Phase 4: Going Fully Offline

This phase pertains to the following acceptance criteria: 10, 14, 16, 17

The final step in making the app offline first is allowing users to use the app without an account. They should be able to sign in/sign up without registering, and be able to use the app completely between browser sessions without losing any data.

NOTE: This ability to use the *web* app without an account is just a stop gap until the desktop/mobile apps have been created. We might not want this when we start charging for accounts, but it's technically not incompatible with our business model.

Todo List

I'm keeping a list of all the things that will need to be changed to support the no-user account functionality:

- Need a link on the Login/Sign Up pages (probably below the forms?) to trigger the 'use without an account' flow
- Need to add a flag to the `users` slice for the 'no account' state
- Encrypted backups should be disabled
- Effects should be disabled (only commits happen)
  - This is basically done by not registering the Offline Request Manager slice/sagas.
- Remove the User Account settings section
  - Actually, it should be changed so that users can sign up for an account from here
  - Although, there should be some other, more prominent call-to-action to sign up for an account (maybe in the header, left of the Add button on desktop, and somewhere in the Settings on mobile?)
- Restores should work, just not the effect half
- Disable the encryption middleware (aka don't init it during login/app boot)
- Disable the data fetches from the app boot
- Disable the auth token refresh during app boot

- Need to make the onboarding sagas into offline request slices so that they're split into commit/effects parts
- Will need to modify `api.isAuthenticated` to account for no-accounts (i.e. yes you are 'authenticated' when you've 'logged in' with no account)
  - Should probably create a selector that returns this info rather than querying `api.isAuthenticated` directly
- Need to warn users that 'logging out' will clear their data from their browser
- Users need to be able to sign up after having used the app without an account
- Need to be able to create a 'demo' set of accounts/transactions if users don't want to do the onboarding process themselves

Design Brainstorming

The following is a brainstorming of ideas for what would be needed to be implemented to achieve each of the acceptance criteria above.

Link on Auth Page

I'm thinking we could put a "Use without an account" link below the auth forms. In terms of styling, white text with an underline (to indicate link) would probably be good enough, although we might want to not bold/use a slighter greyer shade to de-emphasise it relative to everything else, since we don't exactly want to *encourage* this option over the others.

Side note: On the pricing page of the marketing site (which doesn't yet exist), I'm thinking the first 'plan' would be the free 'use without an account' option. In this plan's card/design, the button that would normally be "Sign Up" would just be "Use without an account" and it would redirect the user straight into the app.

Same with the call-to-action button on the landing page.

User Slice Flag

I'm thinking the flag name we'll go with will just be `noAccount`. Setting it to `true` would indicate that the app should function in 'no account' mode.

Disabling Encrypted Backups

This should just be a matter of checking for the `noAccount` flag in the `MyData` section of the `Settings` and hiding the form when relevant.

I'm thinking we should probably create a hook that just uses `useSelector` get the flag value. I think using a hook as the interface will be easier than having to connect components *just* to access the flag. Of course, the hook can abstract it away to default the

value to `false`, since that's the original functionality (and can abstract away the case where the flag doesn't exist, for example).

Disabling the Offline Request Manager

There's a couple of ways we could go about doing this... Basically, the end result is that we either need to prevent queuing effects (so that the Offline Request Manager -- henceforth ORM -- has nothing to process), prevent the ORM from processing effects (e.g. by disabling/not registering the sagas), or by not registering the ORM slice/sagas at all.

My initial thought was just "don't register the ORM", but that's kinda not so trivial. Cause we register at first page load, regardless of what the user is going to do. So we'd need a way to *unregister* a slice/saga, which I'm not actually sure exists.

My next thought was to prevent processing the effects. This could be done pretty easily; we'd just need to wrap the `connectivityCheckSaga` and do the following:

- Fork the `connectivityCheckSaga` as usual.
- Poll for changes in the `noAccount` flag (i.e. listen for the action).
- Whenever the flag is true, cancel the fork (see 3.7 Task cancellation · Redux-Saga).
- Whenever the flag is false, fork `connectivityCheckSaga` again.

The last option (if the above doesn't work), would be to setup a middleware that intercepts all `enqueue` actions, checks for the `noAccount` flag, and discards the actions so that they never hit the reducers.

There's an argument to be made that we should do the middleware regardless, to handle the case where a user signs up after they've used the app without an account -- if we only disable the `connectivityCheckSaga`, but the effects are still be queuing, then when a user signs up, all the effects could be dispatched. Theoretically, this would be fine, but I suspect that it'd be better to just migrate all their data over in one go as part of the sign up process.

So... I guess we'll do the middleware route. Leaves all of the existing infrastructure intact while setting us up more easily for the future.

Although, we might still want to disable the `connectivityCheckSaga` just so it doesn't constantly ping the healthcheck route for no reason.

Removing the User Account Settings

Like stated below, the User Account settings will know be a call-to-action to get users to sign up.

What this will actually *look* like... uh... I don't know. The header CTA will just be a simple button, but this has much more room to work with. I guess just a simple message

explaining *why* to sign up would be a good, then just another button to direct them to Sign Up.

Restoring Backups Should Still Work (Mostly)

One of the things about the no-account mode is that users should still be able to create backups (only regular ones, not encrypted ones) so that they can restore the backup later if they clear their browser data/'log out'.

As such, the Restore Backup button should still work, just without the effect half that propagates the data to the Backend.

I don't think there'll actual need to be any modifications for this work as we expect, since preventing the effects from getting queued should be enough.

Although, we will need to make sure that encrypted backups can't be restored.

Disable Encryption Middleware

During app boot, we'll need to disable initializing the encryption middleware. Since this is already controlled by a flag that is passed into the app boot process, we just need to check at the code that *calls* the app boot process (i.e. `watchAppBoot`) whether the user is in no-account mode and act accordingly.

Disable App Boot Data Fetches

Also part of the app boot process, we need to disable the data fetches that happen, since obviously a no-account user has no data.

We can probably just pass in another flag to handle this... Where the flag directly correlates to the `noAccount` flag.

Disable Auth Token Refresh during App Boot

Another thing to *not* do during app boot; see above for how this will be accomplished.

Splitting the Onboarding Sagas into Offline Requests

Since the `finishOnboarding` and `skipOnboarding` sagas both make API calls, they'll need to be converted to offline requests so that the logic is split into commit and effect sagas.

Modifying how api.isAuthenticated Works

Currently, `api.isAuthenticated` checks for the presence of a non-expired JWT to mark whether a user is 'authenticated'.

However, users without an account will *not* have a JWT, for obvious reasons. For more obvious reasons, this is bad because then they'll get kicked out of the app because of the auth redirection logic.

As such, we'll need to modify how this authenticated check works to account for no-account users. In particular, I think wrapping the call into a selector that also checks the `noAccount` flag will be our best course of action.

As for which slice this selector should go under... uhh... `user`? I guess? Yeah, we'll just chuck it under the `user` slice. Although maybe as a cross slice selector? Since, while it doesn't hit two *slices*, it does interact with two different pieces of *state* (one of those pieces just happens to not be in the store).

Warn on Logout

Users need to be warned on logout that all their will be wiped on logout.

This can be handled pretty simply with just another confirmation dialog; nothing special.

Signing Up after No Account

So right now, I'm thinking there'll be a couple ways for a user to sign up for an account while their using the app with no account:

i. Prominent call-to-action in the header on desktop, to the left of the Add button.

ii. In the Settings under "User Account".

- This is a fallback for mobile, since mobile users won't have the first option.

As for how how the actual sign up process is handled, it'll likely end up being very to the encryption migration process (at least, in terms of data propagation):

i. User will click on "Create Account" call-to-action.

ii. They will be re-directed to/presented with the Sign Up form.

iii. The user will enter their new account credentials.

iv. A user account will be created for them.

v. They will be logged into the app.

vi. All of their data will be encrypted and sent to the Backend.

- This is essentially encryption migration process, except without fetching the data first.

vii. Done! They can now go about using the app as they were.

Obviously, once we have Stripe integrated, they'll be able to pick their subscription plan after signing up, but we don't need to worry about that yet.

Hmm, the one thing I think will go wrong here is that the current login flow (i.e. the current login *action*) causes the store to be cleared. Which is bad cause... there goes all their data.

So. We can't exactly just log them in. I almost feel like the signup-after-no-account flow will need its own dedicated saga flow, which just means refactoring out some of the current signup/login logic.

Because here's how it'll have to go:

i. Sign up, account created.

ii. Login, using the API call, not the login saga.

iii. Login to the encryption middleware.

iv. Encrypt and persist the user's data to the Backend.

v. Log them into the app normally so that they finally go through the app boot process.

So it'll be a bit more work, but we gotta make sure to not lose all their precious data.

I think this can be a separate story, since it's not strictly necessary for making the no-account mode work. Not to mention we have the built-in migration method of creating a backup and just restoring it to a new account.

In fact, I had completely forgot about that. Users can just do that to migrate. I mean, having a CTA to signup when not using an account is better for conversion purposes, but from an 'MVP' point-of-view, it's better to take the low tech approach and not bother implementing this yet.

Hence, separate story.

Also, it would be good to give the user a little explanation message that their data will be migrated to their account after they sign up. This message can probably be shown next to the Sign Up form they are shown in-app. Should probably also update the Welcome Step message with a little blurb on

Yes, that's right, there's gonna be an in-app Sign Up form. We should probably only allow users to access it if `noAccount` is set; redirect them back to `/app` if it isn't set. I'm thinking the URL will just be `/app/signup`.

Demo Account

As part of the onboarding process (i.e. the Welcome Step), no-account users should be able to choose to use a demo account where there already exists a set of accounts/transactions. Basically, a more fleshed out version of the `test@test.com` account.

I want this data to be generated each time, since we want the dates to be relative the user's current date. Additionally, by generating each time, we can generate 'random' accounts and transactions.

As such, we'll need to come up with a scheme for generating lots of somewhat realistic transactions (mainly, transaction *descriptions*). Or we just don't and have obviously nonsensical descriptions to add some charm.

I had initially noted that something like `faker.js` could be used for this, but obviously we should avoid that particular library due to its recent issues. Not to mention that it's a *massive* library (1.3 MB uncompressed) with a ton of functionality that we don't need.

So, we'll need to come up with something for this.

I'm thinking we build a list of verbs (e.g. "Bought", "Got", "Transferred" etc) and then a list of nouns (e.g. products) to generate Transaction descriptions.

You know, I think this could be a separate story, since it's not integral to making no-account mode work.

Stories

- I want to use the app without a user account.
- I want to signup and create a user account with my existing no-account data.
- I want to use the app with a demo account.