Note on this: The EOS+ frontend applications are currently in a transition state between using “Old React” and using “New React”. A lot of the design patterns we follow are considered “Old” and may not be the most up-to-date way of writing React. We are in the process of transitioning our projects from using older to using newer design patterns, but if any of these seem out of date, that’s why!

Building a new route

1. Create a new folder in `src/routes` named for your view name (ex. “Settings” or “Example”)
2. In general, you will want to create 6 files in this folder:
    a. view.js – Where we build the view. This should only build the view and display the data it’s given. It **should not** be doing any data fetching or state manipulation (see the Container Component design pattern)
    b. <Folder-name>.js (so Settings.js or Example.js) – Where the data fetching, data manipulation, and view state manipulation happens. Takes all the updates to data and passes them to the view to be displayed
    c. index.js – Where we will export our view/route we’ve built in this folder to the rest of the application
    d. container.js – Where we connect our component to all common props. Wraps around the react-redux connect function
    e. requests.js – Where the functions that make requests to get data from the API, and then manipulate that data to be put into a form desirable for our application state, are written
    f. styles.css – Any view styles
3. Then, you want to import the route you’ve just made in `src/config/routes/routes.js`. This will add the route to your application.
4. Lastly, you want to add the route to `src/config/components/Navbar/config.js`. This will add the route to your Navbar.

Now your view is in your application!

Your route and state

For this part, make sure you understand React State, Actions, and Reducers

1. Make a <route-name>.js (ex. example.js) file in src/util/reducers. This reducer should have three basic parts:
    a. `const initialState` – Defines the initial state of this route before any actions have happened to it
    b. `const ACTION_HANDLER` – Handles the actions (basically does any remaining checks and adds them to state)
    c. Your exported reducer function
2. Import and add your reducer to `src/util/reducers/index.js`