The sub phases (and accordingly, the epics) will mostly follow the progression that we went through when first building uFincs. This is as follows:

- Authentication (login, sign up, logout)
- Accounts (list, dashboard summary, new form, edit form, details, deletion)
- Transactions (list, new form, edit form, deletion, import)
- Basic Analytics (net worth, expenses, whatever I feel like chucking in)

What's noteworthy is that the lists for the Accounts and the Transactions are both now significantly improved (and thus, more complex) over their old versions. Things like search, better type filtering, *much* better time based filtering, etc. I'll probably mix these things in during this phase because they're what *I* really want, so they'll get done.

Another big (new) thing is recurring transactions. I don't know if those should be done in this phase, or maybe as just another sub phase, because those in-and-of-themselves will be a pretty significant implementation.

Basically, what I'm trying to say is that nothing is set in stone and I'll probably jump between implementation features as I feel like it.

Epic: Authentication

- I want to sign up.
- I want to login.
- I want to logout.

Epic: Accounts

- I want to view my accounts.
- I want to create an account.
- I want to edit an account.
- I want to delete an account.
- I want to filter my list of accounts by type.
- I want to view the details of an account.
- I want to view a summary of my accounts on my dashboard.

New

- I want to filter the date range of my accounts.

Epic: Transactions

- I want to view my transactions.
- I want to create a transaction.
- I want to edit a transaction.
- I want to delete a transaction.
- I want to filter my list of transactions by type.
- I want to import transactions.

New

- I want to filter the date range of my transactions.

Epic: Basic Analytics

- I want a chart showing my net worth over time.
- I want a chart showing my expenses and income over time.

New

- I want a chart of account's balance when viewing the account.

Retrospective

Today is September 28, 2020. It is the day after I have finished implementing the redesign of the Transactions Import process, resulting in the completion of Phase 2 of the uFincs redesign.

This means that the redesign of uFincs is now, at a minimum, at parity with the original implementation!

It took a bit longer than I had hoped (was originally expecting it to take June - August, so 3 months; instead it took 4 months), but considering the absolute *quality* of the end result (half sarcasm), I think the extra month was worth it.

So what next? Well, we have 3 months left in 2020, before the world *actually* ends, and I had brainstormed up a list of things I'd need/want before starting to 'go-to-market':

- Encryption **[1 month]**
- PWA web app **[1 - 2 weeks]**
  - Online status + queued effects display
    - This is actually part of Phase 3 of Offline First, so it won't be done here. Everything below is part of Phase 2, so it'll be done.
  - Notification for when the app can be used offline
  - Notification for when app can be updated + Settings area for updates
  - The general PWA configuration
- Minimal user settings (i.e. change email/password) **[1 - 2 weeks]**
- Marketing site (i.e. the 'landing page') **[1 - 2 weeks]**
- Mailing list (gsuite, mailchimp?) **[mostly part of the marketing site]**
- Start content marketing through blogging (personal blog? company blog?) **[not dev time]**
- Setup billing (Stripe) **[2 weeks]**
- Onboarding process **[2 weeks]**
- Fixing up all the little things in the ancillary todo list **[1 week]**
- Increasing size of production database disk (and other such devopsy tasks) **[trivial]**
- Terms of Service/Privacy Policy/etc **[not dev time]**

- Final logo design **[a day or two]**
- Recurring transactions (or at least, hide the current view) **[2 - 3 weeks]**

Everything put together, at the worst, would take another 4 months, with 3 months being the lower bound (how convenient that there's 3 months left in the year...) -- not including content marketing time, which would be weaved in-between dev time.

I've kinda already decided that Phase 3 will be e2e encryption, so that'll be the bulk of October.

After that, November will likely be a combination of offline-first improvements through PWA as well as general administrative improvements (onboarding process, user settings, etc).

Then December would be when we start to really focus on marketing/business related things, what with the marketing site, billing, logo design, content marketing, ToS, etc.

So... where does recurring transactions fit into all this? Uhhh... good question. Maybe that'd be the big app feature of December? So e2e encryption → October (Phase 3), PWA → November (Phase 4), general administration (user settings, onboarding) → November (Phase 5), recurring transactions → December (Phase 6), getting ready for users (content marketing, marketing site, ToS/Privacy Policy, final logo design, Stripe/billing) → December (Phase 7).

Is it realistic to think this would all be doable in 3 months? Absolutely. If I think otherwise, then it won't get done.

That means we're basically targeting an end-of-year/start-of-new-year date for demoing to and taking the wraps off this thing for the world.

Well, if Phase 2 was anything to go by, then the next set of phases (i.e. the planned work for the next 3 months) will take 4 months, so ya know.

But Phase 2 is now behind us. Forwards, ever more!

Phase 3: End-to-End Encryption

OK, so Phase 2 was a *fairly* large phase, taking 4 months total. Phase 1 was considerably smaller, only taking a couple of weeks. I think, going forward, phases will be more along the lines of Phase 1 than Phase 2 -- small, self contained features. Essentially, phases will more or less correspond to epics.

With this Phase 3 of e2e encryption, I expect things to take roughly a month.

And now to figure out exactly what we need out of this 'e2e encryption' feature. AKA, what is our acceptance criteria?

Acceptance Criteria

- I want all of my personal data (accounts, transactions, import profiles, etc) to be encrypted using my 'password', so that the server (aka uFincs, aka the operators of uFincs) never have access to my raw finances.
- From my perspective as the user, very little should change from a UI/UX perspective.
- I want to know that I can't reset my password without losing all my data.
- I want to be able to change my password while still maintaining my data.
- I want a loading screen when first loading the app to show that my data is being fetched and decrypted.
- I don't want a loading screen after my data has been cached client-side; it should just fetch and decrypt from the backend in the background.
- The encryption/decryption processes should work in a background thread so that the UI thread isn't too heavily impacted. It should also, ideally, be multi-threaded for even faster performance.
- The encryption process needs to be more-or-less a self contained library, so that we can open source it.

The rest of the planning for this phase will done in Idea: Client Side Encryption.

Retrospective

Today is November 3rd, 2020. I'm sure today will go down in history for... various reasons.

In other news, e2e encryption (or at least, the first naive implementation of it) is done! That is, Phase 3 is done!

I had originally planned it take the month of October. Considering I basically nailed that on the head, I guess it could have been done faster? Oh well, it's done. The majority of user data is encrypted (everything except IDs and dates right now), we finally have some email integrations (password resets), and all is good in the world.

I'm sure, sometime in the future, I'll want to implement encryption for dates, but I don't think that should be a priority right now.

So what should be a priority right now? Well, the original plan from the Phase 2 retrospective set out Phase 4 (the next phase) as being for making the app a PWA, but I've decided to switch it with the 'general administration' stuff that concerns onboarding and the rest of user settings.

Why? Because I want some time to research and spike out PWA stuff before I plan it's implementation for uFincs. This is what I've done for other features (most notably, e2e encryption), and I've found it to be quite effective. So I can weave in this research/spiking while procrastinating on the onboarding process.

Having just reviewed the giant checklist of stuff I laid out in the Phase 2 retrospective, while we're technically on track, it's still just a *huge* pile of stuff to do. I really feel like I should be working on marketing stuff (like the marketing site), but... you know how it is... There's just that 'build distribution channels' and 'gain traction' rhetoric that keeps hitting me.

Whatever, Phase 3 was a success, so let's make Phase 4 a success as well.

Phase 4: General Administration

OK, so Phase 4 (general administration) kinda has a meaningless name, but it really means two things: onboarding process and user settings.

The onboarding process is the more important of the. User settings will encompass the ability to change your account email (since we currently only have password changes), as well as data exports/imports, and account deletion.

Honestly, I think we should push the Import Profile settings down in priority since it's really not that important relative to everything else. As such, this phase will also include hiding the settings item for the Import Profile settings.

Thankfully, we have the designs for both the onboarding process and user settings pretty much complete, so this'll be mostly just coding work. As such, I expect this phase to only take **2 weeks**. Cutting the import profile settings will definitely help us to hit this deadline.

Now let's figure out what acceptance criteria we need for each part (onboarding and user settings):

Acceptance Criteria (Onboarding)

- After signing up for a user account, I want to see the onboarding process.
- I want to see my current progress through the onboarding process.
- Each step of the onboarding process should be for picking/creating accounts for each account type (i.e. asset, liability, income, expense).
  - As such, there should be 4 main steps.
- At each step, there should be a pre-configured set of accounts that I can choose to use.
  - I can also choose not to use them.
- At each step, I can add other accounts to make up for those that aren't in the list.
- For each account I create/choose, I can specify an opening balance for it.

- At the last step, I want to see a summary of all the accounts that will be created for me.
- Once I am satisfied with the accounts, I can go to the app and have all the accounts created.
- I want to be able to skip the onboarding process to get straight into the app.

Stories

- I want to create accounts right after signing up.

Acceptance Criteria (User Settings)

OK, so originally I had planned (designed) for being able to export data three ways: in CSV, in JSON, and as a 'proprietary' backup format that we could use to restore from.

However, I don't really know what this 'proprietary' format would look like *besides* being just JSON. Additionally, I don't really want to have to deal with exporting to CSV, since that's actually quite a bit more complicated than just a JSON file, since we'd need multiple CSV files since each file can only have 1 schema (column format).

As such, I think we should change the export/backup formats to just be the following:

- Plain JSON (for use as an export format and a backup format)
- Encrypted JSON (for use as a 'proprietary' backup format)

This way, users can export their data in plain JSON to use with other applications, but still be able to use it to restore their data to uFincs, and they can make encrypted backups if they really care about data privacy/security and only care about restoring data.

- I want to be able to change the email address of my user account.
- I want to be able to delete my account and all its data.
- I want to be able to export all of my data as a single plain JSON file for use with other applications.
- I want to be able to export an encrypted backup of my data.
- I want to be able to restore a backup of my data that was made in plain JSON.
- I want to be able to restore an encrypted backup of my data.
- I want my backups to have a schema/version number, so that any future changes are known and backwards compatible for restoration purposes.

Stories

- I want to change the email address of my account.
- I want to delete my account.

- I want to backup/restore my account data.

Retrospective

Today is November 26, 2020. As you can tell, we're kinda a bit late when it comes to Phase 4: General Administration.

In my defence, the total days worked on stories specifically for Phase 4 *was* just about 2 weeks (maybe slightly more). The rest of the month I filled in with other tickets (most notably: transactions search, transaction summaries, and recent transactions list) that took up a non-trivial amount of time. They were tickets that were bugging me and didn't really fit into a whole phase in and of themselves, so I think it's good that we just got them done with.

Anyways, back to the topic of Phase 4. Well, I don't know if there's much to talk about it; it's done! This was very much a tedious phase to get through though; these weren't exactly the most exciting stories to work on (which is probably why I wove in all those other stories).

If anything, I'm happy this phase is done just so I can more quickly backup/restore data from prod to local dev. That'll be faster than having to get the latest database backup and restore it via CLI.

Other than that... I don't know. Morale check? It's meh. Definitely had my ups and downs throughout the month. I think mixing in those more interesting stories definitely helped out. I was especially down when implementing the Onboarding process; that shit was so boring. Not to mention it felt like it took way longer than it did/should have. Was just procrastinating and not really wanting to work on it. Then I got in some quick wins with the email change and account deletion stories -- those were very fast to implement. Backup/Restore took a bit longer than I expected (mostly because of testing woes), but nothing way out of the ordinary.

Speaking of testing woes, man, it really sucks how long the e2e test suite takes to run (currently, around 25-30 minutes). I feel like I've been slacking on (meaningful) unit tests. I've more or less dropped testing the sagas, if only because it really felt like it was just testing the implementation details. If I really wanted to test the sagas, I think I should go up a level and test functionality through the entire store (i.e. spin up a store, emit actions, check for actions/state). I feel like that'd bring me more confidence than just mindlessly writing 'unit' tests for each saga.

But it's not just sagas, it's definitely unit testing throughout the code base that's been taking a hit -- mostly around edge case tests. I've got a solid base of 'happy path' tests, but there's definitely some stuff that should have been caught with unit tests when I was refactoring some stuff (`objectReduce`) that was only caught through the e2e tests.

But hey, at least the e2e tests *do* give me great confidence that things are working. I can usually debug my way pretty quickly even if an e2e test fails, so it's not *too* bad. Much better than having to test everything manually, of course.

So yeah, Phase 4. It's done. The month is almost over. The year is almost over. Time to move on and keep on plugging. Next stop: PWA.

Phase 5: Offline-First - PWA

OK, it's finally time. Finally time to make this *stupid* web app of ours into a PWA. Time for it to get *truly* offline first.

Now, there's a lot of things that could fit inside this phase. I'll start with the bare minimum: making the app work offline to the point where the page can be refreshed while offline and it'll still work. This mostly comes from setting up the Service Worker that'll enable the 'PWA'ness.

The next bare minimum piece is letting the user know when the app is usable offline and when there are new updates available. Both these things will be done using toasts.

So far, the above things are really just Phase 2: Introducing Service Workers of the Offline First epic. So that's all well and good.

But what *else* could we do during this phase? Well, there are 2 main things that I have in mind:

i. Adding a 'Check for Updates' item to the settings/user dropdown menu. This way, users can manually check for updates in case they don't know that they just need to refresh the page or in case they dismiss the upgrade toast.

ii. Building the 'notification center' that shows the user which effects have been queued up to be done once they are back online, while also showing their online/offline status. Basically, the UI for the Offline Request Manager.
    - This starts to dip into Phase 3: Letting the User Know of the Offline First epic.

For the sake of just getting through the giant list of things to do, I think I'll put 2. outside the scope of this phase. Implementing 1. *should* be simple enough that it's probably only a day or two's time, so I think we can keep it in.

So that means that this phase will cover the following acceptance criteria.

Acceptance Criteria

- I want to be able to refresh the page while offline and still have a usable app.
- I want to be notified that the app can work offline.
- I want to know whenever there is a new version of the and that I can manually refresh to get the new changes.
- I want to be able to check manually for new updates to the app.

The first 3 points are Acceptance Criteria 5, 11, and 12 of Epic UFC-145: Offline First, which comprise Phase 2: Introducing Service Workers, while the 4th point is a new criteria that'll get added to the Offline First epic criteria (I guess, number 15).

The rest of the details regarding these acceptance criteria (i.e. story breakdown, technical implementation details, etc) will be put in Phase 2: Introducing Service Workers.

Other Stuff

On top of the above, there's also some technical work that I wanted to get taken care of. In fact, I want to get it taken care of before we even start the offline-first work. In *fact*, it probably *has* to happen before we start the offline-first work.

Of course, I'm talking about upgrading packages.

In particular, `create-react-app` needs to get to v4, TypeScript needs to go to 4.0/4.1, Storybook should probably be upgraded to v6.1, and React needs to go to v17. While we're at it, we should also probably upgrade to the latest Node LTS version as well as the latest NPM.

All of that should be fairly easy, although the only uncertainty I have is around the Storybook upgrade. I know they introduced some new (integrated) APIs for dealing with knobs, so I don't know if we'll have to port all of the stories to use the new interface or if everything is still backwards compatible. If things *are* backwards compatible, I'm going to opt to leave them as they are, just to save on time.

Other than those upgrades, I guess know would be a good time to also setup a CSP (Content Security Policy) for the frontend, since it currently doesn't have one. That shouldn't take too long.

I know we should probably upgrade `cert-manager` to v1 (since it recently hit v1, finally...), but I *really* don't want to deal with it. God, I hate `cert-manager`...

Oh, maybe now would also be a good time to investigate exactly which packages are being pulled into the production build and remove any extraneous ones. I'm pretty sure, because of some things that get imported from stories, that Storybook is being included in the prod build, so getting that out would be a good step.

Usage without an Account

Date: December 7, 2020

This is my official amendment to add implementing the 'no-account' mode as part of this phase of the redesign plan.

This corresponds with Phase 4: Going Fully Offline of the Offline First epic. Yes, that means that we're skipping over Phase 3: Letting the User Know for now.

I feel like being able to use the app without an account will be an important part of our marketing/pricing strategy, since we don't offer a free tier account. *This* will be the free tier -- complete access to use all the features of the app, just only client side.

All of the planning/design work has already been done in Phase 4: Going Fully Offline. In fact, I'm currently already in the middle of implementing Story UFC-303: No Account Mode. So now it's just a matter of getting it done.

Summary

OK, in summary, I think this phase will consist of the following:

- Upgrade `create-react-app`, React, TypeScript, Storybook, Node, and NPM. **UFC-298**
- Implement a CSP for the frontend. **UFC-298**
- Optimize the production build to remove unnecessary packages. **UFC-298**
- Make the app a PWA so that it works completely offline. **UFC-299**
- Implement the 'no-account' mode so that the app works without a user account. **UFC-303, UFC-304, UFC-305**

Retrospective

OK, today is December 11, 2020. We're now done with Phase 5: the app is now a PWA and the app can be used entirely without an account. In terms of time spent, it took damn near exactly 2 weeks -- 1 week for PWA and 1 week for no-account. Pretty good!

That means we're more-or-less on track with the Launch Plan.

You know what that means? Time to plan the next phase! And what exactly should the next phase? Well, that's the big question.

If you look right below where I'm typing right now (at the exact moment in time while I'm typing), Phase 6 would be for the Recurring Transactions. However, since we already moved Recurring Transactions out of scope for the launch (instead replaced with a simple feedback mechanism), that means the next phase will *not* be Recurring Transactions.

As such, we just have to go investigate our launch plan and... oh, well what do you know. The next step is Stripe integration and the marketing site! Wow! Hooray! Finally!

So yes, the next phase will be Phase 6: Getting Ready for Users (what is currently Phase 7). The main focus of this phase will be, as mentioned above, the Stripe integration and marketing site. On top of that, there will be some ops work that we'll need to finally take care of (increasing disk size, increasing replica count, maybe increasing node count, promoting `redesign` service to `frontend`, integrating analytics + error reporting, setup Slack alerting for sign ups, maybe upgrading `cert-manager`?, maybe migrating out of `us-east1` to one of the Canadian regions?).

Now, the real tricky thing is what *order* to do these things in. Should we integrate Stripe first? Or should we build the marketing site first? Well, I think the Stripe integration will be the more technically challenging of the two, so we should probably do it first.

However, I do believe there's an argument to be made that it'd be better to do the Stripe stuff after the marketing site stuff, since we want to know what the flow will be coming through the Pricing plan stuff of the marketing site, into the app, into signup, and finally into *actually* selecting a plan.

However, since we more-or-less know that *that* is going to be the flow (pricing page → sign up page → plan pre-selected (selected plan passed through as URL query param), then we *should* be fine.

So yeah, I guess this phase went pretty smoothly, and we got about 3 weeks to get the next phase done before the new year. Progress is being made!

Phase 6: Getting Ready for Users

OK, it's time for another 'finally time'. This time, the final time is that, finally, we're gonna have a proper marketing site! No more stupid, shitty, "here's out tagline and nothing else" site!

Oh, and Stripe. Mhmm, Stripe. Based on the fact that everyone and their mom has managed to integrate Stripe, we *should* be fine. Hopefully. Maybe. But it could go *real* wrong *real* fast, if some of the horror stories I've read come to pass.

Anyways, the goal of this phase is, as the title indicates: getting the app ready for users (finally!). As stated in the Phase 5 retrospective, that primarily means building the marketing site, getting Stripe ready to go, and doing some dinky ops work.

So, I guess we'll just do some stories. Implementation details will be planned at the story level.

Stories

- I want to be able to pay for uFincs.
- I want to visit a site where I can learn all about uFincs.
  - I want to know what the features/benefits of uFincs are.
  - I want to know how much uFincs costs.
  - I want to be able to login/sign up for uFincs.
  - I want to know what uFincs' Terms of Service and Privacy Policy are.
  - I want to access the marketing site at `ufincs.com` and the app at `app.ufincs.com`.

- I (personally) want to receive Slack notifications when someone signs up.
- I (personally) want to see analytics for the marketing site.
- I want the app to not fall over when there are a large number of users.
  - Increase production disk size.
  - Increase replica/node counts.

Retrospective

Time for the most belated retrospective *ever*...

It is currently February 10, 2021. The marketing site was finally finished last week. We're roughly 1 month behind schedule in terms of when this phase should have been completed. We all know what's to blame for that...

Anywho, time to move on and finally start launching.

After that, I'll be participating in the latest Startup School Build Sprint. That starts next week. I've set my end-of-sprint goals as the following:

- Finally launch publically.
- Acquire 2 paying customers.
- Build recurring transactions.

So, you know what's next? Building recurring transactions...

Phase 7: Recurring Transactions

I feel like implementing recurring transactions deserves to be its own phase in and of itself. It's just that (potentially) big of a feature that is also very important for the functionality of uFincs going forwards.

OK, so now it's February 2021. And we're finally gonna start working on this. Hooray. Planning. Bleh.

Just get shit done.

Story: Recurring Transactions

- I want to create recurring transactions.
- I want to view recurring transactions.
- I want to edit recurring transactions.

- I want to delete recurring transactions.

Retrospective

Written: March 23, 2021

As is tradition, this phase took a lot longer to finish than anticipated. Took about 5 weeks total to implement recurring transactions. This time, the excess time seemed to be a combination of low morale and actual technical issues. But 2x over the initial estimate is about in-line with how things are.

But with the completion of recurring transactions, I basically have everything I want out of uFincs. At least, as far as launching goes. That's right, *everything* on the launch plan (at least, as far as product development goes) has been completed! Now all we have to do is... actually tell people about it!

I mean, we've been getting like a sign-up per day for the last several days, but I have a *good* feeling those are just bots.

Anywho, uFincs is in a good place. Now it's just a matter of making money.

Phase ???

I'm just gonna dump stuff here that is getting re-prioritized.

Epic: Accounts

- I want to search my accounts.

Epic: Transactions

- I want to search my transactions by description. **UFC-197**
- I want to search my transactions by any field.
  - I don't really feel that this is a high priority story, so we're moving it out of Phase 2.
- I want to view a summary of my transactions list. **UFC-209**