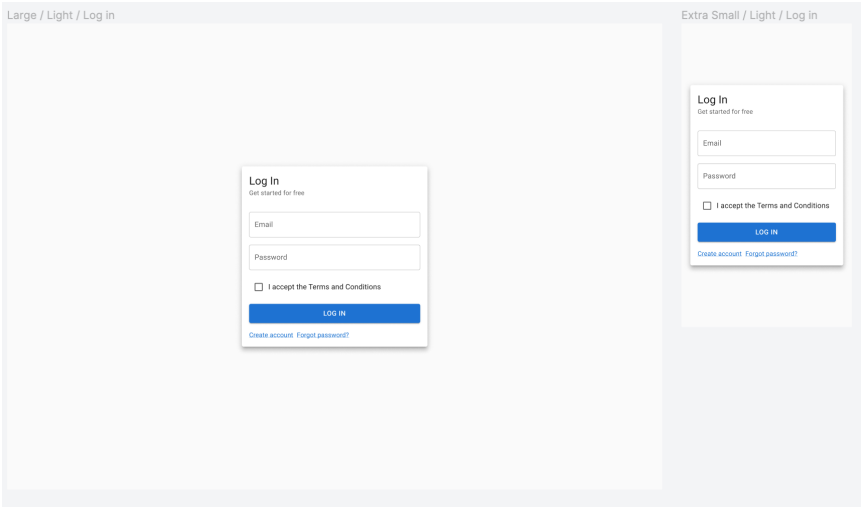
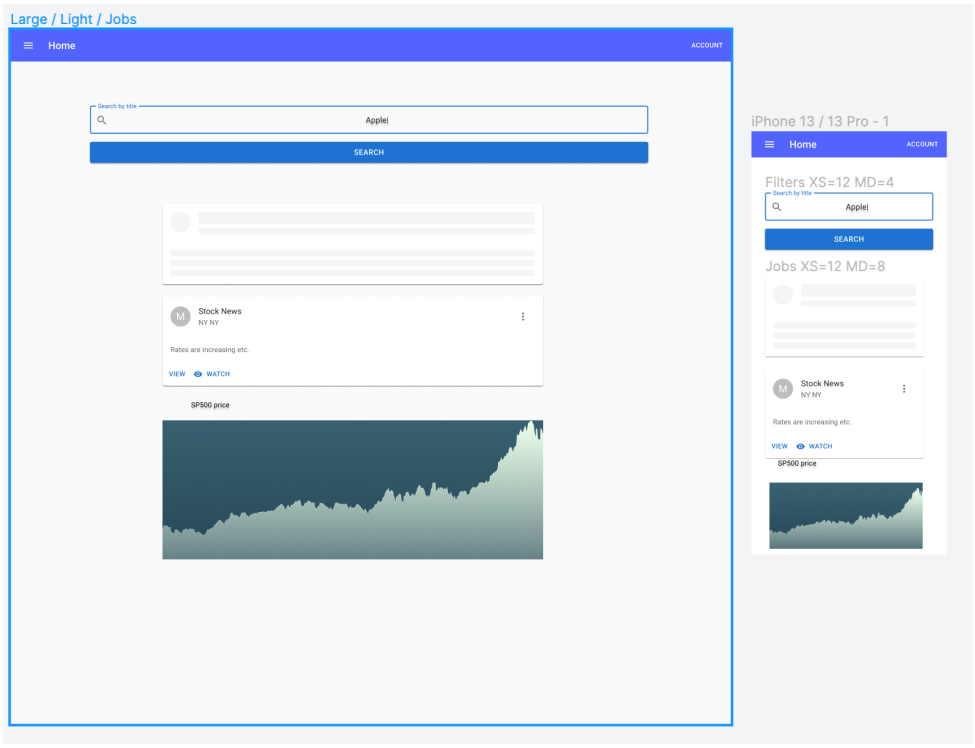
CPSC 458 Final Design Document

- **Why are you building this?**
    - To utilize open source stock market API (Finnhub) data to make a custom, dynamic, and interactive stock wiki web application. Users will be able easily search for a stock of interest and have access to helpful information (provided from the API) such as the company's background, valuation metrics, etc. The API data will utilize the useSWR hook to periodically update the stock data that the user sees. Also, an implementation of Google Maps API will show where a company is located and make a bubble according to the size. In order for users to use the application, they will have to create an account which means authentication and a backend database will be features of the application.
- **What will a user gain from using your site?**
    - The user shall gain basic essential knowledge about a company and its stocks, as well as insightful, up-to-date metrics about the stocks. On top of that, additional geospatial information will be learned.
- **Will you persist user-generated data in local storage, a database, or something else like a CMS? Why is this the right solution for your project?**
    - The application shall persist user data, related to user login and authentication, in a database. We will use a SQL database, supabase, for this information because user data is very important to keep secure, and databases offer the best security in terms of data persistence. We will not use a CMS because we do not require a GUI for end-users to upload / manage content and data of the web application. Oftentimes CMSs utilize RDBMS to persist its data, and they are mainly used when a content manager needs to regularly update or manage the content of a web application. Since we do not need that functionality, it is better to use a RDBMS which will allow the opportunity to customize the database configuration and settings. We will specifically use supabase instead of a local SQL DB or different cloud provider because of how convenient it is to integrate supabase into a Next/React.js application.
- **What accessibility concerns do you anticipate and how will you implement a solution?**
    - Blind or impaired vision users
        - Make sure all content is reachable by the screen reader.
        - Make sure to include appropriate html attributes such as alt, label, htmlFor, etc in html tags.
            - Describing trend lines of stock prices.
        - Make sure all interactions or buttons are reachable by tabbing on the keyboard.
        - Make sure our frontend UI passes the Chrome color contrast test.
    - Login
    - Signup
    - Home (Search page)
    - Stock page (dynamic to each stock)

- **Include a list of each individual component your web app requires, its function, and what each component will take in as props.**
    - **Include a detailed mockup of each page of your web app, for both mobile and desktop screens.**
    - <u>Pages</u>
        - Login
            - Default home page, if the user is not signed in via cache.
            - Where existing users will sign into their application.
            - If the user does not have an account, can click on the sign up link.
            - Error Handling
                - Disabled submit button if an input field is empty.
                - User does not exist message upon failed login.
                - Password is incorrect upon failed login.
        - Signup
            - Where new users can sign up for an account with a unique username and unique email.
            - Error Handling
                - Disabled submit button if an input field is empty.
                - Username already exists upon failed signup.
                - Email is already in use upon failed signup.

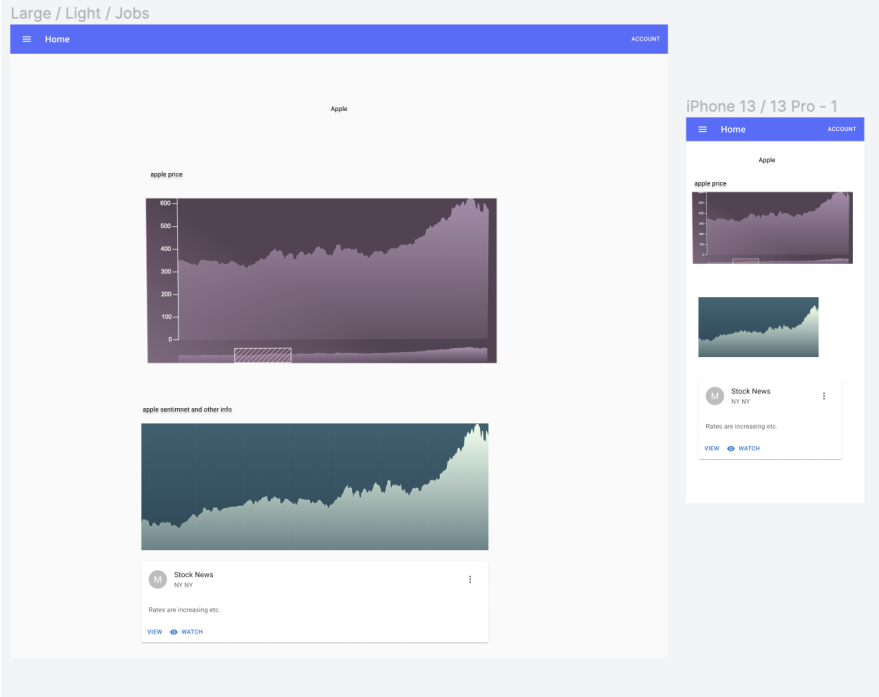


        - Home (AKA Search Page)
            - Where users can search for stocks via the search bar. The results will be populated dynamically in real time under the search bar as the user is typing. It will utilize smart search via regex.
            - Error Handling
                - No results found message when there are no results based on user search.

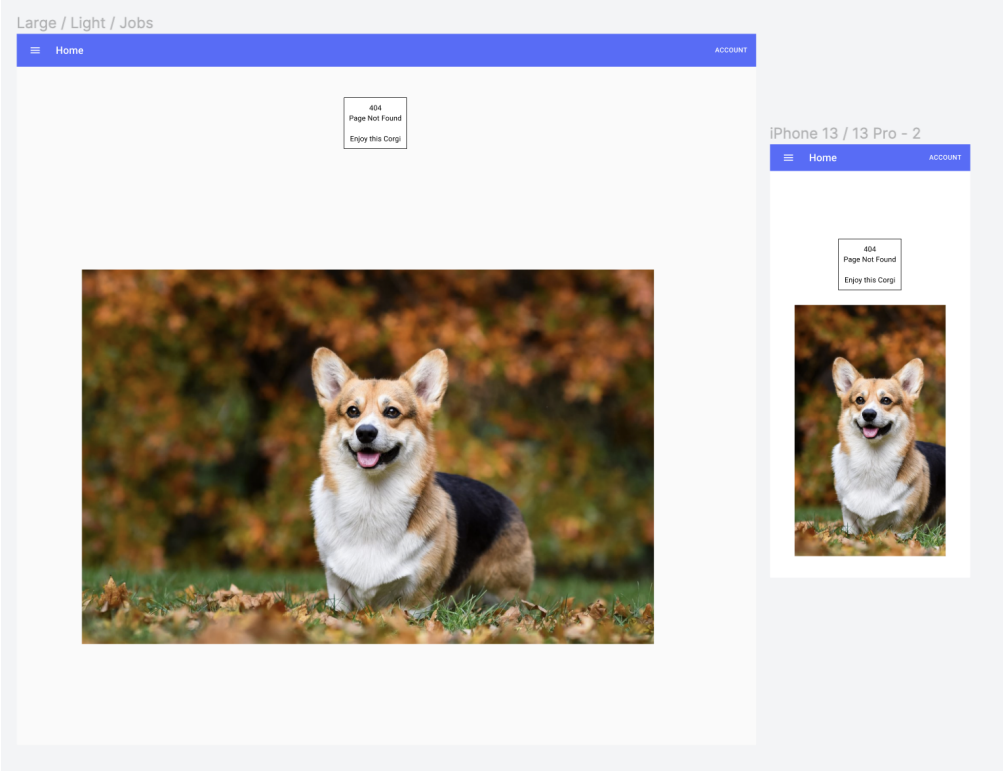


- Stock page (dynamic to each stock)
  - The user is redirected here after clicking on one of the Stock card components from their search results.
  - Interactive map (google maps API) to show location of headquarters (Google Maps React API)
    - Map of any information gleaned from the stock data. At the very least, it will show the headquarters and a size for how large it is (employees or market cap).
  - The page will contain the following info:
    - Company name and stock name/symbol.
    - Description of what the company does.
    - Current evaluation.
    - Stock type.
    - Currency type and country of origin.
    - Web url for the company.
    - IPO date

- Employee total.



- NotFound
  - When a user types in a URL that does not exist, they should be redirected to this page that contains the 404 error message. Unable to make the text bigger on this.



- Components (General → Specific)
    TODO: Add descriptions
    - Container
        - NavBar
        - Footer
    - Button
    - Modal
    - Card
        - Information about a given stock from the stock API
        - Basic financials
            - https://finnhub.io/docs/api/company-basic-financials
        -
    - SearchBar
        - Company name or ticker symbol. Could get these out of a list from the Stocks API (finnhub)
        - https://finnhub.io/docs/api/symbol-search
    - VisXGraph
        - Brush chart for stock price
            - Going back a year or two and have the main window be the past 3 months (90 days)
            - https://airbnb.io/visx/brush
            - https://finnhub.io/docs/api/quote
        - Area Closed chart for sentiment
            - https://airbnb.io/visx/areas
            - https://finnhub.io/docs/api/social-sentiment
    - GoogleMapsObject
        - Implements google maps API
            - https://developers.google.com/maps/documentation/javascript/importing_data
            - https://developers.google.com/maps/documentation/javascript/earthquakes