Frontend Dev Task

- How would you structure the code?

  >...
  >src
  >assets
  >components
  >hooks
  >pages
  >swapnbridge
  >_tests_
  >index.tsx
  >...
  >dashboard
  >_tests_
  >index.tsx
  >...
  >utils
  >...

  - What system do you use to structure react components in order to ensure reusability of components?
    - Following the single responsibility principle and making sure that any new page/feature implemented follows the proposed layout.
  - How do you ensure rapid implementation of new dashboards, products, etc.?
    - Good structure, clean code and having the basic understanding of the codebase. If components are properly separated such as buttons, navbar, pagination, etc, this will save time when implementing a new feature.
- What styling solution would you choose and why?
  - I'd choose a styling tool that is built on top of tailwindcss such as tailwindui or daisyui.
  - I believe that one of the goals that a frontend engineer team should have is to provide a styled system that is owned by them. I chose tailwind since i think it's an improved version of css which can be used to modify the way that the component looks for the user (since the styled solutions i recommended are built on tailwind). Tailwindui also has the perc that on some of the components that it provides these already work with a behavioral library that is headlessUI. This library takes care of dealing with how things will behave (dropdowns, switches, etc).

- We want the site to be accessible to users speaking various languages. How would you provide translations for them?
    - I'd utilize a library called `i18next`
    - This library allows developers to specify which languages will be supported, add the translation text and on the codebase specify where to translate this based on the translation text previously created.
    - Supports the option to auto detect the default language that the browser has and translate it to that language by default.
- How would you optimize the site for SEO?
    - In case the site is built using CRA I'd suggest that it's worth it to switch to Next.js for a couple of reasons. Being able to indicate which pages will use Server Side Rendering and being able to indicate which pages will be static. Google's crawl bots are not the best when it comes to reading through javascript which is needed if components are getting client side rendered.

- ProcessA
    - When calling setExecuteProcess, the react component compares the current state and the one being returned in the setState function, since they have the same reference it does not re-render the component.
    - On the runtime of the function executeProcess after the array ["Started", "Subprocess1"] is created locally there is no change on the reference when adding the string "Subprocess 2" to the array.
    - By returning a new array everytime that setExecuteProcess is called:

      setExecuteProcess(prevExecuteProcess => [...prevExecuteProcess,"Subprocess x"]

      An array with a new reference is being detected so it re-renders.

- ProcessB
    - When HandleExecution() gets called, the try catch is handed down a context of variables which is the initial state of the process. Inside the execute function the component state is being manipulated, since setState is async and the execution context inside the function is still in the initial state when trying to access the process variable we're going to get the same initial state.
    - This is why we must access the prevState inside the setState function and append the error.