Story UFC-373: Custom Pagination

The following document will go over the design of custom pagination sizes, as outlined in Story ticket UFC-373.

Acceptance Criteria

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

- I want to specify a page size different from the default 25.
- I want to specify a different page size from a given set of options.
- I don't want a page size option less than 25, since 25 seems like a good minimum.
- I want to set the page size to show all of my items.

Design Brainstorming

So, basically, we just need to modify the Pagination context to support a method for changing the page size, then create a component that modifies the page size, then add this component into the existing pagination footers.

Got it? Got it.

Steps

Modify Context

- Add an action for `setPageSize`.
  - Make sure to account for "all".
- Update `setTotalItems` to account for "all" page size.
- Add `setPageSize` as a function for `dispatch`.
- Write tests.

Update Hooks

- Update `usePaginateObjects` to account for "all" page size.
  - Alternatively, create a selector that encapsulates the logic for deriving the current page slice that internally handles the "all" page size.

Create Component

- Create new component `PaginationPageSize`.

Update Components

- Add `PaginationPageSize` to `PaginationFooter`.

How to handle the "all" page size?

As I see it, there are two main options at our disposal here:

- Use a *really* large number to represent "all", such that no person would reasonably be able to get past it
- Use a "special value" to denote "all"
  - This could be 0 (dangerous, since accidentally doing math with 0 could explode)
  - It could be just the string "all" (meh)
  - Or it could be null or undefined (less meh but still meh)

Obviously, using a giant number is the easiest and simplest solution, since it doesn't require modifying any other code and allows us to keep the `pageSize` as just a number.

But it probably makes more *semantic* sense to have some sort of special value to denote "all".

However, I'm lazy, so I'm just going to decide to use a large number. I'm thinking... 10,000,000 should be sufficient. Why? Cause 1 million seems too small but 100 million seems excessive.

What size options should there be?

I'm thinking 25, 50, 100, 200, all.

That should be sufficient.

I mean, realistically, you either don't care about the pagination size or you want to see everything. So there's *almost* no point in having more options than just "25, all".

How to persist page size between refreshes?

Uh.... local storage, I guess? I don't want to put it in Redux, and I don't want to deal with IndexedDB *outside* of Redux, so... I guess it *has* to go in local storage? I mean, I don't use cookies, so where else would it go?

However, now that means that we need to associate the this piece of stored state with each instance of `usePagination`, since they are separate per call by component.

So... maybe we just shouldn't persist the page size. I mean, other sites don't do that, so... I think we can get away by being lazy and not doing this.

This decision has caused the removal of the following acceptance criteria:

- I want my chosen pagination size to persist between page refreshes (but not necessarily logins).

Component Breakdown

Atoms

N/A

Molecules

- PaginationPageSize

Organisms

- [modification] PaginationFooter

Scenes

N/A

Tasks

- Modify context.
- Update hooks.
- Create new component.
- Update existing components.