**Frontend technologies used in our case**2021-11-27 version by JV

*SPA =* ***Single-Page Application****, Only one web page downloaded from the Web Server, but then with JS & AJAX that single page’s DOM updated constantly. Showing/hiding certain Views so that it looks like we would have several Pages*

**React**: SPA component library

* renders the output page (HTML+CSS+JS) DOM from 1) your Page template code (React Components inside React components) AND 2) the data provide by AJAX

**react-dom:** How react interacts with the browser DOM. Mappings, rendering etc.

**shopify/Polaris**: Shopify's React components for creating the admin plugins

**shopify/app-bridge,** **shopify/app-bridge-react:** These together allow e.g. Shopify authentication features in React app = in the plugin  
<https://www.npmjs.com/package/@shopify/app-bridge>   
https://www.npmjs.com/package/@shopify/app-bridge-react

**react-apollo,** Enables GraphQL queries done by the React components

**apollo-boost**: Enables GraphQL queries

**graphql**: The JavaScript reference implementation for GraphQL, a query language for APIs created by Facebook.

**Koa, koa-router, koa-session**: Koa is a replacement for Express in the backend API creation and routing. Just because our Shopify-plugins are both React frontend apps and can also act as backends. Right?

**shopify/koa-shopify-auth**: Middleware to authenticate a Koa application with Shopify = Shopify working version of the Koa authentication.  
<https://www.npmjs.com/search?q=shopify%2Fkoa-shopify-auth>

**Not used or covered this time – Thus most are grayed out!**

**react router**: SPA front-end routing between the Views (~like “going” to different page)

* though really just showing and hiding React Views
* we can also send parameter data while going to another View, e.g. id:s

**Redux:** Front-end side (=in browser memory) State management.

* part of the data model temporarily kept in browser memory (in Redux store/state)
* update frequency depends on the nature of the data (colors maybe not before next login, messages maybe polled/refreshed a lot more often)
* many components can share the same data

**React-Redux:** React components will be data- and event-bound to Redux store.

* actions dispatched to Redux code.
* redux store state bound to the React state of the components

**React-Material-UI:** Material-UI styled React components that share common theme and styles

* Those React components need to also be 1) Redux-connected and 2) React-routed!