

# Step 1 :

I created a logo using Canva and used this site : <https://picsvg.com/> to transfer it to .svg



# Step 2 :

I created two folders: components and pages folder for organization

Create empty templates of the pages that I need, and for the main components that I expect to use (navbar, header and footer).

Start with the navbar component

I used icons from react library and installed the needed libraries

npm install react-icons --save

I used useRef hook that target the dom elements

It does not trigger rerender but preserves the value between rerender

# Step 3:

Created the context.js file. We used the useContext hook.

useContext will give back two components Provider and consumer but we are interested in the provider

provider works as a distributer

I implemented a custom hook useGlobalContext that is just for the sake to not import appContext and useContext each time

to create an appProvider that will wrap my whole app in the index.js and help me in order not to do so much of prop drilling.

I added the navbar functionality in the context.js

And setup the functionality in the navbar component

# Step 4:

In the app.js set up the routing that handles routing between pages

Install the needed dependencies

npm install react-router-dom

<https://reactrouter.com/docs/en/v6/getting-started/tutorial>

I tried to use the react-router-dom given by the instructor but it did not work so I installed react-router-domv6

//Switch which takes the first rout it sees did not work as expected from v5

// this syntax did not work also

//<Route path=”/ ”> <Home/> <Route/>

// the <Routes> is added

Wrap the app component with the <BrowserRouter/> in the index.js

react router set up

# Step 5:

Structure the about us page

It contains the staff component which uses external api (LinkedIn api) just to showcase how to implement the useFetch hook

**second** I implemented the custom hook useslider, which I created in context.js and the call it in the staff.js

2 css properties are important here:

The overflow: The overflow property specifies whether to clip the content or to add scrollbars when the content of an element is too big to fit in the specified area.

The overflow property has the following values:

* hidden - The overflow is clipped, and the rest of the content will be invisible

and the opacity: 0

# Step 6

Same last steps to do the products

Used useSlider hook

The new thing here is that we are using routing and passing parameters in the rout

Which is generating dynamic routing to each product

# Step 7

The contact page has the ReviewForm component which holds the form

In the ReviewForm component I showcase used useReducer hook that combines many values together so I created the reducer and also a modal which displays a massage and change classes between danger and normal according the case.

we have also the allReviews component which displays all reviews. I passed the state as a prop

# Step 8

In the contact us page, we have the ReviewForm.js component. Which uses useReducer hook. Which returns a state and a dispatch function. And takes the reducer function

The reducer function takes the state and the action and returns the state edited according the action

I created the reducer.js file

# Suggested edits

In staff.js I should use data fetched in the useSlider hook as a parameter and not the URL

Add local storage on the reviews to saved from one session to another

Add prop types hook on the employees since it is an external api and some props may be missing