Fetch Rewards FE Test

Joel Patel

Nov 8th 2022

Tech used: React.js with Vite build tooling, Tailwind CSS.
Web deployed link: https://joel-patel-fetch-hiring-fe.web.app/
Features:

1. Responsive.
2. Validations on every fields
    - For email: regular expression
    - For password: minimum 6 length, trimmed
    - For fullname: minimum 2 length, trimmed
    - For state: using n/a as default state and validating based on that
    - For occupation: same like state
3. Red outline + text upon shows when input fields are invalid.
    - It'll show on the fields when the value changes and the field is invalid or when form is submitted and fields are invalid, that is default form just press the create me button.
4. Using debouncing to validate the entire form which enables the form to be submitted.
    - Using this so that not a lot of state changes are being requested.
    - NOTE: Other validation functions can be further optimized by only setting the state when the field becomes valid. However, it'll be a few more lines of code and less clean code and more importantly the current version is not throttled/bottlenecked by any means. So I went with the default state setting, where the app will set the state when the field is changed.
      That is from something like this (current),

```
const stateValidationHandler = () => {
  setValidations((prevState) => {
    return { ...prevState, state: stateRef.current.value !== "n/a" };
  });
};
```

to something like this (next page).

```
/**
 * this will only update the state when the current state
 * is false and the field becomes valid when user changes values
 * AND
 * when the current state is true (i.e. it was previously "validated")
 * but user changes the field and made in invalid
 */
let fieldValid = stateRef.current.value !== "n/a";
if (fieldValid && validations.state === false) {
  setValidations((prevState) => {
    return { ...prevState, state: true };
  });
} else if (!fieldValid && validations.state === true) {
  setValidations((prevState) => {
    return { ...prevState, state: false };
  });
}
```

5. Upon successful submission, UI shows a modal with all the properties.

For code reusability and rapid development, I have separated a few UI elements into standalone components so that it can be modified and built upon quickly.

If I had to give this project to fellow developer then other than UI I would suggest them to look at,

1. even more reductions to set state than what I have in validation handlers which will then result in less component re-renders
2. create a standalone select component like the input one I have used for reusability and faster development.

⇒ Thank you recruiters for giving me this chance to work at FetchRewards. I hope I met your expectations in this test. I'm very excited to hear back from you.