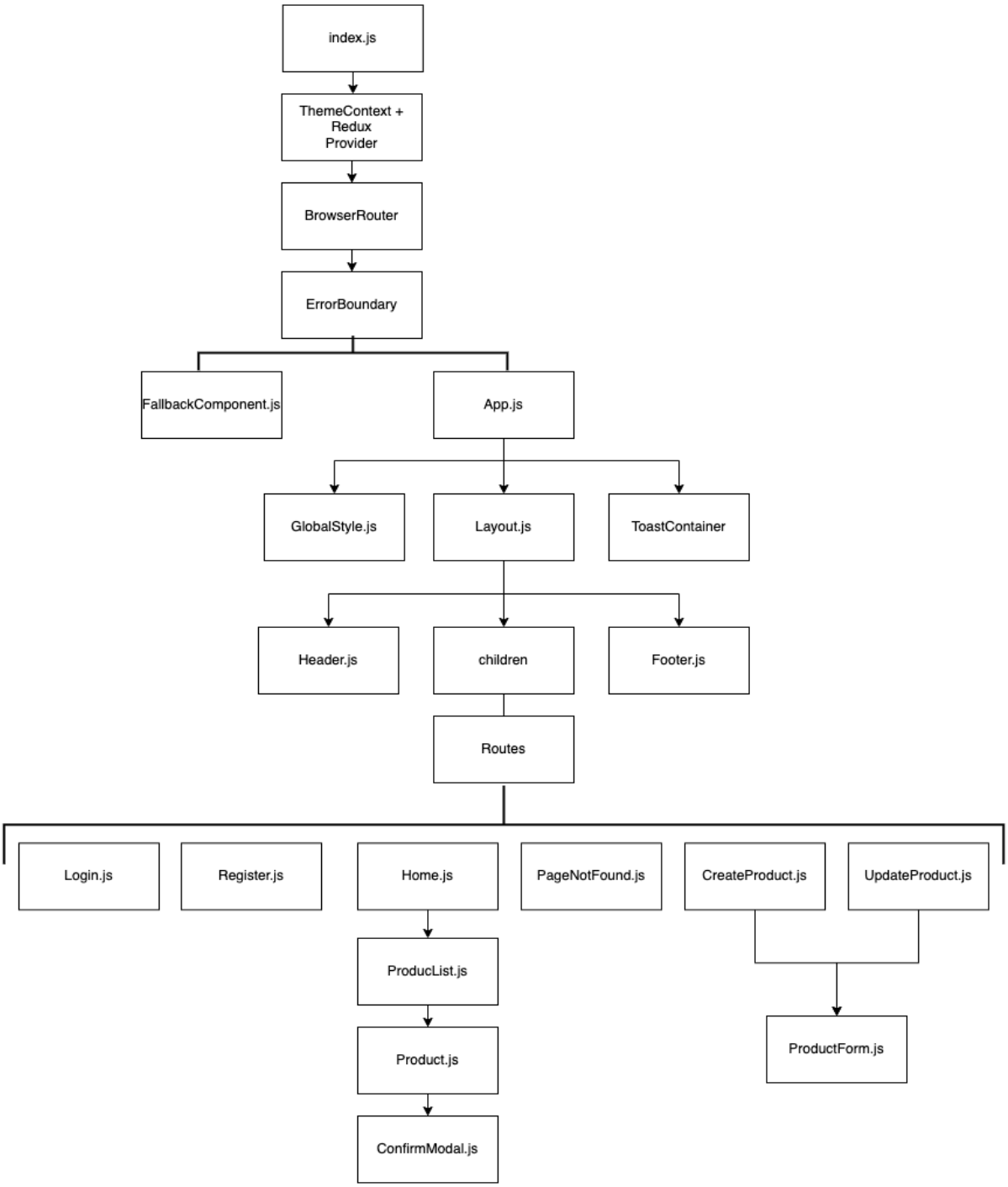
Leo’s final project structure design

Layout flow:



Components specifications:

src/components/common/FallbackComponent

- state : none
- props: {error, resetErrorBoundary}
- logic : display error.message, user click continue button to redirect to /home

src/components/App.js

- state :
    - {theme} = useTheme();
- props: none;
- logic :
    - use useTheme to get theme context then config for GlobalStyle.js for global CSS
    - wrap Layout.js around routes
    - provide routes and protected routes

src/components/common/Layout.js

- state: none
- props: children
- logic : wrap children prop inside a layout made of <Header/> and <Footer/>

src/components/common/Header.js

- state:
    - {theme, toggleTheme}=useTheme();
    - {isLoggedIn} = useAuth();
    - {pathname} = useLocation();
- props: none
- logic :

        - contain Logo
        - contain toggle theme button
        - contain action button:
            - if !isLoggedIn: redirect to Register / Login screen button
            - if isLoggedIn: logout btn + redirect to product list screen button

src/components/common/Footer.js

- state: none
- props: none
- logic: render footer UI

src/components/Login.js

- state:
    - formHelpers = useForm();
    - isLoading = useSelector(state=>state.auth)
- props: none
- logic:
    - render Login UI
    - apply react-hook-form + yup to validate fields
    - dispatch an asyncThunk when form is submitted to request login api and dispatch reducer update

src/components/Register.js

- state:
    - formHelps = useForm();
    - isLoading = useSelector(state=>state.auth)
- props: none
- logic:
    - render Register UI
    - apply react-hook-form + yup to validate fields
    - dispatch an HTTP request for registering when form is submitted

src/components/Home.js

- state:
    - {isLoading,productList,search,sortBy,isFirstLoad,setSortBy,setSearch} = useSortedAndSearchProduct()
- props: none;
- logic:
    - render Search Bar UI + ProductList.js component
    - if search === “”, show a list of all Products fetched from server
    - if search !== “”, show a list of Product given by search Api
    - has create a new product btn
    - Has sort by filter to sort product list

src/components/AddProduct.js

- state:
    - {isLoading} = useSelector(state=>state.product)
- props: none
- logic:
    - create and pass a function down to a UI component, this function takes the product from a form and makes an HTTP request

src/components/UpdateProduct.js

- state:
    - {isLoading,byIds} = useSelector(state=>state.product)
    - {productId} = useParams()
- props: none
- logic:
    - create and pass a function down to a UI component, this function takes the product from a form and makes an HTTP request

src/components/Common/ProductForm.js

- state:

        - formHelpers = useForm();
- props:
    - product = {}
    - handleFormSubmit
- logic:
    - render Product Form UI
    - apply react-hook-form + yup to validate fields

src/components/ProductList/index.js

- state: **note sure what need to apply pagination**
- props:
    - productList=[]
- logic:
    - render a list of Products using Product.js UI

src/components/ProductList/Product.js

- state: none
- props:
    - product
- logic:
    - render Product information
    - has Edit btn + Delete btn

src/context/themeContext.js: useTheme

- Get theme context from useContext(themeContext)
- Detect wether the context in called inside its provider

src/hook/useSortedAndSearchedProducts.js

- Separate the logic of fetching products out of the caller component;

Context:

src/context**/themeContext.js**

State:

- [theme, setTheme] = useReducer(reducer,”light”)
- theme: dark/light

Provider:

- value={theme, toggleTheme}

Redux:

src/store/productsSlice.js

Name

- products

initialState: {byIds:{},ids:[],isLoading:false,isFirstLoad:true}

- byIds {}
  - id
  - imageUrl
  - title
  - price
  - createdTimestamp
- Ids : []
- isLoading
- isFirstLoad

Reducers

- setIsLoading
  - Set isLoading state
- resetProductState
  - Reset product state to initial state

Thunk

- getProductList
- getProduct

- addProduct
- updateProduct
- deleteProduct

src/store/authSlice.js

Name

- auth

InitialState: {isLoggedIn: false, userName: ‘’,isLoading:false}

Reducers

- setIsLoading
  - Set isLoading state
- logout
  - Clear isLoggedIn and userName and remove token from localStorage
- resetAuthState
  - Reset auth state to initial state

Url specifications:

- Login
  - **/login**
- register
  - **/register**
- redirect to /home
  - **/**
- home
  - **/home**
- view a product
  - **/product/:productID**
- add a product
  - **/product/create**
- NotFound
  - **/***

Backend APIs:

- Authentication APIs:
  - register:

    POST /login

  - login:

    POST /register

- Products APIs:
  - Get sorted products:

    GET /products?&search=...&filter=...

  - get a product:

    GET /product/:id

  - add product:

    POST /product

  - update a product:

    PUT /product/:id

  - delete a product:

    DELETE /product/:id