COMP3900 - Computer Science Project

Stock Portfolio Management System - W18A-group-group

Name	Email	Student ID	Role

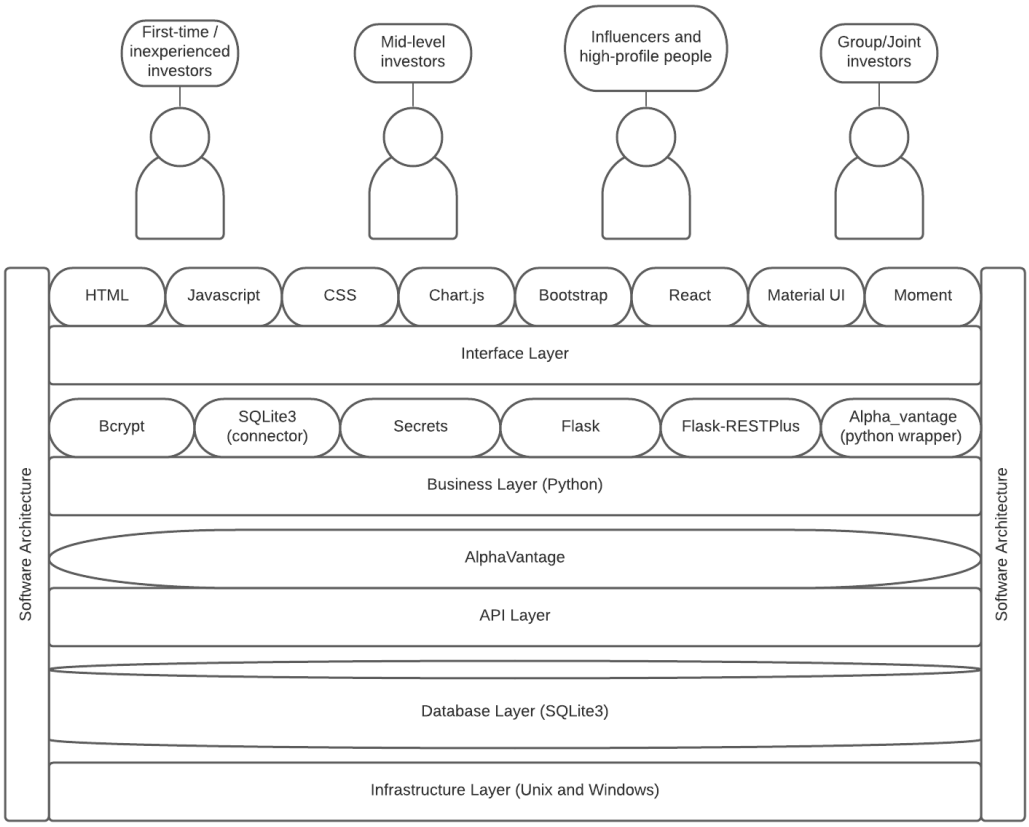
Proposal Submission Date: 18th October 2021

Table of contents

Overview

Architecture/design of the overall system and functionalities.

The architecture and design of the overall systems and functionalities have been updated from the proposal. Below is the updated software architecture diagram.



The main additions in the frontend are Moment and Material UI. The Moment library was used to parse date and time for stocks, and Material UI provided additional resources for styling the frontend. The main additions to the backend were Flask-RESTPlus, a Flask extension for building REST APIs, as well as the alpha_vantage python wrapper, which wraps around the official Alpha Vantage API. More detailed information for each third-party functionality we used can be found further on in this report.

In terms of project layout/structure, the code has been split between two different folders: `stock-portfolio` (frontend) and `stock-portfolio-backend` (backend). Within each folder, the components of each system are further split up into their respective folders, leading to an organised and structured repository.

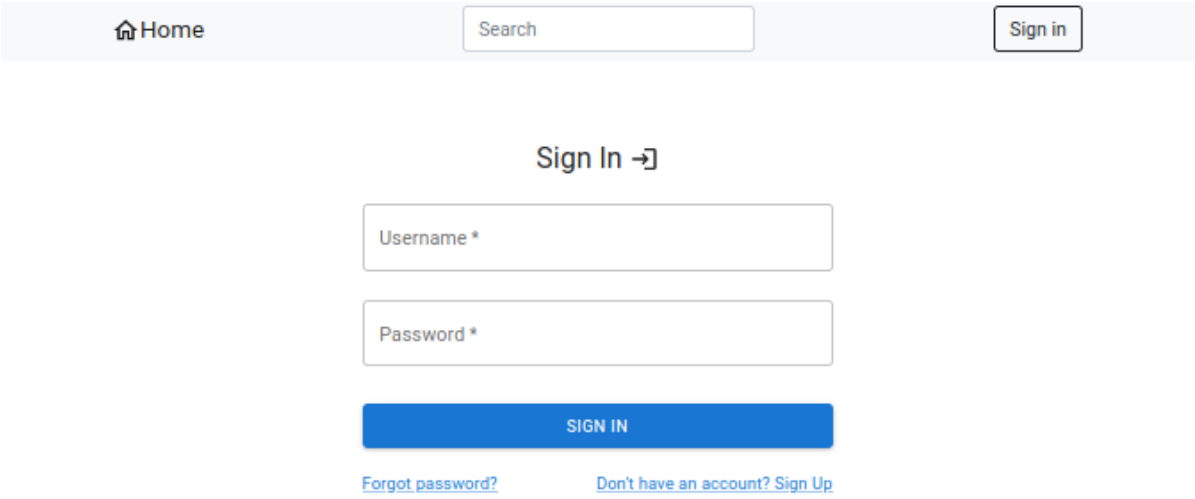
Overall, these work together to provide a functioning system for the management of stock portfolios.

Functionalities developed by us

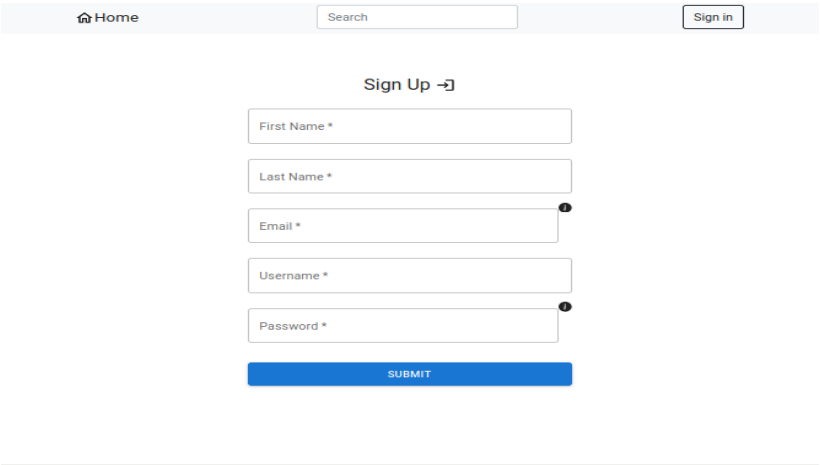
Descriptions of functionalities

Account Creation and Management

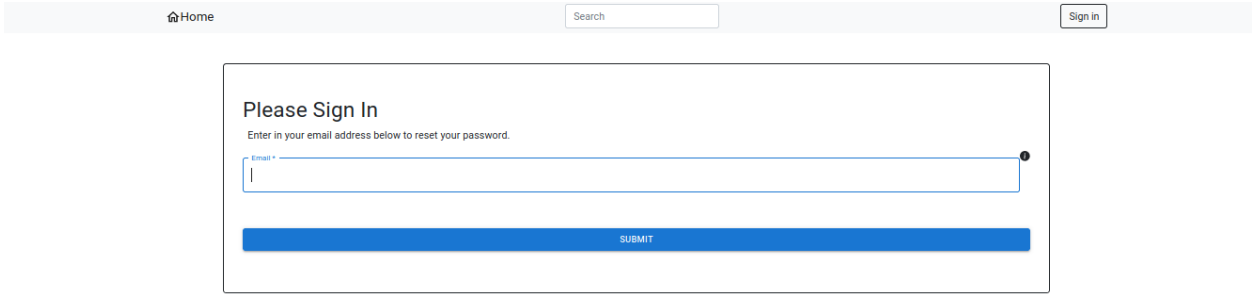
In the Sign-Up Page, users can create an account by providing their first name, last name, email, username and password and using the username and password to sign in to their account. Users will be able to view, first name, last name and email in the Account Details Page. Users also can retrieve their account by resetting their password on the Sign In Page by clicking on the link sent to their sign up email. Also, users are about to delete their account on the Account page.



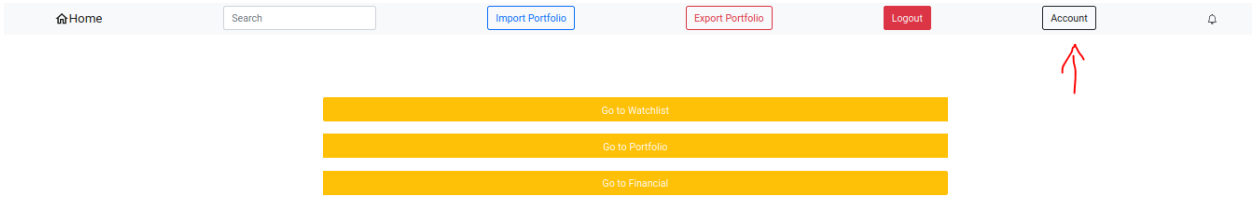
Here we allow the user to sign up.



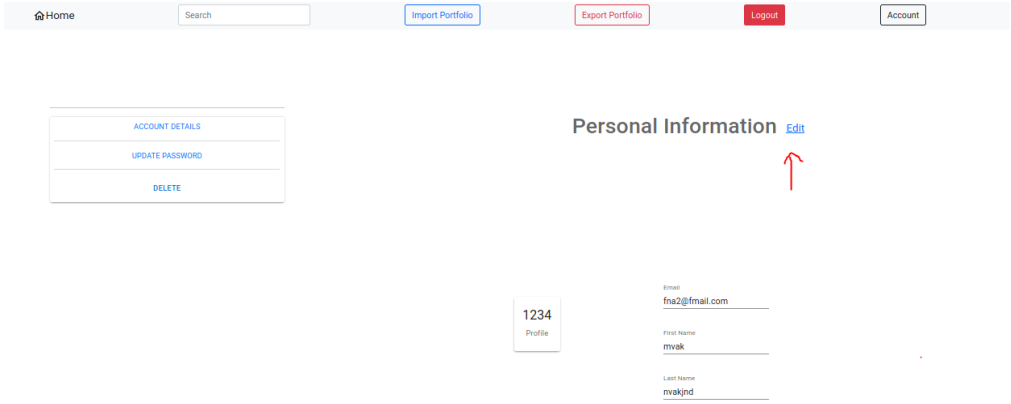
In this image, there is the option to reset the password if a user forgets it.



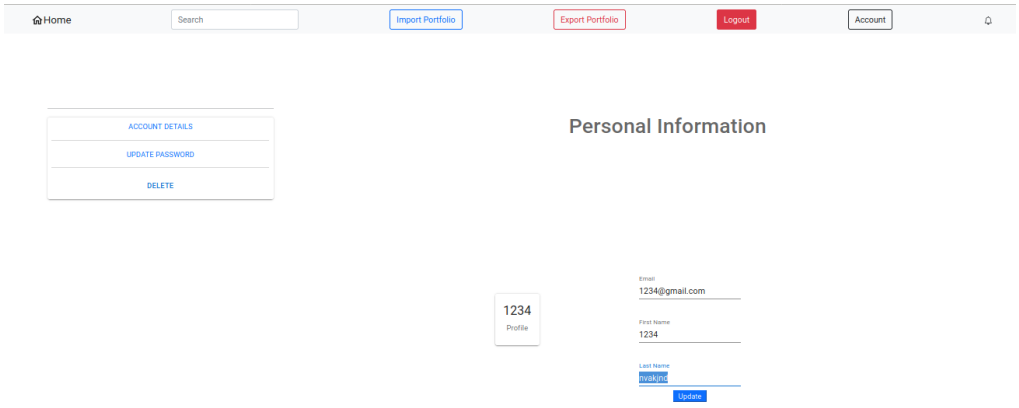
The Account tab will allow you to view your account profile.



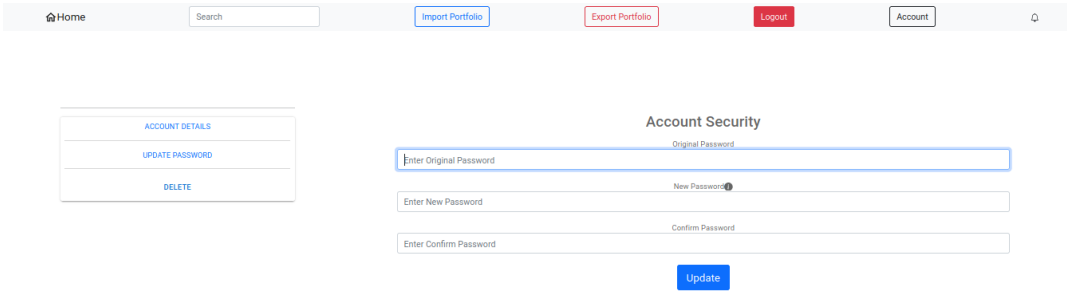
On the account details page, we allow a user to edit their details with the edit button.



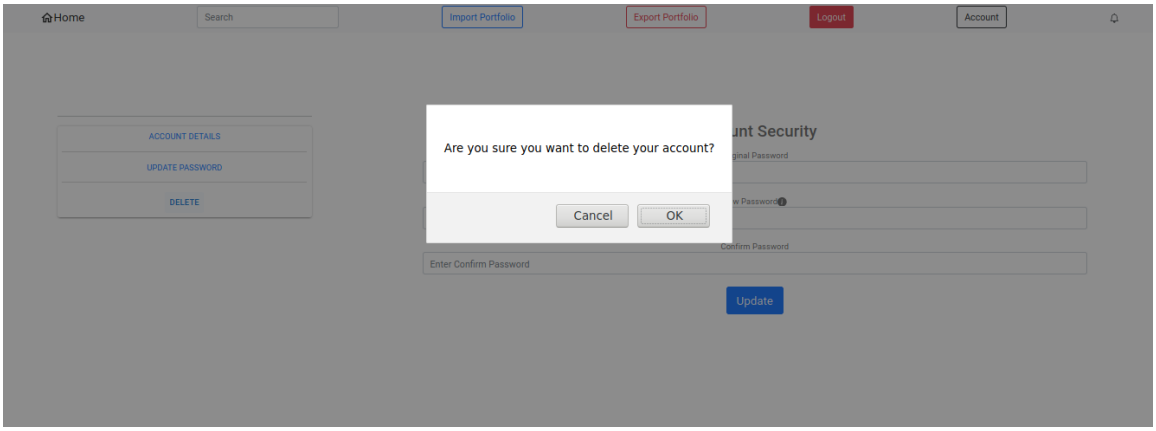
We can also change the name and email.



Once you click the update password tab, it brings you to the page which allows you to reset your password.

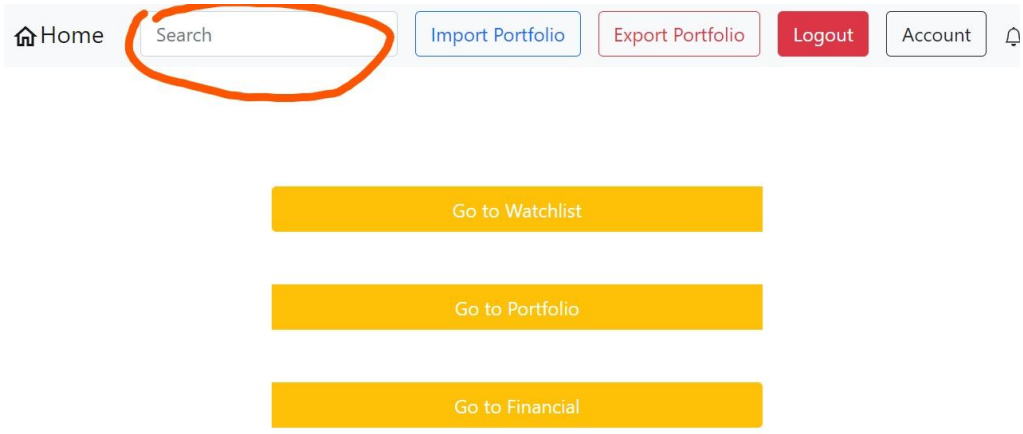


There is also the option to delete your account if you press the delete button.

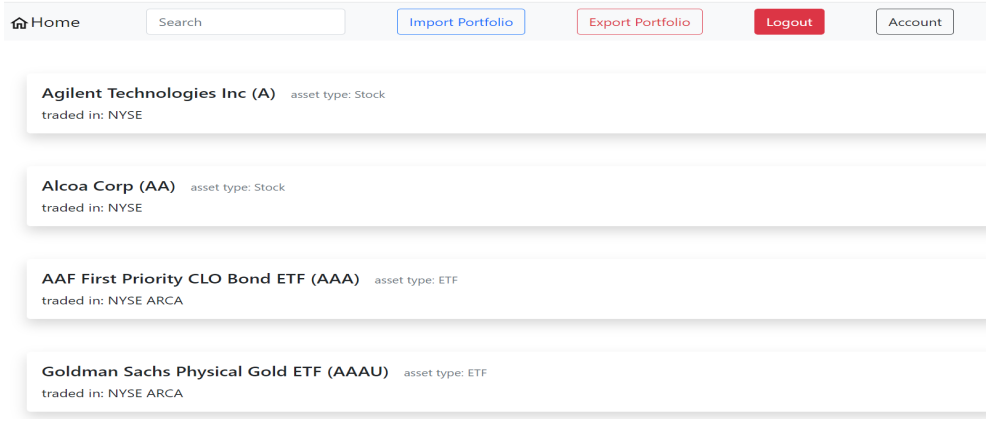


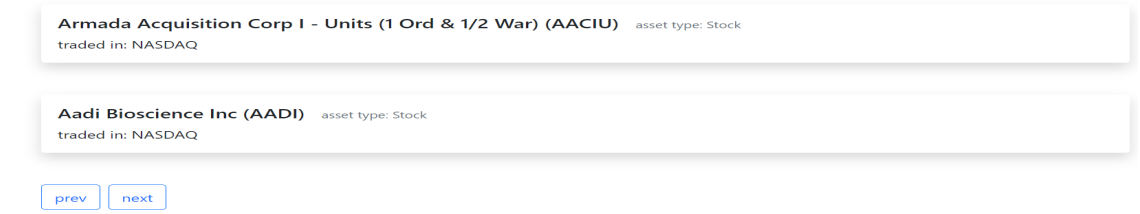
Stock Information and retrieval

Users can check active stock listings in the market. Using the search bar at the top of the site users can choose to either enter nothing and view a list of all stocks or search for a particular stock by typing in the symbol. It also supports pattern matching in case investors would like to search for stocks with a symbol starting with a certain character, or searching for a stock based on company name.

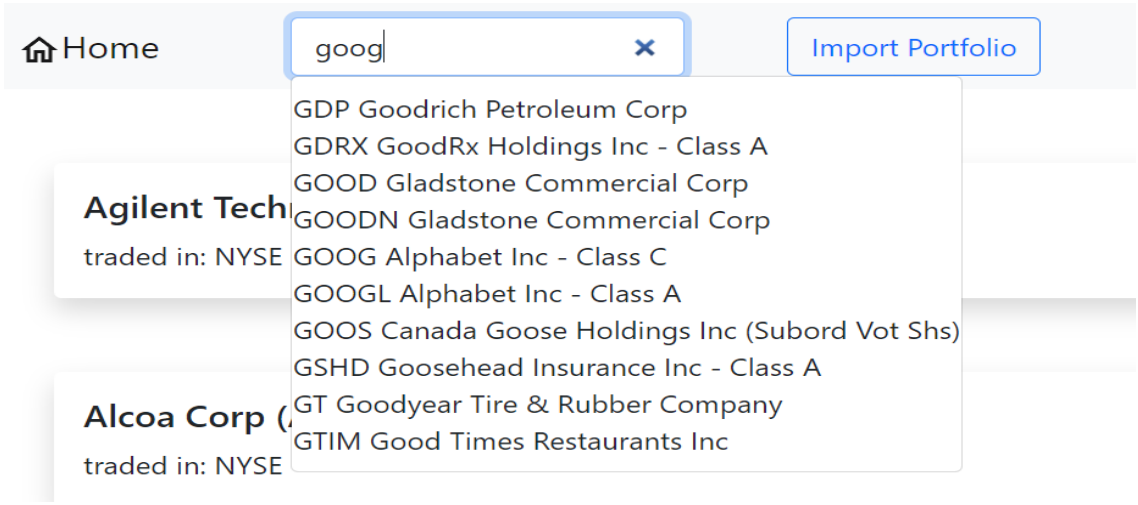


If the user enters nothing it will send them to a listings page which shows all currently active stocks in alphabetical order as shown above, where they can navigate through the list using the “prev” and “next” buttons. They can then click on a stock to navigate to the stock details page which shows detailed information about the stock including its price and historical data.





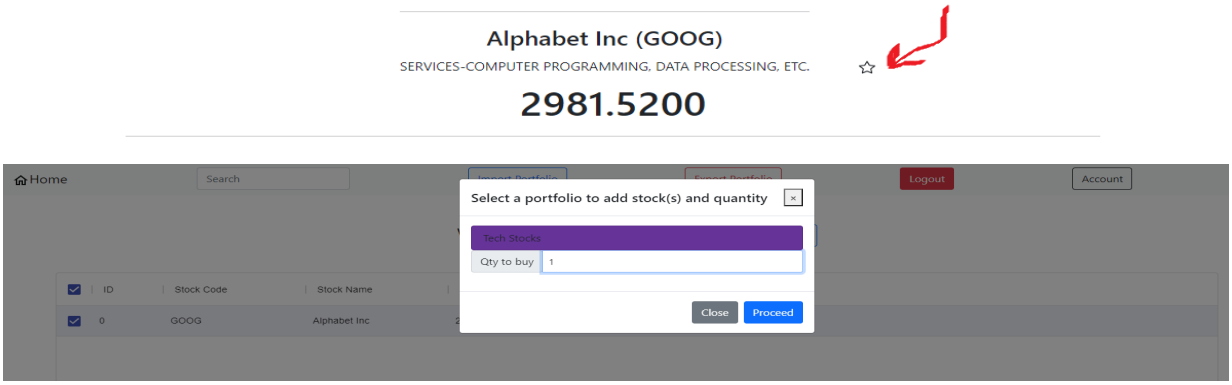
If the user enters something into the search bar, it will show a dropdown of all matching stocks, which the user can click to move on to the stock details page.



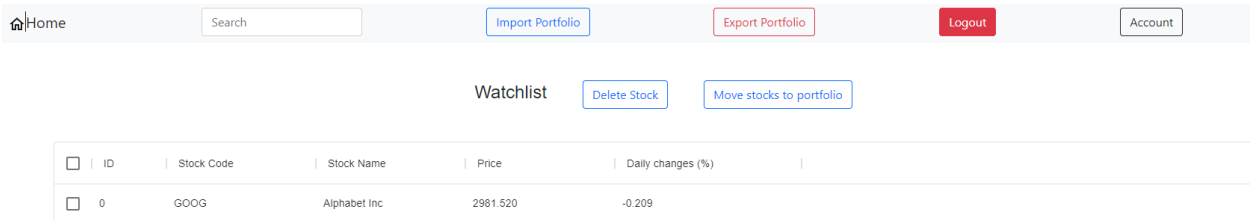
Using this search list and concise stock details page users will be able to view details or search for stocks easily, helping them find what stocks they would like to add to their portfolio.

Watchlist

The application provides a watchlist feature that allows users to keep monitoring the stocks that they are interested in. By monitoring the stocks, they can make decisions to buy or delete from the watchlist.



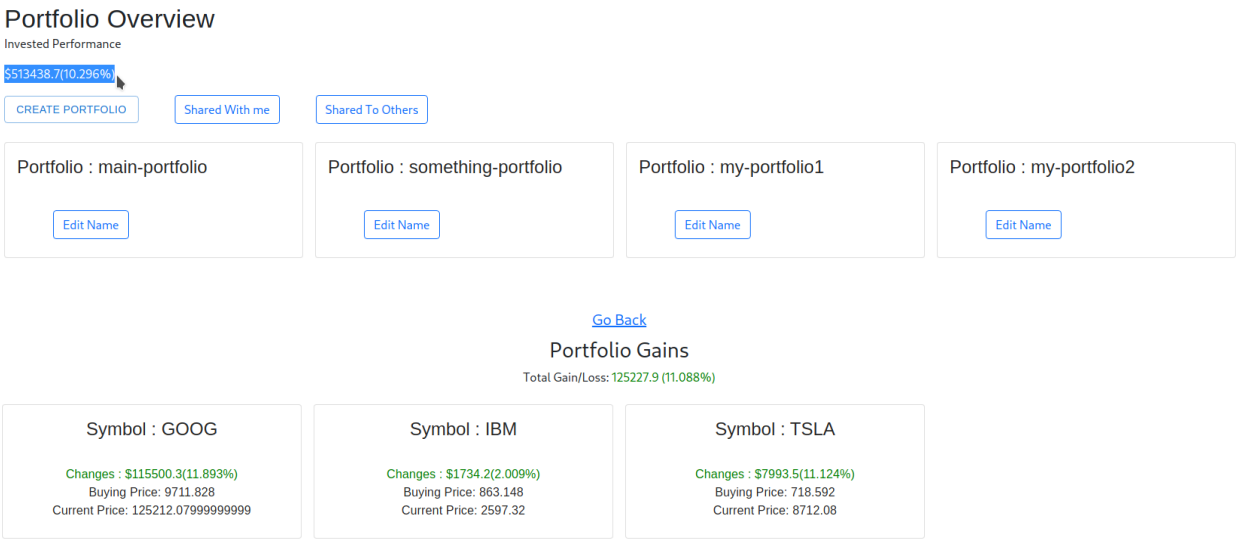
Users have a watchlist where they can add/delete stocks to/from it. Users can add stocks into the watchlist by clicking a star icon on the page of a stock they're viewing. The star icon is going to be shown as black if it already exists in the watch list or upon clicking it(add to watch list).



Users can delete stocks in the watchlist either by clicking a highlighted star icon on the page of a stock they're viewing or clicking a delete button in the watchlist.
Users can move a stock in the watchlist to their portfolio. Such transition can be considered as buying stocks and therefore, units of stock and portfolio name must be provided in order to successfully process the action.

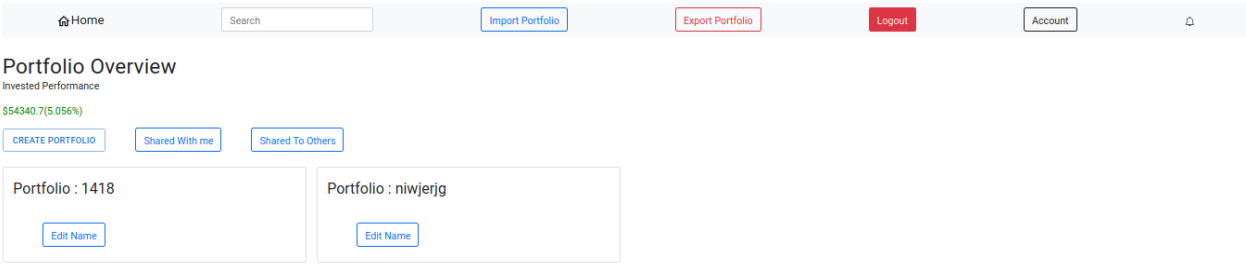
Performance Tracking

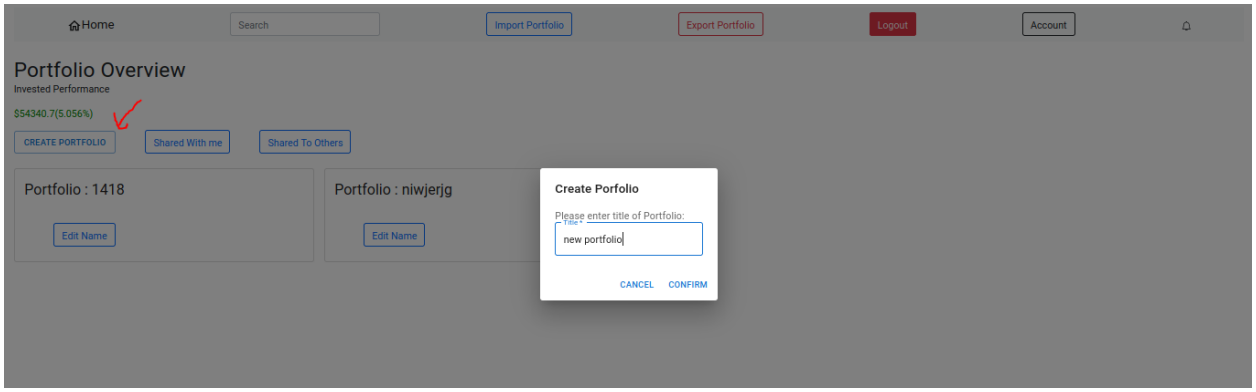
Users can see the overall performance of all portfolios in the Financial Page. In each portfolio page, users will be able to see the current price and the change value and percentage since the data was added. There is also a balance page to display the overall performance of each portfolio and each stock in the portfolio.

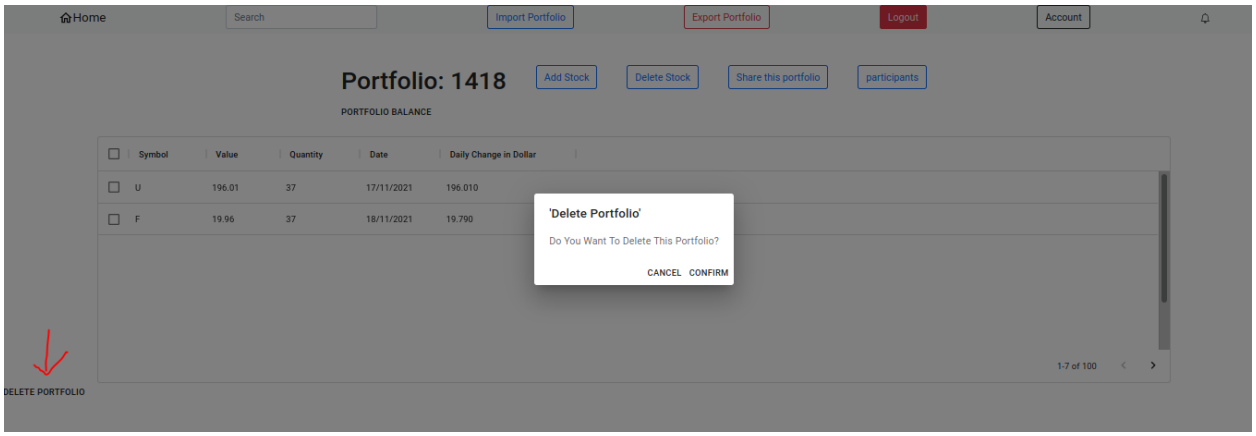


Portfolio Management

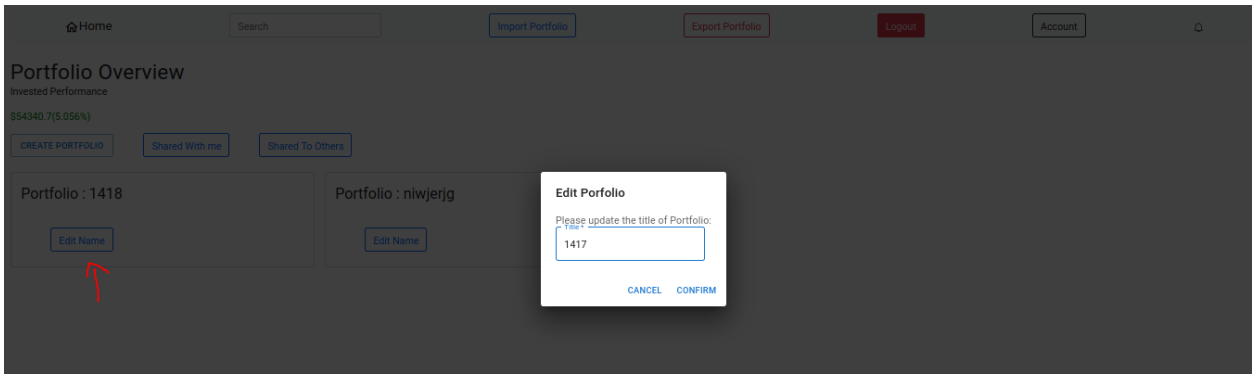
Users can view all the portfolios and their names on the Portfolio Overview Page and create and delete portfolios by clicking the button “Create Portfolio” and “Delete Portfolio” on the individual portfolio page.



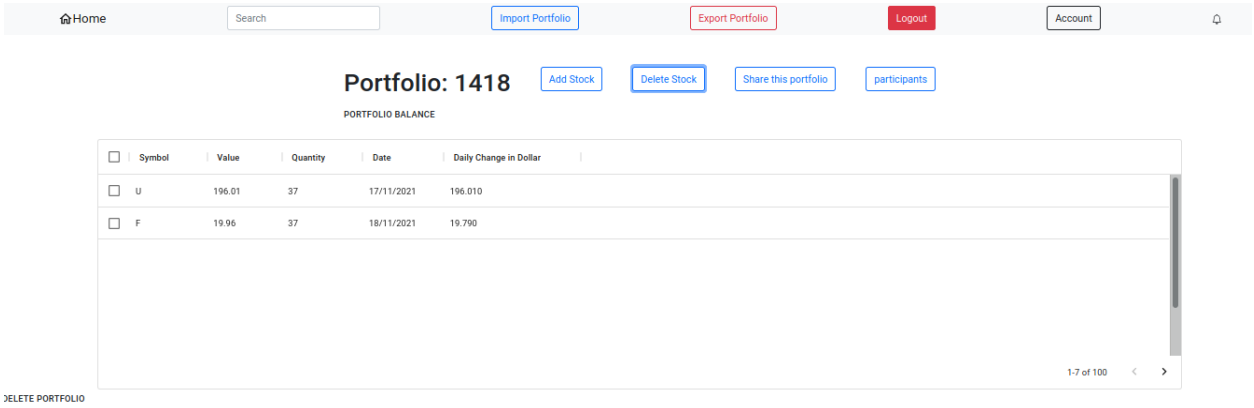


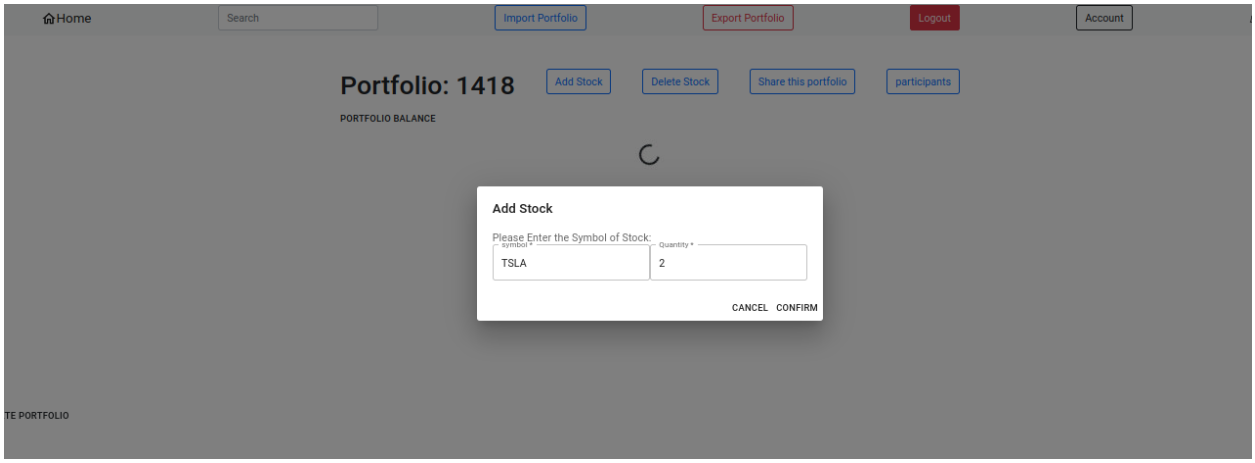


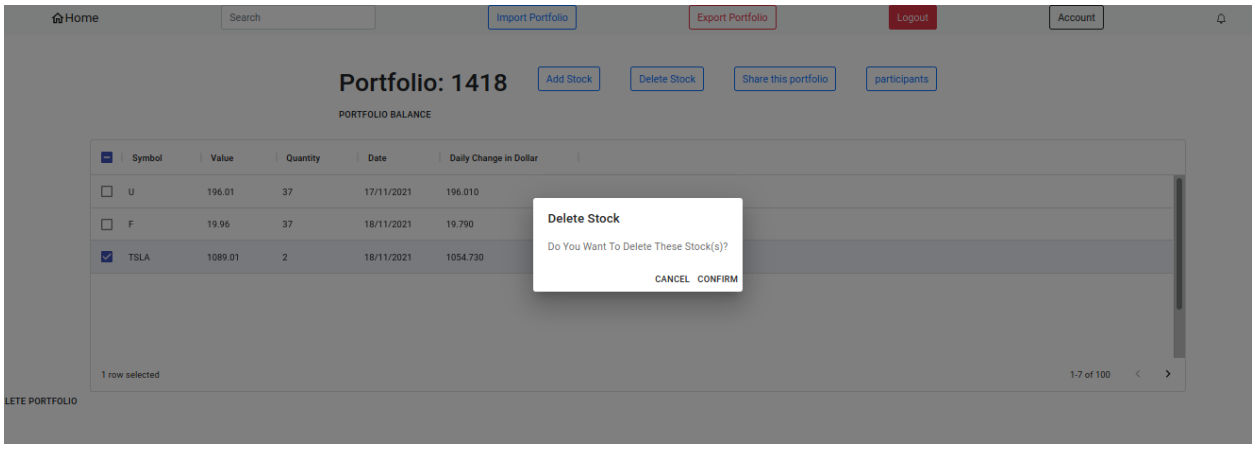
Users can also edit portfolio names in case they change their minds about getting a more accurate name.



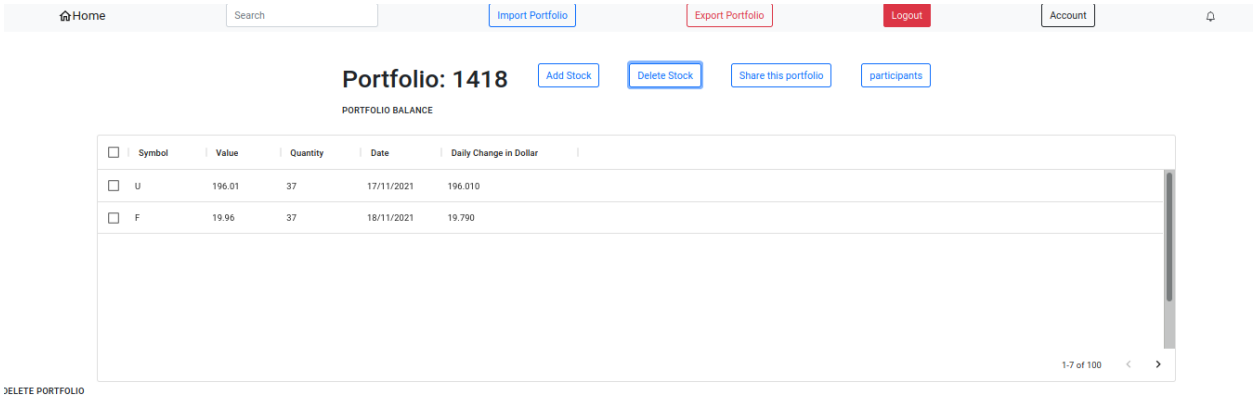
By clicking the name of stock in the portfolio overview, it will navigate to the portfolio click by choice.
In each portfolio, stocks can be added and deleted. Stocks are displayed in a grid with the information of symbol, current price, change of value and percentage.





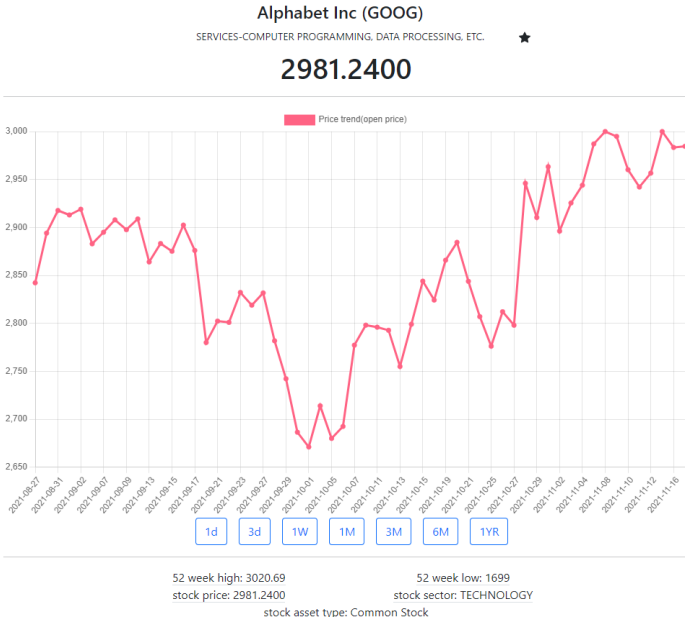


After adding and deleting stocks, the grid will update by itself.



Being able to manage stocks in this manner allows the user complete control over their portfolio and lets them view important information about their portfolios so that they can make informed decisions. These portfolios are designed to mirror the user's actual buying and selling actions so that the portfolio information is always up to date.

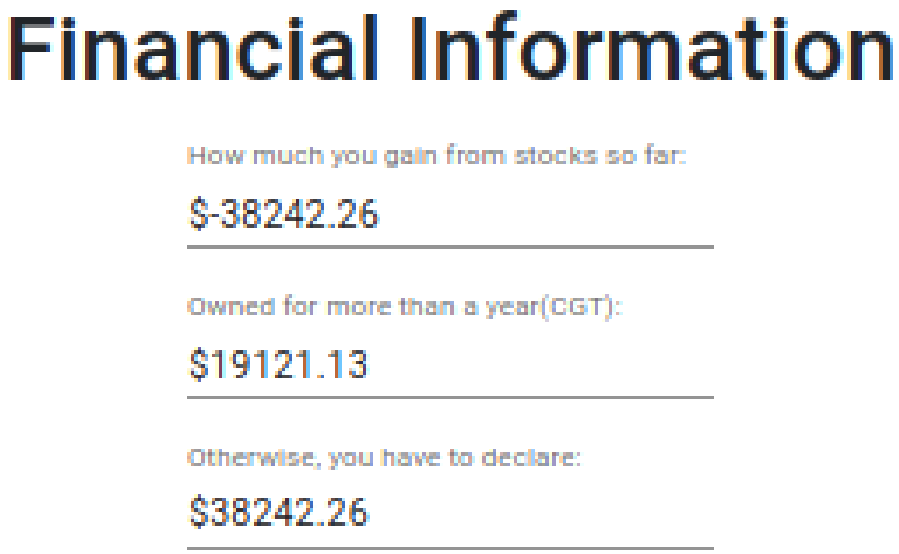
Information visualisation



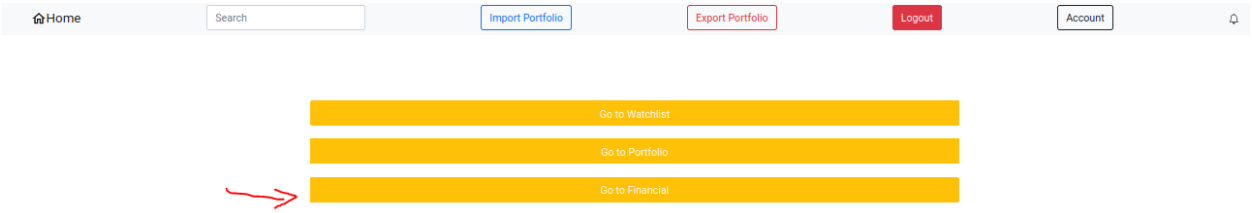
Users are able to dynamically search a stock keyword by keyword. The recommended stocks for the keyword will appear in the search bar. Once a user clicks a stock in the dropdown list of the search bar, users can see the stock's statistics (current value, 52 week high, 52 week low, sector and its asset type. On the stock detail page, there is a graph that shows the 3 months trend of a stock and has options to change time options(1 day,3 day, 1 week, 1 month, 3 months, 6 months and 1 year).

Data aggregation

On the Financial page, it displays the yearly gain/loss of all the stocks across all the portfolios from last year and tax elimination including stocks existing for more than a year (CGT) or overall (to declare). This will be convenient for the user to declare tax and view the financial income from stocks.

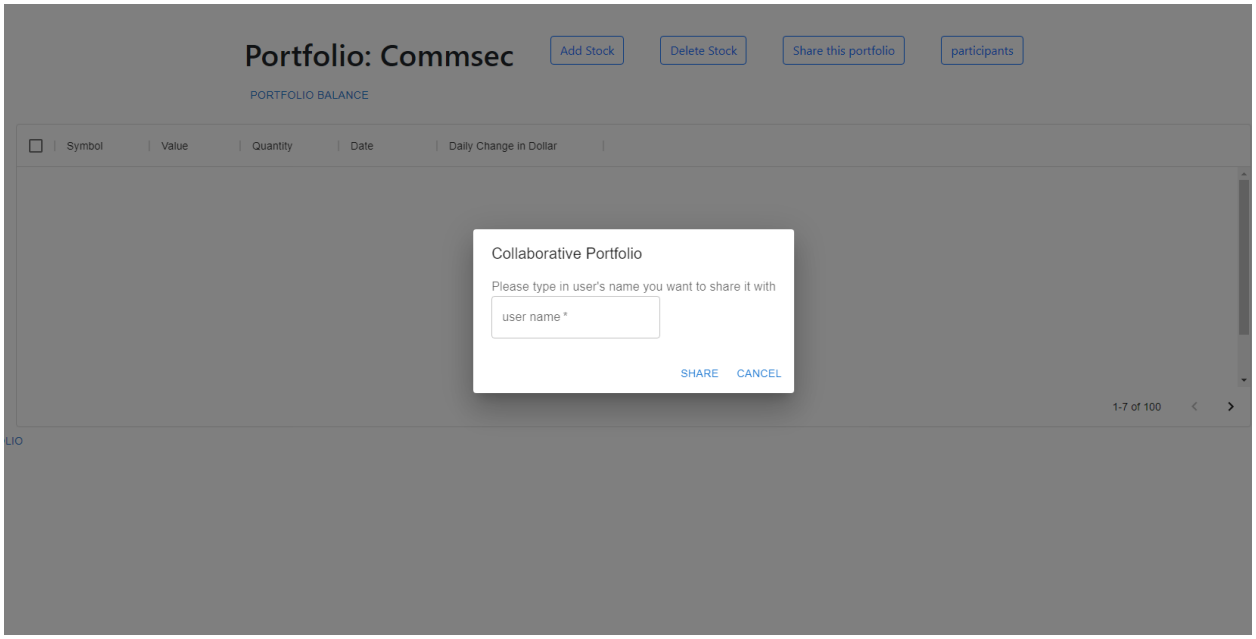


The Financial Page can be found in the button ”Go To Financial” button on the home page.

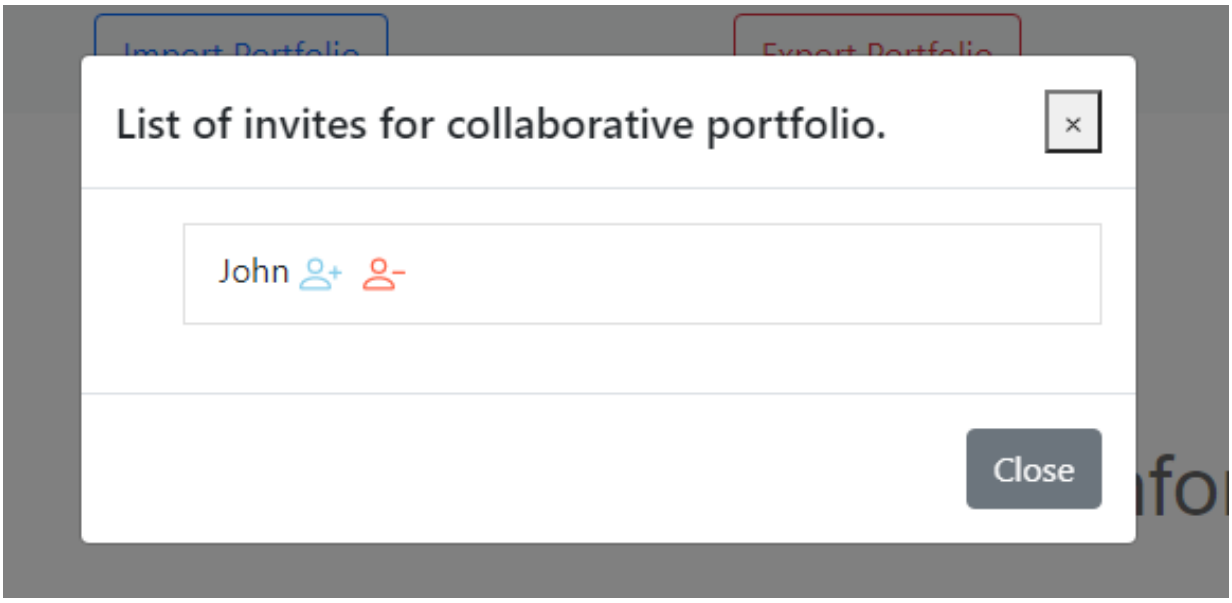


Collaborative portfolio

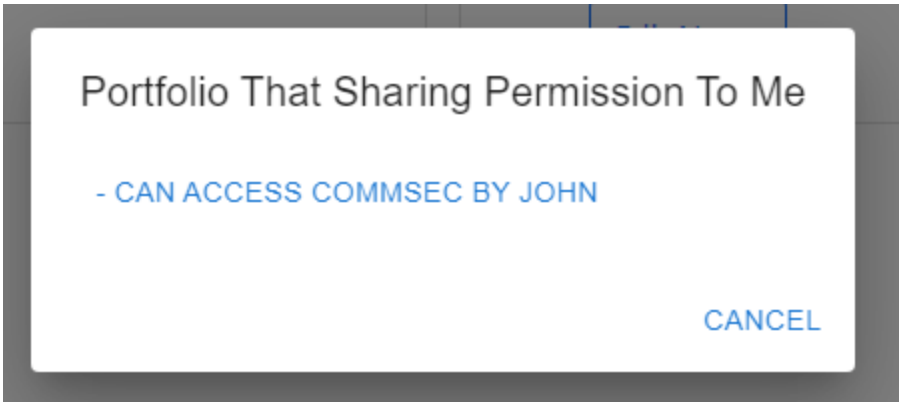
Users are able to share their portfolios with other users on the website. This is achieved by an invite system where you can send invites to other users on the website.



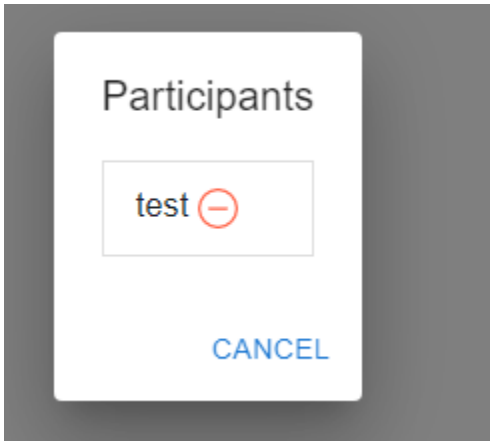
The invited user can choose to accept or reject the invitation to collaborate on another user's portfolio.



On accepting the invite, the shared portfolio will show up on the user's portfolio page where they can view, add, edit or delete stocks there.



The owner of the shared portfolio can also remove permissions of people that they previously invited before, as well as view a list of people that they are sharing their portfolios with.



This feature will be useful to the user if they have a joint account with another person, or if they are sharing an account.

Summary of project objectives from proposal

Our project objective was to create a lightweight stock portfolio management system for newer investors so that they could track stocks easily without being overwhelmed by too much information and features that current portfolio management systems have. We also aimed to add on extra functionality such as portfolio collaboration and data aggregation so that users could work together with others and view metrics that would be useful to them at the press of a button. To do this we proposed functionalities including

- Account management
- Portfolio management
- Stock search and details
- Watchlist
- Authentication
- Collaborative portfolios
- Data aggregation (tax declaration amount, gains statistics, profits since the start)

How these functionalities address all project objective

Our proposed stock portfolio management system allows low to mid level investors or organise and track their invested stocks without overwhelming them with data. We achieve this through the display of **stocks and its details**. We also allow the user to **manage their portfolio** by adding/removing stocks from their portfolio and track stocks they wish to invest in by adding it to their **watchlist** by allowing the user to easily **search** for the stocks. We allow them to seamlessly transfer their watched stocks from their watchlist straight to their portfolio. We also allow users to **collaborate on a portfolio** by sharing it with other registered users. We also give the users the feature to estimate how much Capital Gains Tax they would need to pay by aggregating the data of the gains/losses throughout the year.

Third-party functionalities

Frontend

Bootstrap

Bootstrap is an HTML, CSS and Javascript library which allows users to design/develop a website faster and easier. Bootstrap was used as the main frontend library as it comes with pre-made components with basic knowledge of HTML and CSS.
Bootstrap has saved a lot of time because its pre-build component allows the front-end developers to focus on the logic. Bootstrap components are used only for the basic UIs such as buttons, input box and canvas due to its limitations of customisation. For the complex UIs, Material UI was used(discussed below).

Material UI

Material-UI(MUI) is a react library that allows users to import and use styled components in react.js applications. MUI is easy to use and purely css without third-party libraries. The components design are customizable. It is great to distinguish our web apps by styling it and since stock API is pretty heavy weighted, there is a great balance.
We use MUI Grid, Button, List...etc. For Grid as example, it is flexible in terms of changing the margin widths per column and layout will automatically adjust to the screen.

Moment

Moment is a JavaScript library that is used as a wrapper to the Date Object to help parsing, validating, manipulating and displaying date/time in an easy way. We were able to retrieve time and date using Moment library.
Moment was used to get the current year, month, day when the users add the stocks.

React

React is a javascript library that helps to build and connect complex user interfaces. The team has used React as the main frontend library to decouple the complexities of component’s relationships. Managing states in the application was important to render dynamic components such as stock’s Graph,statistics and Loader. The team was able to implement hooks that React provides to effectively manage and render complex components. Also, with react-router-dom’s help, which is one of the React’s packages, the team was able to build a single page application which is faster and more responsive than the multi page application.

Chart.js

Chart.js is a free and open source javascript library which allows users to customise graphs and visualise it in the Frontend. The team used Chart.js to draw graphs that demonstrate stocks trend/performance in various time options.

Backend

Alpha_vantage

Alphavantage is an API to obtain realtime or historical data for stocks, cryptocurrencies and forex in several markets. In addition to pricing data it also offers a wide range of other data including technical indicators and performance data. It was used as the primary source of market data, providing us with the necessary stock prices to calculate and show information useful to investors.
The API was mainly used to obtain data including:

- Intraday time series data to show key stock values (price, open, close, etc) throughout the day.
- Fundamental data including company information for investors to find out more about the company.
- Historical daily data (close price for each day for up to 20 years before) to show the stocks performance throughout the years and portray this as a graph.

Alphavantage was chosen instead of the other APIs mainly due to its free yet extensive data as many others required a subscription to use even the basic features. It also provided a ready integration with python allowing us to access its features easily, and the data was given in an easy to use json format.

Despite being an easy to use and free data source it also came with limitations due to the licensing terms. The biggest issue we faced while we were using this API was that the free version did not allow for real time stock pricing or any intraday stock data throughout the day. Only data up to the previous day's close was available which impacted how we could approach this system. This was the same for other sources (google finance, yahoo finance, worldtradingdata (marketstack), reuters eikon API, polygon, twelve data) where all the free services had strict limitations on real time data. This meant that any feature that required live data (real time stock prices, intraday price graph) had to be replaced to use the previous day's pricing data, and that we could not show dynamic pricing throughout the day.

Source: https://www.alphavantage.co/

Bcrypt

Bcrypt was used when registering a user, signing in, changing the password and recovering the account feature. It is a password hashing function.

These features required that a plaintext password be passed through a hash function which turns input data into meaningless but unique for different inputs.
The purpose of this is to prevent any meaningful data exposure as the password will be turned into a meaningless piece of data.

Secrets

Secrets is a python module that generates cryptographically strong numbers as opposed to a pseudorandom number generator. A cryptographically strong is less vulnerable to successful attacks due to their unpredictable randomness as opposed to a pseudorandom number generator which can have patterns, thus making it predictable.
This python module is used to create random strings for the recover password feature to create a random password and to create a random string for session tokens.

Flask and Flask-RESTPlus

Flask, a web application framework, and Flask-RESTPlus (a flask extension for rest API's) were used to build the backend of the stock portfolio management system. Flask was used because the majority of the team was familiar with this framework and would cut down on development time due to the ease of use of this framework. In addition, Flask-RESTPlus was used to provide documentation on each and every route that was exposed in the backend, leading to a well organised and documented backend. Both Flask and Flask-RESTPlus are under the BSD 3-Clause License, meaning that there is minimal impact on the result of this project
Source (Flask): https://flask.palletsprojects.com/en/2.0.x/
Source (Flask-RESTPlus): https://flask-restplus.readthedocs.io/en/stable/

SQLite3 Connector

The built-in Python module SQLite3 was used in this project. This module allowed for the communication of code in python with the sqlite3 database, enabling us to easily execute SQL code on the SQLite database. The reason we chose this module was because of the ease of use, since it was built-in into the python standard library, as well as having plenty of support and documentation. This module is open source, so there are no impacts on the results of this project.
Source: https://docs.python.org/3/library/sqlite3.html

Implementation challenges

Circular imports

A minor issue that we encountered in the backend near the start of this project. We got a circular import error since we were loading the database module but needed to call a function in another file, which required access to the database to put some information in. The solution was to move the function call in the database to the end of the file so that all database functions would be loaded and ready to use. This worked, however, it took a while to find this solution, and the resulting code looks out of place, but it works.

DataCollector and SearchIterator classes

A DataCollector class was made so that API calls could be centralised to one class. It was used for any interaction with Alphavantage where it would hold 1 connection at a time.

An iterator was implemented to simulate pages in a search engine, where it would allow the user to go to the previous or next “page”. This was done by creating a bidirectional iterator which would query results based on the current offset/page.

Frontend Style

For the frontend UI library, we choose to use Bootstrap and Material UI. Bootstrap provides lots of pre-built components which makes development work a lot easier but it has got limitations on customisations. With Bootstrap, it is hard or almost impossible to build custom components and therefore, the team has decided to use Bootstrap for essential components such as buttons and search box and chose Material UI to cover the complex components such as data grid and layouts. We were confused if there is a conflict of style in terms of how the apps look with two different UI libraries. After consulting with our tutor, we are reassured that it is ok to use 2 libraries.

Port in use error on VLAB

While we were testing our system on VLAB, we found that the default port number was in use. Since we wanted to reduce setup steps to eliminate any problems that could arise so that it is easier for the person setting up the system, we decided to hardcode a random port number (27439) to use instead of dynamically allocating a port number and using that throughout the system, as this would mean that you would have to change the backend port in the frontend every time you start up the backend.

Limited API capabilities

Due to the limitations of the free version of API (5 requests a minute and no live data) we had to go through a lot of trouble when developing our product. Some major issues were that we could not get live data, which meant that our system had to build around the fact that the latest data we could get as the previous days close, and also the fact that we had limited requests which meant that we had to wait several minutes when doing integration testing. It was to a point where we could not do the demonstration in time without the premium API, which resulted in us purchasing the premium package for the demonstration. Due to the number of API calls limitation, the team had to pre-populate a list of stocks to show in the frontend so that the application pretends to have a dynamic stock search engine. Even the premium package only allows 75 api calls per minute and therefore, it was not enough to provide a dynamic stock search engine.

User documentation

The frontend and backend are split up into two different folders, utilising different languages and frameworks. There will be separate documentation for setting up and running the frontend and backend. This has been tested to work in the VLAB environment. You'll want to start the backend first before the frontend so that there are no errors.

Frontend

Setting up the Frontend

In a terminal:
1. Navigate to the frontend directory using
    `cd stock-portfolio`
2. Install the dependencies defined in the frontend package. A JSON file called package-lock.json and a node_modules folder will be installed.
    `npm install`
3. The team has discovered that not all people were able to install dependencies by the above command. Such problem could be resolved by
    `npm install --force`

Running the frontend

To start the frontend, make sure that all the dependencies are installed by running `npm install` or `npm install --force`. Then run the command `npm start` to start the frontend. Make sure the backend is running before using the apps.

Backend

A more detailed copy of the following instructions can be found in the root directory of the backend.

Setting up the backend

In a new terminal:
1. Navigate to the backend directory using `cd stock-portfolio-backend`
2. Set up a virtual environment using
    `python3 -m venv env`
3. Activate the virtual environment.
    `source env/bin/activate`
4. Install the required dependencies:
    `pip install -r requirements.txt`

Running the backend

To start the backend, make sure that your virtual environment is activated by running `source env/bin/activate`. Then run the command `python3 run.py` to start the backend. First-time start-up of the backend will take the longest due to populating the database.

Viewing documentation

If you want to view the documentation for the backend, you should navigate to `http://127.0.0.1:27439/` when the backend is up and running.

References

1. 2021. [online] Available at: <https://docs.python.org/3/library/sqlite3.h> [Accessed 18 November 2021].
2. Flask-restplus.readthedocs.io. 2021. *Welcome to Flask-RESTPlus's documentation! — Flask-RESTPlus 0.13.0 documentation*. [online] Available at: <https://flask-restplus.readthedocs.io/en/stable/> [Accessed 18 November 2021].
3. Flask.palletsprojects.com. 2021. *Welcome to Flask — Flask Documentation (2.0.x)*. [online] Available at: <https://flask.palletsprojects.com/en/2.0.x/> [Accessed 18 November 2021].
4. Vantage, A., 2021. *Free Stock APIs in JSON & Excel | Alpha Vantage*. [online] Alphavantage.co. Available at: <https://www.alphavantage.co/> [Accessed 18 November 2021].
5. Mark Otto, a., 2021. *Introduction*. [online] Getbootstrap.com. Available at: <https://getbootstrap.com/docs/4.6/getting-started/introduction/> [Accessed 18 November 2021].
6. Momentjs.com. 2021. *Moment.js | Docs*. [online] Available at: <https://momentjs.com/docs/> [Accessed 18 November 2021].
7. Mui.com. 2021. *Usage - MUI*. [online] Available at: <https://mui.com/getting-started/usage/> [Accessed 18 November 2021].
8. Reactjs.org. 2021. *Getting Started – React*. [online] Available at: <https://reactjs.org/docs/getting-started.html> [Accessed 18 November 2021].
9. Chartjs.org. 2021. *Chart.js | Chart.js*. [online] Available at: <https://www.chartjs.org/docs/latest/> [Accessed 18 November 2021].
10. PyPI. 2021. *bcrypt*. [online] Available at: <https://pypi.org/project/bcrypt/> [Accessed 18 November 2021].
11. Docs.python.org. 2021. *secrets — Generate secure random numbers for managing secrets — Python 3.10.0 documentation*. [online] Available at: <https://docs.python.org/3/library/secrets.html> [Accessed 18 November 2021].