Yohana Interviews Oct 19th-20th 2021

Tuesday
Interviewed with Chen 1st - backend eng, we did the fire sim test

Interviewed with Matt 2nd - eng team lead, we chatted about engineering in general

Wednesday
Interviewing with Peter Zenger 3rd - react & data layer

Interviewing with Alex 4th - react & data layer

He stumped me right out of the gate with the challenge.
I didn't understand what he was asking.
There wasn't much instructions on the coding challenge, and then when he tried to explain it, and show me the code, he realized what he wanted to cover wasn't supported in Coderpad.

I think this challenge was a bit too hard to cover in an interview, and I believe that he hadn't used this challenge in Coderpad, so I'm curious as to what his feedback would be for me.

He gave me a challenge that wasn't supported in Coderpad.
And then we ran out of time.
So went over two of my projects.

Interviewing with Reuben 5th - backend

===

```
const _ = require('lodash');

function sayHello() {
  console.log('Hello, World');
}

_.times(5, sayHello);

/*
We are cogs.com. We have a lot of products that customers can view and
```

eventually purchase.
We'd like to implement a feature where customers can view their recent browsing history.
Let's design this feature.

Requirements ->
 -- Browser history should not include duplicates
 -- Browser history would be displayed back in reverse order in which the customer viewed the items

======

Authentication? - Yes & No
YES - store products in a stack in the profile config
NO - store locally in browser cache
  cache - should store the products in a stack, and return the top of the stack in a component rendered on the DOM,
  DOM - this should be scrollable, renders the top of the stack, but also allows the user to scroll to the bottom of the stack.
  Product - ID, the cache would store the id of the product seen. This capture product function would capture the id, that's being returned from the database, in the initial product call. Piggyback off of the existing API call.

Component - "RecentlyViewed" Container
before the component state updates, we should traverse the stack, looking for duplicates.

Also, this stack traveral function should be saved in the Utilites folder, and imported into the component.

Large List of recently viewed -
- infinite scroll, call for only the first 20, and then before the scroll reaches half way, then make another api call.
Also, have a loading or animated skeleton appear if the api call is taking too long.

How does it maintain it's state as customers are browsing the site?

How are we interacting with the stack over time?
two things:
duplicate product IDs - who decides? back or front?
stack of product IDs.

API what would the object look like that we are posting?

```
profile : {
 username: '',
 userid: '',
 isAuthenticated: true,
 productID: 23432,
}

=======
Component - "ViewedItem" within Container
the viewed item has state, for sure.
The state would either be loaded, or loading
Visually, the component would initially be rendered as a skeleton, and then when the api call returns with the info, then the component state would update and render the new info.



*/
```