DESIGN DOCUMENT FOR TOP TEN ALBUMS PROJECT

What technologies and libraries I used and why ?

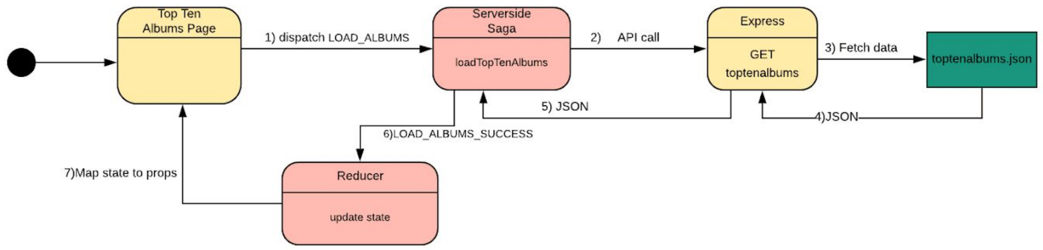
1. **React** as the framework on the UI side
2. **Next js** for server side rendering. I chose Nextjs as the framework because it provides server side rendering and is widely used by many large companies like Netflix, Uber Marketplace, Audible, etc. So it has been widely tested and there is a lot of community support. Some of the important features in Nextjs are
    - Hot Code Reloading
    - Automatic Routing
    - Server Rendering
    - Automatic Code Splitting
    - Single File Components for scoping styles
    -
3. **spotify-web-api-node** library to fetch data from Spotify. This also gave me the feature where I could get data from Spotify in batches. This wasteful for pagination. I started of with *node-spotify-api* library but found that it did not have the search functionality
4. **custom server** I used express server in the backend because it gave me more flexibility in how to cache the initial access token
5. Used nextjs routing with **nextjs/link** and **nextjs/router**
6. Libraries used for client side
    - Redux, redux-saga, next-redux-wrapper, next-redux-saga for state management
    - Isomorphic-unfetch for fetching data
    - React-paginate for pagination.
    - React-sortable-hoc for reordering top ten albums table. This module has touch support and accessibility support
    - Material-ui for UI components. This provides responsive design and touch support
    - Fortawesome for icons
7. Libraries used for server side
    - express and express router
    - node-cache to cache the access token for spotify
8. Utility libraries

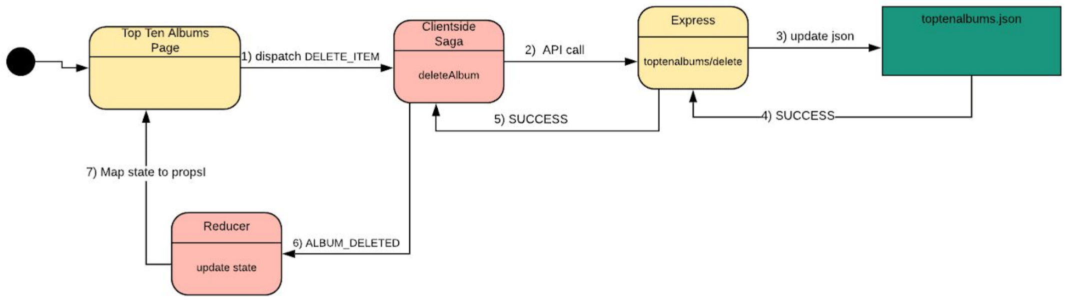
- body-parser, deep-equal, array-move, etc

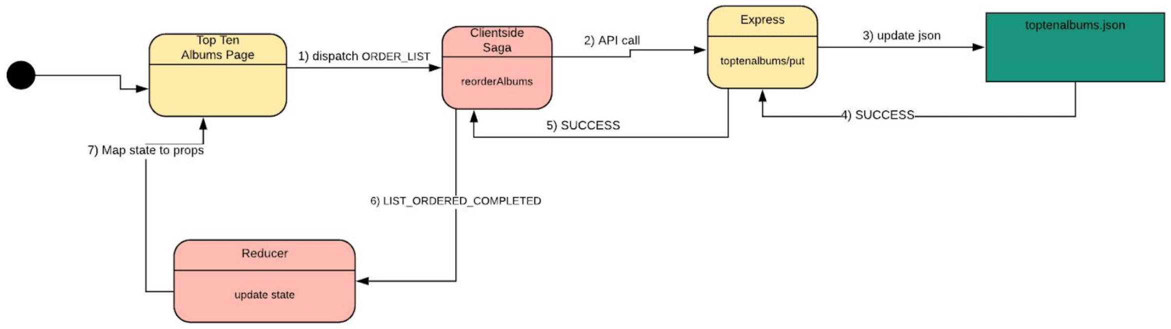
9. Testing libraries
    - jest
    - react-test-renderer
    - Enzyme, @testing-library/jest-dom", "@testing-library/react" are useful. But did not have time to write tests using those.
10. Static code analysis
    - Eslint
    - @next/bundle-analyzer
11. Formatting code
    - Prettier

Some UML diagrams to explain the project flow:

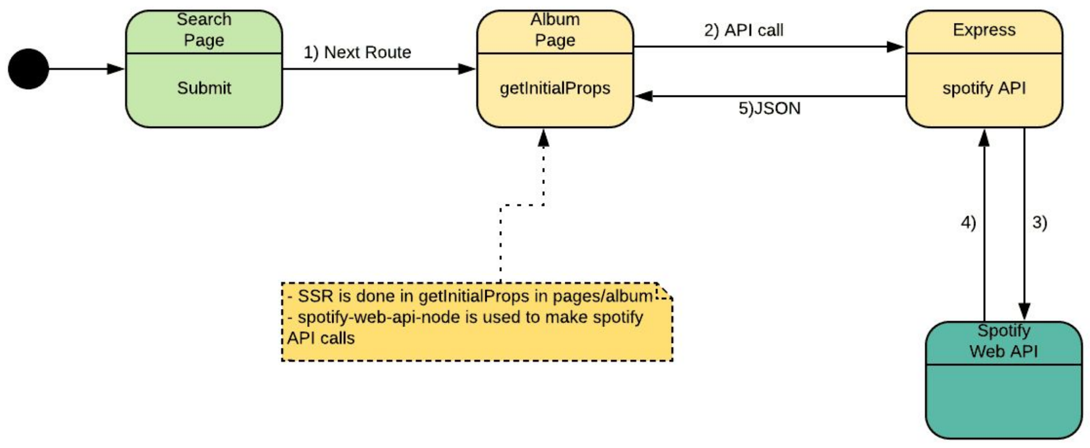
Top Ten albums flow:







Search Albums flow:



Requirements for the project and what I could complete

1. **A dark/light mode feature.** I did not have time to do this. This can be done using context as explained here- https://reactjs.org/docs/context.html
2. **Accessible to users with disabilities.** Tried my best to implement this. Followed principles from here
    a. https://www.w3.org/WAI/GL/WCAG20/
    b. https://reactjs.org/docs/accessibility.html

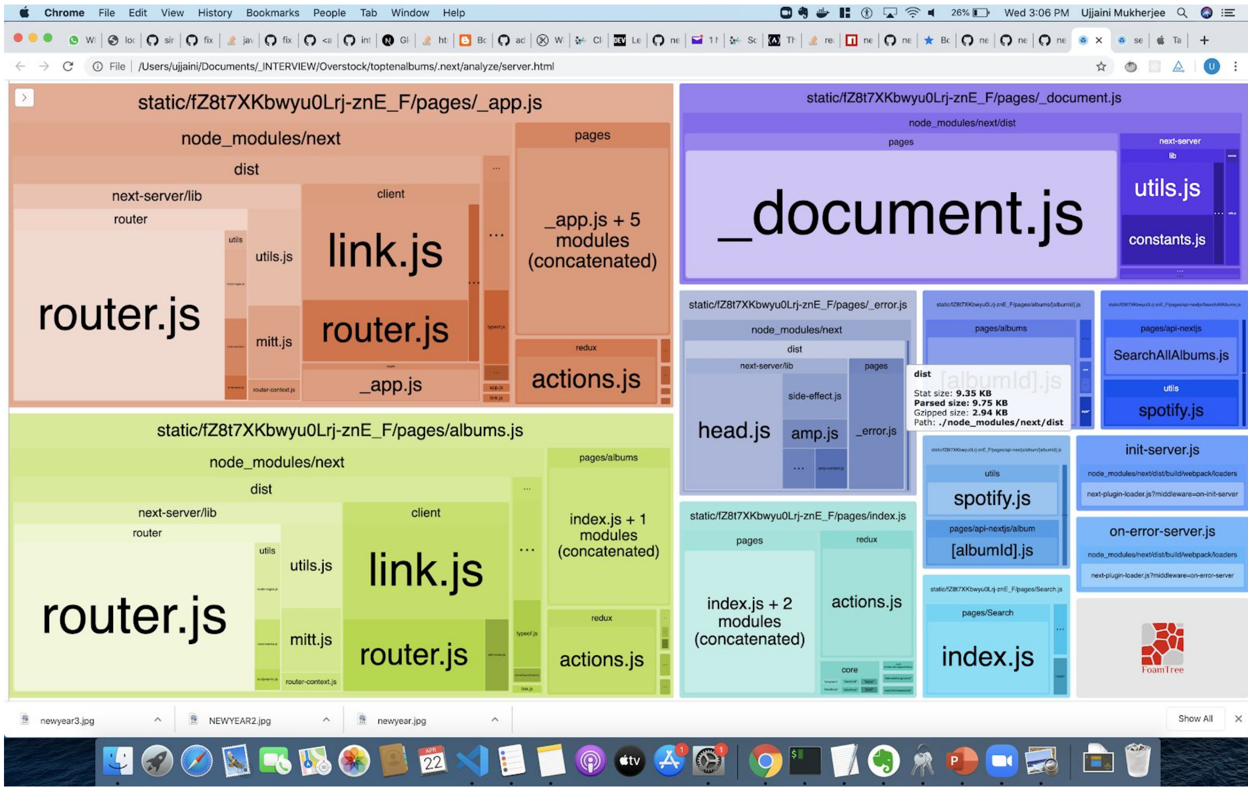
    Followed principles like providing alt text for images, using semantically correct HTML and use of aria-label. Testing may be done using **Voiceover** on Safari browsers
3. **Performant**

   a. React is performant because of concept of shadow DOM
   b. Nextjs is performant because of server side rendering, code-splitting and other optimizations
   c. I used functional components as far as possible in react to reduce overhead of class components
   d. Avoided unnecessary DOM items and redrawing.
   e. Did strict static code analysis using eslint to avoid unnecessary imports or unused variables
   f. Used node-cache on the backend to store the access-token for spotify API so that we do not have to fetch it every time

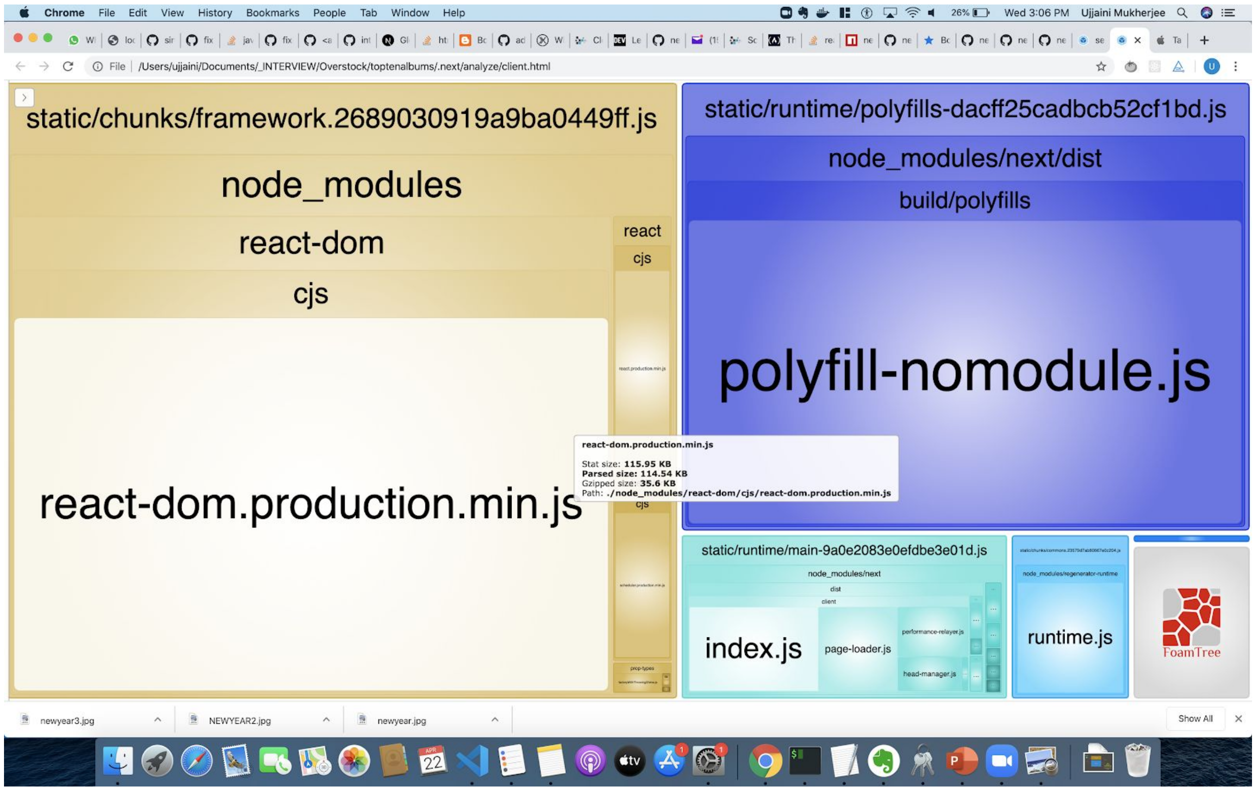
4. **Responsive design**
   a. Used material-ui components (eg. grid) which are responsive
   b. Used CSS feature flex which gave responsive design
5. **Touch features.**
   a. Used react-sortable-hoc and material-ui libraries which work with touch
6. **Config/env store.**
   a. Added an .env store to store the client id and secret for spotify
7. **Tests**
   a. Created a testing and coverage framework with jest. Tests can be run with ‘npm run test’ and coverage results may be created using ‘npm run test:coverage’
   b. Added a single snapshot test under __tests__ folder. Did not have time to write more tests.
   c. The test coverage should at least be 80%, ideally 100%
8. **Build system**
   a. Used docker to automate the build system
9. **Static code analysis**
   a. Used eslint

b. Used @next/bundle-analyzer. (Currently not working because of config issues in next.config.js)
Some examples of bundle analysis images are below:

Client analysis



Server analysis


/Users/ujjaini/Documents/_INTERVIEW/Overstock/toptenalbums/.next/analyze/client.html
static/chunks/framework.2689030919a9ba0449ff.js
node_modules
react-dom
cjs
react-dom.production.min.js
react
cjs
react.production.min.js
scheduler.production.min.js
static/runtime/polyfills-dacff25cadbcb52cf1bd.js
node_modules/next/dist
build/polyfills
polyfill-nomodule.js
static/runtime/main-9a0e2083e0efdbe3e01d.js
node_modules/next
dist
client
index.js
page-loader.js
performance-relayer.js
head-manager.js
node_modules/regenerator-runtime
runtime.js
FoamTree

```
react-dom.production.min.js

Stat size: 115.95 KB
Parsed size: 114.54 KB
Gzipped size: 35.6 KB
Path: ./node_modules/react-dom/cjs/react-dom.production.min.js
```

10. **Routing**
    a. Used nextjs routing to show various pages
11. **Clean code**
    a. Used eslint to point out errors and prettier to format the code

Improvements which may be made on the existing project

1. User login and authentication
2. Store album details in a database like Mongoldb
3. Prefetch the albums data and cache it so that it can be stored ahead before user makes the next page selection
4. Use of node-sass for CSS preprocessing