Web Scraping and Hosting

Team 1: Jack Chen, William Frame, Samuel Perebikovsky
Team 2: Aniruddha Dutta, Junhao Zhang, Vincent Lan

Background

Investment portfolio management tools are increasingly ubiquitous. Many retail investors use trading platforms like Robinhood and Etrade to manage their stock portfolio. However, these retail trading platforms do not extend well into portfolio management. While institutional investors have strong portfolio management tools, these have not been shared with the retail investor. The following are the issues inadequately addressed by using trading platforms as a portfolio management software.

(1) Lacking functionalities of portfolio visualization on many popular platforms

The most popular millennial trading platform, Robinhood, provides you a pie chart to tell you how much of one's portfolio is invested in each company. This is a solid first step toward applying data visualization to portfolio management; but these platforms have meaningful limitations, as discussed in part (2) and (3).

However, not every platform has such functionalities. Etrade, for example, despite being ranked as a top 5 trading platform, does not offer on demand portfolio visualization to its users. Without appropriate portfolio visualization, investors will not be able to understand their portfolio allocation easily.

(2) Zero account integration across platforms

Not every platform is built the same and each platform has their strengths and weaknesses. For example, Robinhood has better visualization and website design, as well as low margin rates; however, their order routing methods guarantee that investors will not get the lowest price for each stock when buying into the positions. Robinhood lacks tax lot selling which can generate otherwise avoidable ordinary income, which is taxed at a higher rate than capital gains.

TD Ameritrade has favorable order routing for investors, allows tax lot selling and provides the most education and free technical indicators, but their margin rates are far higher than other alternatives. Interactive Broker provides different pricing models for amateur and pro investors, and the lowest margin rates. However, most of their features are provided via their professional trading platform, Trade Workstation, available only at a cost to investors with a portfolio under $100,000. To add insult to injury, these costly features are poorly designed and are technologically reminiscent of the early 2000s. Another competitor, Merrill Edge, provides the best equity research on the market, but their pre-market and

post-market trading is a few hours shorter than other platforms, at a disadvantage for investors interested in trading time flexibility.

The fact that there is not a clear winner in trading platforms heightens the need for a central portfolio management platform, where users can integrate their stock portfolio across different platforms and have a one-stop shop to allow users to manage their portfolio wisely.

(3) Inaccurate sector categorization

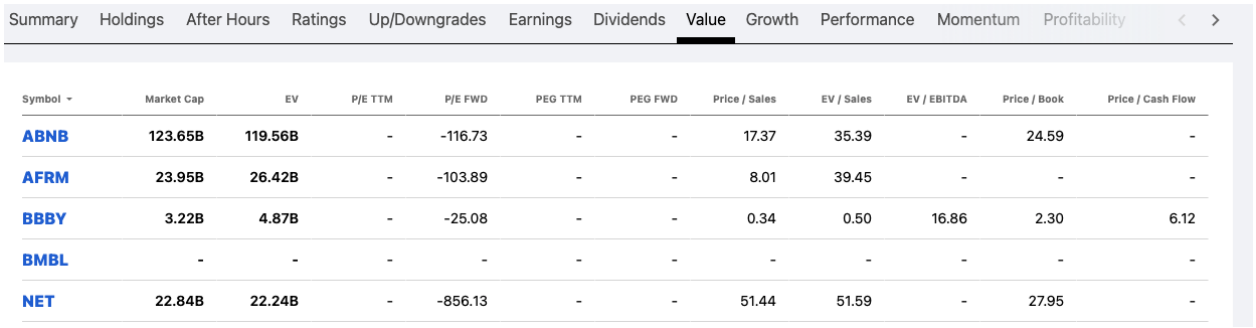
Popular portfolio management platform Personal Capital, for example, provides visualization of sector allocation. However, its sector allocation is confusing. REITs are put into alternative assets, while investors would benefit from knowing what type of real estate is owned by these firms. Many portfolio and trading platforms have unconventional sector categorization, unintuitive to investors.

(4) Lack valuation information, with too much day-to-day price information

Trading platforms are built for trading, not for managing portfolios. This leads to issues with data availability, as trading platforms do not provide data with portfolio management outcomes in mind. For trading, one needs price data, while for long term portfolio maintenance, one needs to look at long term valuation analysis and, for some prudent investors, SEC filings. Trading platforms do not provide enough metrics to be portfolio management platforms.

(5) Inability to customize views

The best alternative on the market, by far, is Seeking Alpha, under its “My Portfolio” tab. It provides a dashboard with multiple subtabs, each presenting an important factor of financial analyses, such as dividends, valuation, and momentum. However, subtabs complicate the user experience. Users need to click through multiple pages to get a full view of their portfolio.



Symbol	Market Cap	EV	P/E TTM	P/E FWD	PEG TTM	PEG FWD	Price / Sales	EV / Sales	EV / EBITDA	Price / Book	Price / Cash Flow
ABNB	123.65B	119.56B	-	-116.73	-	-	17.37	35.39	-	24.59	-
AFRM	23.95B	26.42B	-	-103.89	-	-	8.01	39.45	-	-	-
BBBY	3.22B	4.87B	-	-25.08	-	-	0.34	0.50	16.86	2.30	6.12
BMBL	-	-	-	-	-	-	-	-	-	-	-
NET	22.84B	22.24B	-	-856.13	-	-	51.44	51.59	-	27.95	-

Figure 1 Customize Views

Hopefully, by now, we have managed to convince you that there is not a good solution to portfolio management. There are adequate local solutions but no viable global alternatives. Through this project, we want to provide a portfolio where users can have a one-stop shop where users can monitor their investment and conduct initial due diligence on new investment opportunities.

User Profile

Our target audience is someone who has a basic understanding of portfolio management and company valuation. As Benjamin Graham, the “father” of value investing, put it, “*Mr. Market has another endearing characteristic: He doesn’t mind being ignored. If his quotation is uninteresting to you today, he will be back with a new one tomorrow. Transactions are strictly at your option. Under these conditions, the more manic-depressive his behavior, the better for you.”* While an average retail investor focuses on the price of a given stock, an “intelligent investor” (a term coined by Mr. Graham and the title of his best selling investment book) does their own research based on all available market information, and purchases stocks solely based on their own research. Market sentiment, the primary driver of excessive market volatility or what is referred to as “Mr. Market’s manic-depressive behavior” by Mr. Graham, does not affect one’s performance in the long run. Fundamentals or the intrinsic value of stocks is what drives stock performance in the long run.

A price is a data point you can get anywhere, whether you get it delayed for free or real time for a fee. To intelligent investors, price is not what matters most. We want to allow users to compare fundamentals that indicate the intrinsic value of the stocks. We will be building a platform where interested investors can compare their “favorite” metrics when searching for their next stock purchase or help them understand when to sell a stock.

Our users should have some basic understanding of how to use our program, and possess basic computer skills. Ideally, no Python programming or web development skills are needed to use our project. However, due to the time constraint, some of the accessibility features might not be implemented by the end of this quarter. Therefore, a user will need basic Python and Django knowledge. We believe that an investor with basic portfolio management, company valuation, and technology knowledge will be most likely to pay for our tool if a pricing model is adopted in the future.

Data sources

Our data is scraped directly from Yahoo Finance by using the scraper functionality encoded in our project. Yahoo Finance obtains its data through SEC filings.

Use Cases

Use cases. Describing at least two use cases. For each, describe: (a) the objective of the user interaction (e.g., withdraw money from an ATM); and (b) the expected interactions between the user and your system.

- Case 1: A value investor user wants to decide which company to purchase stocks from next. Their objective is to decide which company to invest in. In this case our program consolidates data such as P/E ratio, EBITDA multiple, volatility, etc. for a value investor about various companies into one screen so the user can get an immediate understanding of the value of a particular group of companies that is inputted by the user. We allow the user to bypass the process of navigating to a new webpage for each company and switching tabs back and forth to compare companies. In this case our program streamlines the research process for value investors.
- Case 2: A data scientist wants to collect data on all the variables associated with some set of tickers. Instead of scraping from various websites individually, they will be able to input the tickers they want to look at and scrape directly from our single page which will contain most of the data they are looking for. This provides a more efficient solution to collecting information on stocks.
- Case 3: A company in a particular sector wants to compare their financial performance to its competitors in the same sector. In this case our program allows the company to view the financial information of any number of its competitors and make informed decisions to grow and beat their competitors. In this case our program can help a company gather information that can assist in creating a competitive analysis.
- Case 4: The government wants to see if an economic bubble is forming within a certain industry. In this case they can compare things like P/E ratio and other data from big companies in the sector of interest to find out if there is a common pattern of over-valuation throughout the sector.

Software Components

1. Financial statistics scraper, which retrieves data relevant to a company's finance and stock. It sends HTTP requests to Yahoo Finance, receives the corresponding webpage in raw texts, and separates financial data from a large textbody of information related to a company. It takes the company URL as input and returns financial statistics in the form of a Python dictionary.
2. Company profile scraper, which retrieves data relevant to a company's general profile. It sends HTTP requests to Yahoo Finance, receives the corresponding

webpage in raw texts, and separates more general information relevant to a company from a large textbody. It takes the company URL as input and returns a company profile in the form of a Python dictionary.

3. Metrics name processor, which transforms the metric names in data to be more understandable. Metrics are not named nicely in text data, so the metrics processor performs some cleaning, as well as abbreviating some metrics names to better display them in dashboards. It takes raw metric string name as input and returns well-formatted metrics name as output.
4. Data scraping error handler, which defines and processes possible errors encountered in data scraping. Some possible errors include invalid URLs, invalid page information, wrong-format data, etc. As an error handler, it does not have any particular inputs. Instead, when an error occurs, it reports the company that the scrapers are trying to fetch data from, or the URL in cases of invalid URLs.
5. Data model constructor, which constructs stock objects that can be directly displayed in our home page. It takes Python dictionaries with data as input, and returns Stock objects with data as attributes.
6. Home view controller, which specifies the information shown in the home page. It displays the stock information in a table with one stock per row and one attribute per column. As a controller, it takes user actions as its input. Instead of giving specific outputs, it changes/filters information in the table as necessary. While time permits, it should also allow as many view manipulation functionalities as possible, such as searching, sorting, etc. to support some of our use cases.
7. Router, which navigates users to different webpages in our application. Instead of receiving direct input, the router is activated whenever users click on one of the links (home page, add stock, or user I/O for user data) in the navigation bar, and direct users to corresponding pages.
8. Stock data manager, which is responsible for adding, deleting, and updating all stock data kept within our application. It takes user actions (add, update, or delete) as input, calls the scrapers if necessary, and adds, deletes, or modifies existing stock data. It has no direct outputs.
9. User data manager, which is responsible for keeping track of whether a user is logged in. Our application only sends information to registered users.

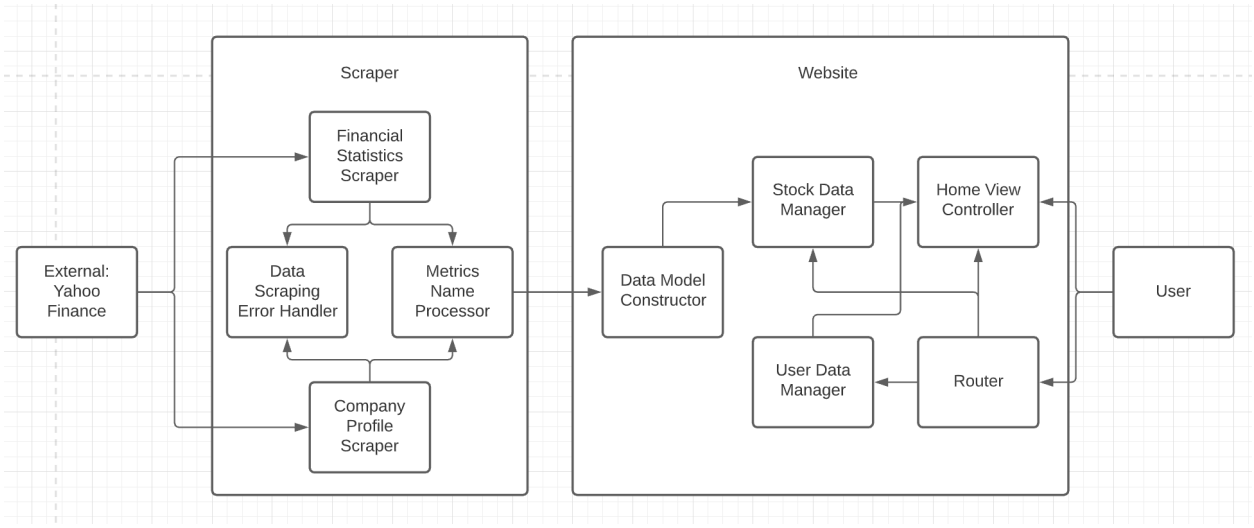


Figure 2 Software Components

Interactions to Accomplish Use Cases

For use case 1, when a value investor comes to our site, he can see a financial dashboard in our home page displayed by home page controller (#6). The stock data on this page came from stock data manager (#8), which, for each individual stock, called the financial statistics scraper (#1) and the company profile scraper (#2), processed the returned data for better readability with the metrics name processor (#3) and data model constructor (#5). Of course, all of this information is provided only if the user data manager shows the user has logged in(#9).

For use case2, the user will be able to input the fields/features they are interested in which will be managed by the home view controller (#6) and the stock data manager (#8). The stock data manager equipped with the user input selections will fetch data using the financial statistics scraper (#1) and the company profile scraper (#2).

The home view controller (#6) also enables the users to compare two stocks as described in use case 3. The user can input the name of the stocks they wish to compare along with the features they are interested in which will call the financial statistics scraper (#1) and company profile scraper (#2) and display the processed data with metrics name processor (#3) and data model constructor (#5).

Preliminary Plan

1. Build the data scraper. Completed

2. UI Design. Completed
3. Build the front-end financial dashboard with company information. Completed
4. Stock search function. Completed
5. Implement Pytest, Travis CI, Confidence, Flake8, Pylint. Completed
6. Incorporating scraper and django components. Completed
7. Transform scraper output to readable strings. Completed
8. Unit Test incorporation. In Progress
9. Enable user inputs (shares, purchase date), and implement I/O of user information. In Progress
10. Export the user inputs and dashboard information into excel. In Progress
11. Readme.md and Documentation. Scheduled for the week of March 1st.
12. How-to documentation. Scheduled for the week of March 8th.

Stretch

13. Visualization. Potentially for the week of March 8th, if time and bugs permit.
14. Dynamic Tables on HTML that allows users to pick fields of interest. Potentially for the week of March 8th, if time and bugs permit.
15. Test, Test, Test. In Progress

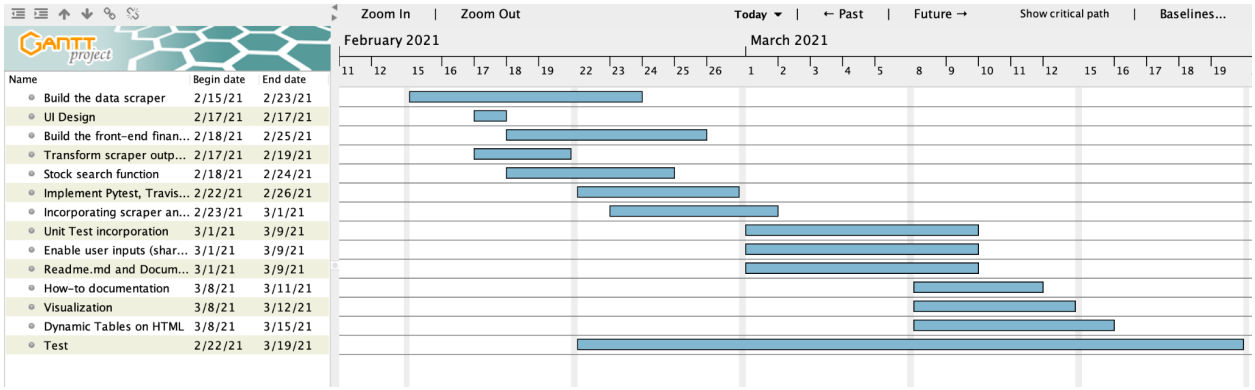


Figure 3 Gantt Chart