Collapsible Sidebar Navigation

Problem

Currently the Project and Notes app have different looking sidebars that accomplish the same functionality. Having a single sidebar across all apps would provide users with a consistent experience across both apps.

Solution

Creating a reusable sidebar for both the Project and Notes app can be accomplished with a single react component. This React component needs to implement the display of the sidebar and maintain the open or closed state across the two sites.

Sidebar display

To display the sidebar, we will stick with the company's design system choice which is Chakra UI components that are themed with the company's branding. We are able to create and style the components much faster with the user of our design system. The specification does not require a mobile friendly view so we will assume a desktop sized screen. The specification calls for some icons, we should prefer the design system icons in `@chakra-ui/icons` but `react-icons` will be used when there is no fitting icon in the design system.

The display of the sidebar should take about 30 minutes.

Maintaining state

There are a few options for maintaining state between different domains. The choices are:

1. Pass the sidebar state in the URL parameter when navigating between sites. While this would maintain the sidebar state for your current session, it would not save across sessions. Given that cross session storage was not an explicit requirement, this is a valid option. To address the cross session issue, each domain could have its own local storage for cross session and the URL query parameter could be used for cross site communication.
2. Store on the server. If we are using server side rendering for our app, this initial state can be added in on the server. For a single page app, this would add an extra request on the first page load that would slow it down.
3. Store in the user json web token (JWT). If our site happens to use JWTs for authentication, we could add an extra field for the sidebar state. This extra data will be passed around with all requests to the server and require issuing a new token every time the user changes the open/closed state. While it would work, it does not seem to be a viable solution.

4. Store with Storage Access API in the browser. One downside of this is that the user may be prompted in some cases to confirm they want cross site sharing. This extra permission pop up from the browser would be confusing for the user and break the experience if they do not consent to the cross site sharing.
5. Cross domain local storage with an embedded iframe. Sadly Safari has this disabled by default so that seems like a dealbreaker unless the company has already marked Safari as an unsupported browser.
6. Related Website Sets. This is unfortunately only supported in chrome with both firefox and safari opposing the standard. It does not seem like a viable solution for this reason.

Considering all of these options, option 1 seems like the clear winner in terms of ease of implementation and covering the vast majority of use cases. While there are some edge cases, they will only be hit in abnormal usage patterns where the expected behavior is unclear anyways.

The cross page navigation with a saved query param should take about 30 minutes.
Adding in saving the state in local storage should take about 30 minutes.
Test setup and comprehensive unit tests will take about 2 hours.

Showing a different list of data based on the URL

This is straightforward as we just need to set up some config on where to fetch the data based on the URL. The use of the term “filter” here is very general and leaves a lot to the imagination. To keep it simple, I will do a very basic demo of this where the filtering is just done to a project list and a note list. It will take about 30 minutes.