**Home.vue-**

1. This is the homepage of the website (the landing page).
2. Class= “home-box” includes the whole html file which contains what the user sees.
3. Class= “banner” contains the search bar for searching stocks and the moving images such as reddit, YouTube etc.
4. The following variables are created when the home page appeared

a. searchQuery: '',

      stocks: [],

      selectedStock: [],

      filteredStocks: [],

      showDropdown: false

b. the website section is created with 3 clickable cards-

1. Sentiment score- redirected to "/insights?tab=sentimentAnalysis"

2. ML modelling- redirected to "/insights?tab=relativeValuation"

3. FCFF Valuation- redirected to "/insights?tab=modelling"

This is created inside the class=”website-box” , when any of them is clicked, the toPage(path) method is used which will be explained below.

1. Inside the class= “search-bar”,
2. the v model= “searchQuery” exists which links whatever the user writes inside the search bar to the variable searchQuery.
3. @input="debounceFilter" runs a function to filter matching stocks as you type.
4. @focus and @blur show/hide the dropdown list.
5. Dropdown List: When typing a stock name, matching names show up using filteredStock variable.
6. The following are the methods used on this page-
7. As soon as the page is opened, the mounted method is called which calls the fetchStockList() method.
8. fetchStockList()- 1. Creates a constant “response” which calls the API /stocks from which the whole json body is stored inside the response.

2. This.stocks= response.data/stocks stores all the stock names inside the variable stocks.

1. debounceFilter()- 1. Runs a function to filter matching stocks as you type.
2. Prevents the filter from running *too often* while typing. (Waits 300ms after the last keystroke before filtering).
3. Also calls another method called filterStocks().
4. filterStocks()- 1. trims the searchQuery which the user typed and then filters all the stocks inside the variable stocks which starts with searchQuery and stores them inside the variable filteredStocks.
5. selectStock()- when user finally selects a stock, this method is invoked and it does the following-
6. request API /stocks to get stock details and is stored inside the session storage and also hides the dropdown menu.
7. encodedStock= encodeURIComponent(stock) is used so that the URL is successfully sent to the API to collect the stock details because symbols like ‘ are not used in URLs.
8. Finds similar stocks inside the subindustry using API call /stocks/subindustry/${encodedStock}, Fetches details for each of those similar stocks and this is stored inside the session storage too.
9. Then the user is finally navigated to the insights page where the whole stock page opens and this is done via the following command-

this.$router.push(`/insights?tab=modelling`);

1. hideDropdownWithDelay()- hides the dropdown menu when user click outside the input box
2. toPage(path)- Generic function to go to a new page, used when user clicks one of the three cards (Sentiment Score, ML Model, etc.).

**Login.vue-**

1. This is the login page for the user
2. Class= “wrap” contains the whole template which is shown to the user.
3. Username (v-model="username"), Password (v-model="password", hides characters), a login button and a link to register which redirects to register.vue
4. Contains three variables since the start, which are username: "",password: "", loading: false, loading (Used to show a loading spinner on the button).
5. Contains 2 methods-
6. Login()-
7. If already loggin in, return
8. Make the loading variable true
9. Calls the \_\_check() method to see if the fields are empty or not.
10. It calls the API /login which checks the username and password.
11. If successful, then the user information is stored inside the local storage and Call Vuex’s setUserInfo() to update user data.
12. Finally, its redirected to the home page which user profile visible.
13. \_\_check()- to check if the given fields are not empty, error message is shown otherwise.

NOTE\* for now the user when loggin in is stored inside the local storage.

**Register.vue-**

1. A full-page box with 4 input fields: First name, Last name, Username, Password, register button, a link to go back to the login page. NOTE\* All form inputs are bound using v-model, so Vue keeps the data live as you type.
2. Variables of the same fields are created plus the loading variable which is created to be false at first, this also prevents duplicate submissions.
3. Two methods-
4. Register()-
5. Runs when the user clicks the button or presses Enter in the password field.Prevent duplicate submission if loading is true.
6. Uses the \_\_check() method like the one used in login.vue and then calls the API /register which stores all the data inside the database.
7. On success a message is shown and the user is redirected the login page.
8. \_\_check()- same as before.

**Profile.vue-**

1. This is the user’s profile page on the top right corner to be clicked.
2. A greeting with the user’s name and avatar (from Vuex userInfo).
3. A list of favorited stock names (retrieved from backend) using /user-favorites. Clicking any stock name from the favourites redirects to its details page i.e. using a v-for loop and links them to the insights page via query parameters.
4. Variable favoriteStocks: [] This will hold the **list of favorite stocks** from the backend.
5. Class= “plan-card” displays the current plan and provides a button to navigate to the pricing page.
6. The following variables are created inside the user variable- first\_name, last\_name, username, plan. The default plan is set to ‘Free’, but it won’t stay that if backend provides a different plan using the axios user/username. The statement this.user = res.data; will replace the default value with the real plan.
7. Then there is a logout button which calls the method logout() defined below.
8. Mounted method is called which Checks if the user is logged in (using localStorage). if not logged in, it redirects to the login page.
9. 2 API requests are made-
10. /user/username which fetches first name, last name, and username from the backend. If the request fails (e.g. token expired or user doesn't exist), it logs out and redirects to login.
11. /user-favorites/username which fetches a list of favorite stocks, if nothing is added to favorites, an empty list is shown.
12. The following methods are used-
13. logout() which Clears user info from local storage, redirects to the login page and reloads the page to refresh the app.
14. goToPriciing() method redirects to the pricing.vue page. Calls a custom event “forceActiveUpdate” to update UI state like active tab in header.

**404.vue-**

Used when a path is not directed to any page, then this page appears which is just images and text showing that “the page you are looking for does not exist”.

**Verify.vue-**

Ask for a verification code for a device who is using the localhost for the first time.

**Pricing.vue-**

1. This page is if the user wants to see/upgrade the plan. It presents different subscription plans offered by the platform. Plans- Free, Pro, Enterprise.
2. The el-container class= “app-layout” and el-main gives a consistent structure and styling across the page.
3. Class= “plans” consists of three plans as mentioned above.
4. Includes the <el-backtop> component which lets user back to the top of the pricing section.