

RO2

RO2 FYP Final Report

Intelligent Financial Trading Platform - Customizable Machine Learning Predictions and Automated Trading Strategies using IFTTT Methodology

By

HUANG Izen Brast, ROHATGI Tanay, WONG Siu Kit Martin

Advised By

Prof. David Rossiter

Submitted in partial fulfillment

of the requirements for COMP 4981

in the

Department of Computer Science

The Hong Kong University of Science and Technology

2019-2020

Date of submission: April 29, 2020

Abstract

Stock trading has been utilized by many as a potential way to yield profit. While a large number of people are trading these days, many others are discouraged by the potential risks and complexity of trading. Some seek to mitigate risk through Electronic Trading (e-trading) platforms, which provide the user with a plethora of information displays, analysis tools, and commonly practiced trading strategies. Others seek to mitigate this risk by utilizing Machine Learning (ML) to predict stock prices. However, we have yet to see an e-trading platform that seamlessly integrates ML stock price predictions into the User Experience (UX).

Our application, which we named **Delta Trading Platform**, is a proof-of-concept intelligent financial trading platform aimed to improve the trading UX through bringing ML predictions to an e-trading platform. It not only matches a state-of-the-art e-trading platform, but goes above and beyond by exercising the possibilities of what an integration of an e-trading platform with ML predictions would look like. Our platform is equipped with a clean & user-friendly dashboard, a stock analysis page with a plethora of stock analysis tools for both real market data and predicted future data, a settings page that allows users to customize the prediction model, and a strategy page that allows users to include ML predictions as part of strategy. Delta Trading Platform showcases how powerful an e-trading platform could be if integrated with the power of ML.

We spent the majority of our focus on designing a well-rounded application, with a peripheral focus on the arduous task of generating an accurate prediction model. Of the 34 responses we received from our User Experience survey, 28 users (82.35%) rated 8 or above out of 10 on their overall satisfaction level, and see Delta Trading Platform as a product they would use on a regular basis when stock trading.

Table of Contents

1. Introduction	6
1.1 Overview	6
1.2 Objectives	7
1.3 Literature Survey	8
1.3.1 Literature Survey on Existing e-Trading Platforms	8
1.3.1.1 E*TRADE	8
1.3.1.2 TD Ameritrade	8
1.3.1.3 Summary	8
1.3.2 Stock Price Prediction with ML	9
1.3.2.1 Stock Price Prediction Using Machine Learning and Deep Learning Techniques	9
1.3.2.2 Machine Learning Techniques applied to Stock Price Prediction	9
1.3.2.3 20 Minutes Trying to Predict the Stock Market with AI	10
1.3.2.4 Summary	10
2. Methodology	11
2.1 Design	11
2.1.1 Designing the System	11
2.1.1.1 System Architecture	11
2.1.1.2 Frontend	13
2.1.1.2.1 Portfolio page	13
2.1.1.2.2 Stock Analysis	18
2.1.1.2.3 Strategy Page	23
2.1.1.2.4 Settings Page	30
2.1.1.3 Designing the Backend	32
2.1.1.4 Designing the Data Pipeline	36
2.1.2 Designing the ML Prediction Model	36
2.2 Implementation	38
2.2.1 Implementing the System	38
2.2.1.1 System Architecture	38
2.2.1.2 The Platform	38
2.2.1.3 Data Pipeline	43
2.2.2 Implementing the ML Prediction Model	43
2.3 Testing	47
2.3.1 User Experience	47
2.3.2 Browser Loading Test	49
2.3.3 Further Testing	51

2.4 Evaluation	51
3. Discussion	52
3.1 Shift of Focus from Research to Application	52
3.2 Challenges in Stock Price Prediction	52
3.3 Challenges in Application Development	52
4. Conclusion	54
5. References	55
6. Appendix A - Meeting Minutes	57
6.1 Minutes of the 1st meeting	57
6.2 Minutes of the 2nd meeting	58
6.3 Minutes of the 3rd meeting	59
6.4 Minutes of the 4th meeting	60
6.5 Minutes of the 5th meeting	61
6.6 Minutes of the 6th meeting	62
6.7 Minutes of the 7th meeting	63
6.8 Minutes of the 8th meeting	64
6.9 Minutes of the 9th meeting	65
6.10 Minutes of the 10th meeting	66
6.11 Minutes of the 11th meeting	67
6.12 Minutes of the 12th meeting	68
6.13 Minutes of the 13th meeting	69
7. Appendix B - Project Planning	70
7.1 Division of Work	70
7.2 Gantt Chart	71
8. Appendix C - Required Software and Hardware	72
9. Appendix D - Miscellaneous Items	73
9.1 A list of supported Technical Indicators	73
10 Appendix E - User Testing	74

1. Introduction

1.1 Overview

In the financial world, there exists a plethora of electronic trading platforms, all designed as platforms to help clients make financial transactions. For example, Robinhood [1] provides the ability to buy/sell products commission free, while other platforms such as Bloomberg [2] provide a multifaceted dashboard to monitor stock prices and news headlines in great detail. These platforms may provide more boutique options, such as detailed portfolio analysis & rebalancing, or a strategy page to implement algorithmic trading.

Furthermore, in the world of artificial intelligence, there are several projects aimed at predicting stock prices, but none has yet to be integrated into an e-trading platform. If a client wishes to gain machine learning (ML)-generated insights, he would have to search for a 3rd party stand-alone application outside of the trading platform and then go back to the trading platform to make judgments on transactions. In the market of e-trading platforms, there has yet to be a mainstream application that fully integrates ML-generated stock predictions.

While features such as stock price monitoring, live news feed, and portfolio analysis are all essential for a retail client, there are few retail trading systems that provide services beyond basic monitoring and “displaying” of information. A traditional stock trading platform still relies on the user to analyze the information from the dashboard and/or other sources to make judgments. A dashboard providing monitoring of stock prices and news is limited, because the client cannot take any follow-up action based on this information. Strategy and algorithmic trading tools are very verbose and difficult to understand for a user without a strong technological background. Thus our system aims to provide a system for retail investors to leverage these different types of tools (dashboard monitoring, portfolio analysis, algorithmic trading, and stock prediction) within a single platform to make financial gains.

Our project, **Delta Trading Assistant**, is an intelligent all-in-one platform for the trading needs of retail investors. It provides a graphical dashboard with customizable ML algorithms for stock price prediction, and an If This Then That (IFTTT) - based automated trading.

We aimed to create a system that goes above and beyond providing the basic functionalities found in other trading platforms, serving as a digital ‘assistant’ for the retail investors. The system assists the user by providing machine-generated stock predictions and giving the user the option to include the machine-generated predictions as part of his strategy. Machine-generated predictions provide the user with advantageous insights for stock prediction; if the ML model can accurately predict stock movement, the users can capitalize on the predictions by buying/selling ahead of time. By adding ML-predictions into the set of strategic trading conditions, the user may have an advantage over other retail investors, who normally don’t have ML tools to work with.

Moreover, we aimed to provide ‘customizable predictions,’ in which the user can modularize the ML model to generate predictions. For example, user ‘A’ might prefer using ML model ‘A’, whereas another user ‘B’ might trust the accuracy of model ‘B’. In our system, the user gets to see both the graphical display and quantitative prediction values for their respective models. This allows users to use the models in which they have higher confidence in the prediction of stock prices.

Lastly, we aimed to create an easy-to-use, non-programming-based strategy page. With our *IFTTT*-based graphical user interface (GUI) logic, users can create strategies without needing to learn the platform’s custom language (like NinjaTrade users need to use Ninja Script [3]), or any traditional programming languages (C#, Java, etc.) Our users can create custom strategies with a few clicks, and without traditional programming. We also provide a portfolio page for the users to analyse their stock holdings.

1.2 Objectives

In our project, we aimed at researching and designing a proof-of-concept trading platform that provides stock traders with an integrated online trading system. The majority of focus (roughly 80%) is on software technology, with the smaller portion on artificial intelligence. The system includes aspects of ML to generate stock price predictions in a modularized fashion on our online platform. To achieve this, we worked on the following 4 objectives:

1. Providing a strategic asset analytics portal for our clients to analyze and formulate their trading strategies.
2. Providing multiple ML-based stock prediction models based on technical analysis¹ (technical indicators).
3. Providing our clients the freedom to use a variety of ML models’ predictions as signals in their trading algorithm.
4. Providing users with the ability to create simple *IFTTT*-like trading strategies that utilize ML predictions as part of the condition, on a user-friendly GUI.

¹ Technical analysis is a trading discipline employed to evaluate investments and identify trading opportunities by analyzing statistical trends gathered from trading activity, such as price movement and volume. (Investopedia.com)

1.3 Literature Survey

1.3.1 Literature Survey on Existing e-Trading Platforms

We conducted a survey of two of the generally recommended online platforms for online stock trading by various financial blogs. The three platforms we surveyed were: E*TRADE, and TD Ameritrade.

1.3.1.1 E*TRADE

E*TRADE's flagship platform [4] allows users to build diversified portfolios and manage assets. It provides investing guidance, including the option to "let" E*TRADE make investments for the client with professionally-managed accounts from E*TRADE Capital Management.

E*TRADE further builds the user's investment knowledge by providing online videos, articles, and classes. They also provide clients with E*TRADE takes on current market trends, so as to help them stay up to speed with active trader commentary.

1.3.1.2 TD Ameritrade

Much like E*TRADE, TD Ameritrade [5] also provides education, with an Education Center users can visit to expand their investing knowledge with learning tools and practice assessments. A distinguishing feature of TD Ameritrade is its "Social Signals" feature, which pulls insights from Twitter and compiles them all on the trading platform.

TD Ameritrade also has its own custom strategy language: thinkScript. With their "condition wizard jumpstart tool", or a modicum amount of object-based coding knowledge, users can build and execute, and strategic testing algorithms and experiments to see how different scenarios would play out.

1.3.1.3 Summary

Most online trading platforms cover the same basic features (i.e. easy stock trading capabilities, a clean GUI, and stock trading tips), and some provide a great amount of knowledge-based services (videos, blogs, and webinars).

Although our system does not include the education aspect of stock trading, it does provide ML predictions, which provide further insight into stock market movement. As of now, there are only a few retail trading platforms that integrate ML predictions into trading strategies.

1.3.2 Stock Price Prediction with ML

The task of stock price prediction using ML has been attempted by many. We did an online survey and found the following systems related to our project. The first two detail the authors' attempts at predicting the adjusted closing prices of certain stocks, while the third one explains a 20-minute project for developing a model to predict stock price movement based on news headlines.

1.3.2.1 Stock Price Prediction Using Machine Learning and Deep Learning Techniques

[6] compares different approaches for technical analysis to predict the daily closing prices of stocks. It starts from simple algorithms, such as averaging and linear regression, and moves on to advanced techniques, such as long short-term memory (LSTM). The dataset used in this project was acquired from Quandl.

The article discusses several approaches, but it states that LSTM is the most effective one. LSTM is a common approach used for time series prediction. The reason why LSTM works well is that it can store information that is important and “forget” information that is not. Although the LSTM approach resulted in the least mean squared error out of all the models considered, it was not a good choice to predict whether the stock price would increase or decrease. The author concluded the article by mentioning that the stock price is influenced by a multitude of factors, and there are intangible factors as well which can be impossible to predict beforehand. It is also important to note that the forecast horizon for LSTM was only 1 day while the other methods discussed had a forecast horizon of 1 year.

1.3.2.2 Machine Learning Techniques applied to Stock Price Prediction

The problem statement of [7] was to “predict the daily adjusted closing prices of Vanguard Total Stock Market ETF (VTI), using data from the previous N days.” The author clarifies that his approach was inspired by the previous article [6] mentioned in section 1.3.2.1. While most of the approaches discussed in this article [7] were similar to those of [3], the main difference was in the last approach: extreme gradient boosting (XGBoost). The author used several features: adjusted closing prices of the last N days, as well as the volume of the last N days, the difference between high and low for each day of the last N days, and the difference between open and close for each day of the N days. Compared with the approach outlined in [6], [7] involves more fine-tuning of the parameters.

Much like the approaches discussed in [6], our system also heavily incorporates numerical & technical features in its models, such as the difference between high and low for each day, or the difference between open and close for each day of the past N days. We tried to incorporate several features into our models to make them more accurate.

1.3.2.3 20 Minutes Trying to Predict the Stock Market with AI

The author of [8] trained a model that received input of news headlines, and classified the news into two different classes: the market went up after the headlines, or the market stayed the same or went down. While the resulting model didn't have amazing results (54% accuracy), the author provided a valuable dataset ("Kaggle Daily News for Stock Market Prediction"), which was valuable to us in our model training. The dataset was essentially a giant table of news headlines, labeled with the Dow Jones' performance each day.

A key takeaway from this project was a clearly defined "market" the author wanted to predict, the New York Stock Exchange, which is represented by the Dow Jones Industrial Average (DJIA). This concept became quite crucial in our own project. Defining a scope reduced the variance. The concept of sentiment analysis is also something we decided to utilize.

The author concluded that news headlines can't be used to predict the daily DJIA closing price with the dataset that he had. The article further gives further insight on the fact that using machine learning to predict stock prices may not be the best use of machine learning, especially because predicting stock prices is a multi-factored, complex, and interdependent task.

1.3.2.4 Summary

The approach described in section 1.3.2.3 is extremely similar to our initial model/attempt at predicting stock movement through financially targeted news, while the first 2 pieces of literature (1.3.2.1, 1.3.2.2) provide insight on how we could potentially add/improve upon the approach discussed in 1.3.2.3 Our approach will start with relatively simpler models, such as the model described in section 1.3.2.1, and continue to analyze, and potentially implement more numerical features described in sections 1.3.2.1, 1.3.2.2.

The approaches of [6], [7], and [8] were all helpful in designing our system's models. We are using both time series data and sentiment analysis based on financial news headlines to predict stock prices.

2. Methodology

2.1 Design

The majority of our project focus is on software technology. The following sections will be a thorough walkthrough of the design component of our proof-of-concept (POC) product, to demonstrate the idea of adapting machine learning models created in a variety of frameworks within our single system and allow users to do automated trading using predictive insights generated by our these models. We will first discuss our design process for the Intelligent Financial Trading Platform, then the Machine Learning Prediction Model we made using LSTM.

2.1.1 Designing the System

2.1.1.1 System Architecture

The architecture comprises three major components, 1) the user-facing frontend interface, 2) backend web servers that process all requests from the frontend interface and 3) databases to facilitate the storage and provide data for computation.

Not only do we have to provide real-time data so users can react and update their trading strategies, we also have to store the data for further analysis (to be used by the ML models). In order to accomplish these features, we will utilize a data pipeline that fetches data from our data provider and stores it in our time series database, thus providing data necessary for users' algorithmic strategies and user-defined monitoring rules to extract important events. Moreover, the user-defined machine learning algorithms will be run on the data in our database to generate predictive insights.

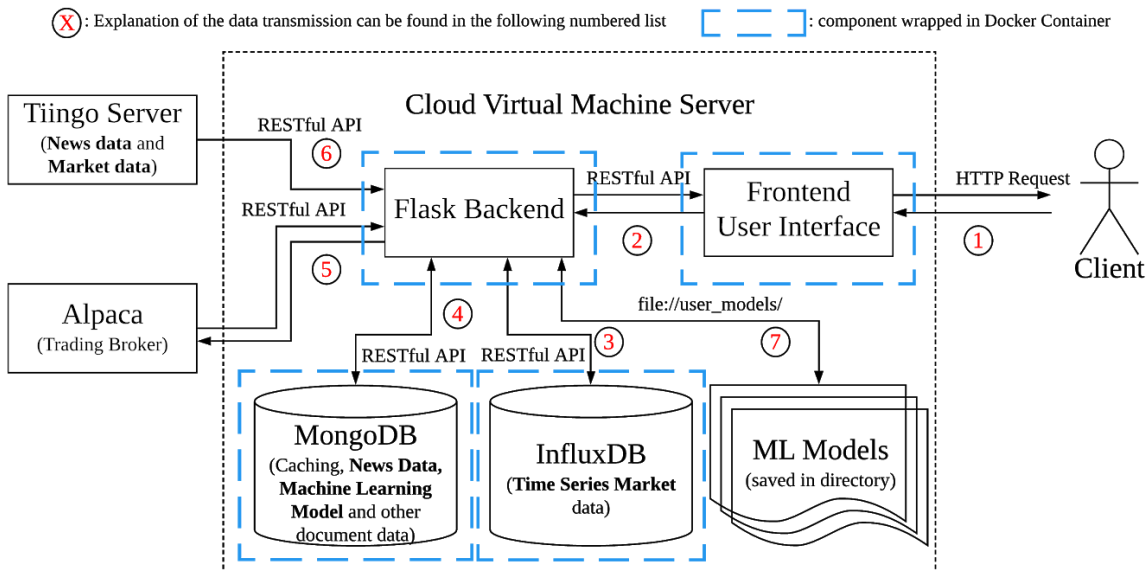


Figure 1. System Architecture; the user interacts with the frontend, while the frontend communicates with the backend through RESTful APIs

1. Clients access our frontend interface through an Uniform Resource Locator (URL) through HTTP Protocol in their web browser. The frontend user interface microservice will return an interactive web page to the client.
2. The frontend will send requests from the users to the Flask backend REST²ful API³ endpoints. The backend server will process the requests, then query and compute the data from databases and finally return the response to the frontend.
3. The backend will fetch time series data, like stock prices and volumes over time from InfluxDB⁴ to be shown on the frontend. And it will upload time series data downloaded from the data provider to InfluxDB.
4. The backend will use MongoDB⁵ as a caching server to enhance the performance of data fetching, like getting news headlines. Also we will store machine learning profiles needed for the system here.
5. The backend will send requests to Alpaca⁶ for querying assets' positions⁷ and trading orders status. Our system will also send trading orders to Alpaca for trading executions,
6. The backend will periodically fetch data from Tiingo so as to feed data to users' trading strategies and make it available for further analysis.
7. The backend flask server will access the Machine Learning Model files in order to load the models into memory and generate stock price predictions using the models.

² REST: REpresentational State Transfer

³ API: Application Programming Interface

⁴ InfluxDB is an open-source time series database developed by InfluxData

⁵ MongoDB is a cross-platform document-oriented database program

⁶ Alpaca, a registered securities broker, lets you build and trade with real-time market data for free

⁷ A position is the amount of a security, commodity or currency which is owned by an individual, dealer, institution, or other fiscal entity (Investopedia.com)

2.1.1.2 Frontend

The frontend is divided into four main components: 1) the Portfolio page, 2) Stock Analysis page, 3) Strategy page, and the 4) Settings page.

2.1.1.2.1 Portfolio page

This page contains multiple sections that allow the user to keep track of the performance and activities of his/her portfolio. In the following sections, we are going to walk through each of the components inside this page.

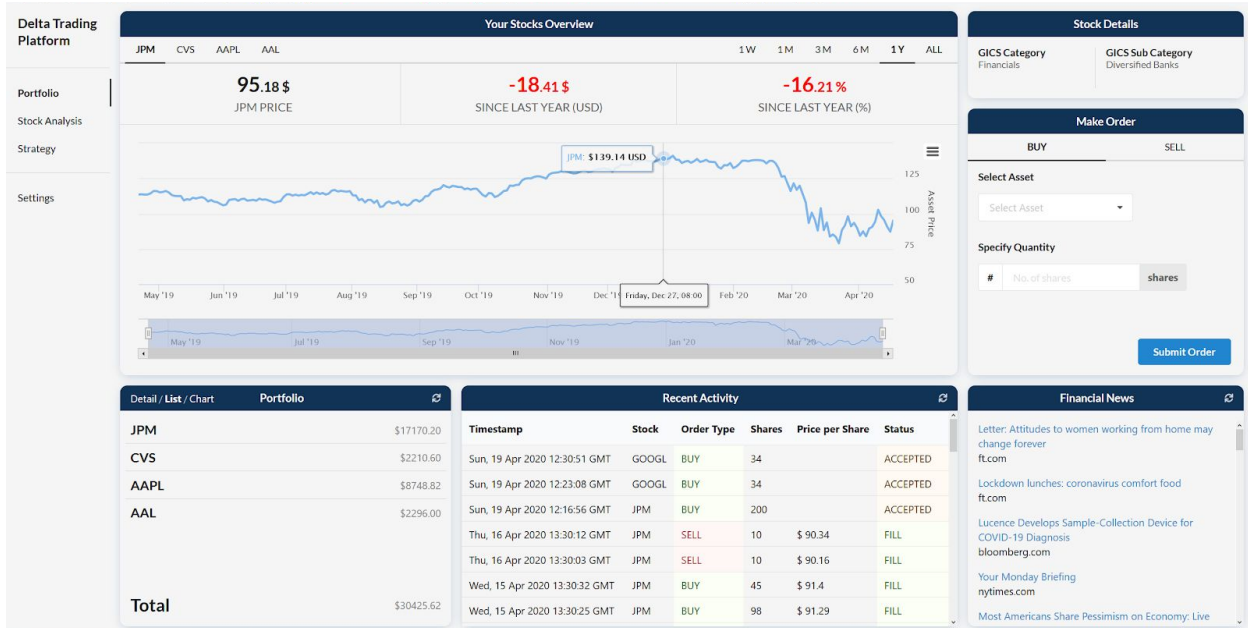


Figure 2. View of the Portfolio page

“Your Stocks Overview” provides an overview of the stocks performance since various timeframes and the current price for the stocks in your portfolio. Users can choose to view their stock performance since last week, last month, or since last year. Change in absolute and relative price are also reported in US Dollars (USD) and percentage (%) respectively. Furthermore, we have color coded the positive/negative profits - any positive change is displayed in green, while negative change in red; this is done to improve readability and overall UX.



Figure 3. View of “Your Stocks Overview” component. Users can choose the asset and timeframe. Performance of that stock in the chosen time frame is color-coded in green or red.

“Stock Details” simply provides the information of the category and subcategory according to Global Industry Classification Standard[9] for the asset chosen by the user at the moment of using the page.



Figure 4. View of “Your Stocks Overview” component

“Make Order” allows users to make their trade on the Portfolio page. Users can choose either buy or sell order, then pick the asset from the drop down list. A real-time quote price will be displayed once an asset is chosen, and then users can choose whether or not to make the trade.

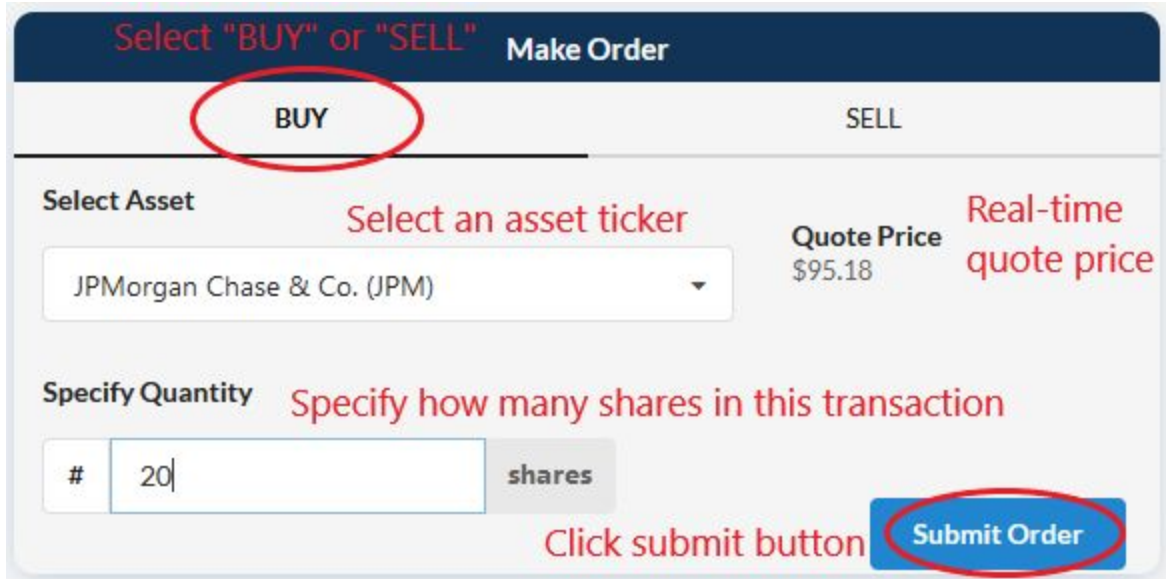


Figure 5. View of “Make Order” component

“Portfolio” renders three different views of the portfolio. The “List” view is the default view which shows the current market value of the asset in the portfolio and the total market value of the portfolio. The “Chart” view generates a pie chart which shows the market value proportion of each of the assets in the portfolio. The “Detail” view lists each the price, number of shares, the market value and the total profit for each asset in the portfolio. The user also has an option to liquidate the position by clicking the “cross” button, which upon clicking, opens a confirmation modal on whether the user wishes to liquidate the clicked asset.

Asset	Market Value
JPM	\$35792.00
GOOGL	\$83980.00
CVS	\$2114.00
AAPL	\$13782.75
AAL	\$2170.00
Total	\$137838.75

Figure 6. List view of “Portfolio” component, user can choose the view via top left corner, and the total portfolio amount is separated from the individual assets for clarification

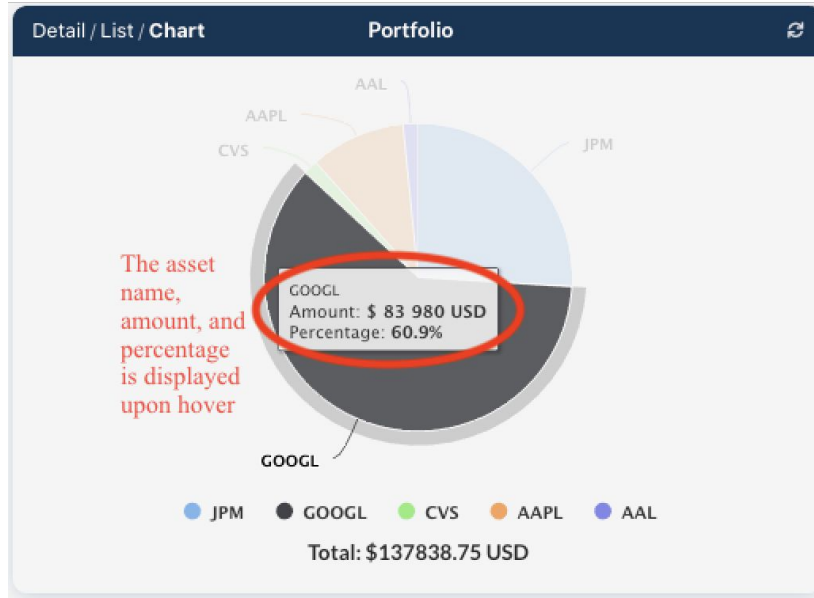


Figure 7. The chart view of the “Portfolio” component allows the user to get a quick view of his/her portfolio spread. On hovering over individual pie chart sections, the user can gain more detailed information

Stock	Price	Shares	Market Value	Total Profit	Liquidate
JPM	\$89.48	400	\$35792.00	-\$993.75 Liquidate asset	X
GOOGL	\$1235.00	68	\$83980.00	-\$2344.98	X
CVS	\$60.40	35	\$2114.00	+\$118.41	X
AAPL	\$270.25	51	\$13782.75	+\$182.84	X
AAL	\$10.85	200	\$2170.00	-\$770	X

Figure 8. The detail view of the “Portfolio component”. The profit is color-coded for readability, and the user can choose to liquidate asset with the X button, opening a “confirm liquidation” modal

“Recent Activity” provides a list of the trade orders made by the user and their strategy. We have color-coded the order type and order transaction status for better readability.

Recent Activity					
Timestamp	Stock	Order Type	Shares	Price per Share	Status
Mon, 20 Apr 2020 13:30:27 GMT	GS	SELL SHORT	20	\$ 178.54	FILL
Mon, 20 Apr 2020 13:30:18 GMT	GOOGL	BUY	34	\$ 1269.31	FILL
Mon, 20 Apr 2020 13:30:17 GMT	JPM	BUY	200	\$ 92.32	FILL
Mon, 20 Apr 2020 13:30:09 GMT	AAPL	BUY	20	\$ 278.19	FILL
Mon, 20 Apr 2020 13:30:06 GMT	GOOGL	BUY	34	\$ 1269.66	FILL
Mon, 20 Apr 2020 13:30:05 GMT	JPM	BUY	20	\$ 92.34	FILL
Thu, 16 Apr 2020 13:30:12 GMT	JPM	SELL	10	\$ 90.34	FILL

Figure 9. View of “Recent Activity” component

“Financial News” displays a list of the most recent financial news. The user can specify their desired news feed sources in the settings page, which will be discussed later. The title is an HTML link which redirects the user to the corresponding news article.

Financial News
Coronavirus pushes Virgin Australia into administration ft.com
KKR-backed Envision Healthcare hires restructuring advisers ft.com
Coronavirus should focus the minds of Asia's entrepreneurs ft.com
Japanese biotech seeks to turn 'brown gem' into gold ft.com
Lucrative offers for Asian start-ups stutter amid pandemic ft.com

Figure 10. View of “Financial News” component, each title is a hyperlink.

2.1.1.2.2 Stock Analysis

The stock analysis page contains: an ‘Assets’ chart. The page serves two views, the “General” and the “Analysis” view. When users first enter this page, “General” view will be served. In the “General” view, users could choose up to 10 assets to be displayed in the chart for general comparison on their performance. They could also evaluate their portfolio by clicking the “Import All Assets from Portfolio” button. All assets in the portfolio will then be chosen and displayed in the chart.



Figure 11. General view of the Stock Analysis page

The stock analytics page contains an “Import All Assets from Predictions” button, which when clicked, imports all the assets from the user’s portfolio, and renders the assets’ performance in the graph below. This feature was implemented so that the user can directly import all his/her assets, and cross-compare how each asset has performed.



Figure 12. "Import All Assets from Portfolio" allows user to check the portfolio assets' performance against each other

Users can also display the predictions from their own configurations to be displayed in this page for comparison purposes too. The predicted data are displayed in a dotted line for easier discernment from past market movement.

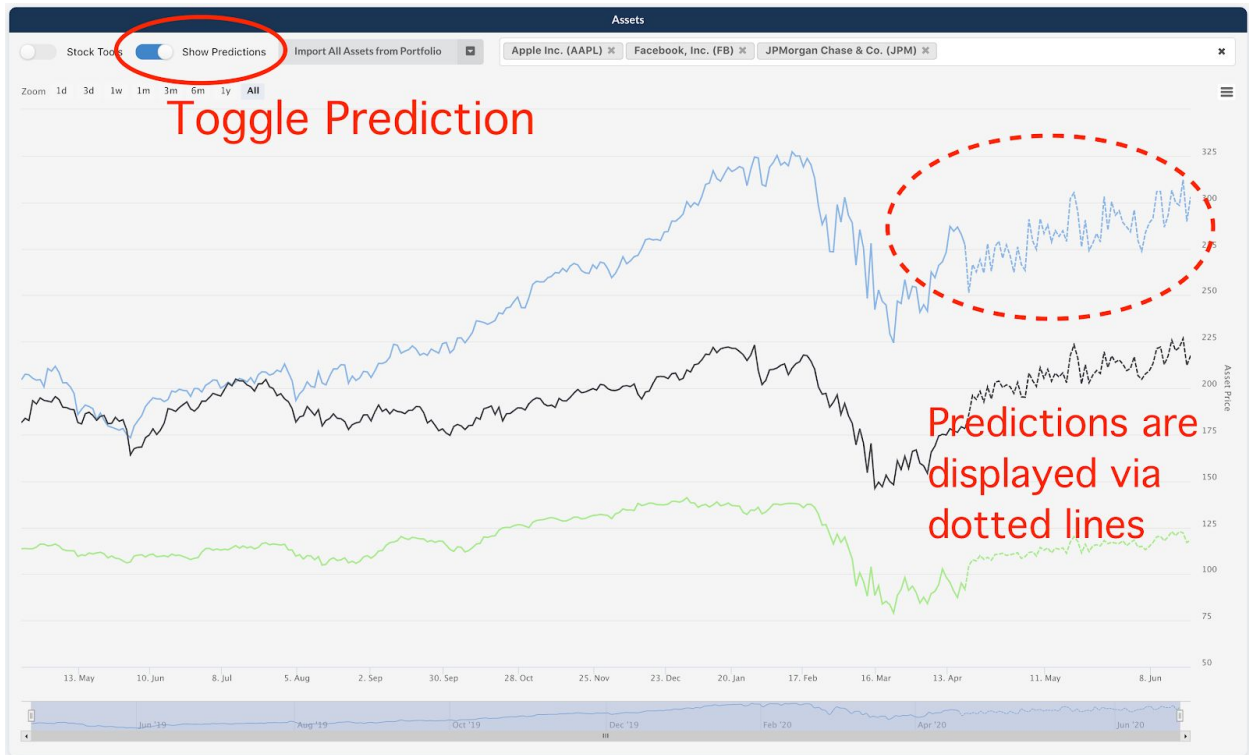


Figure 13. Analysis view of the Stock Analysis page showing predictions for multiple stocks

Another feature on this page is the “Show Tools” view, which opens the list of available stock tools provided to the user for technical analysis.



Figure 14. Analysis view of the Stock Analysis page. Here, we have chosen to analyze the AAPL asset. The top blue line indicates the price movement, while the bottom bars indicate volume per day.

In the “Analysis” view, users could add a couple of technical indicators to the chart for their own study. Same as the “General” view, users could also display predictions and the chart will include the data from the prediction in the technical indicators. Pure line charts may not be sufficient for users to work on their own analysis. To address this, we have implemented candlestick displays and OHLC (open, high, low, close) charts for further analysis.



Figure 15. Analysis view of the Stock Analysis page with candlesticks display

Our application supports a variety of technical indicators (Appendix 9.1) and allows the user to define their own parameters for the indicators.

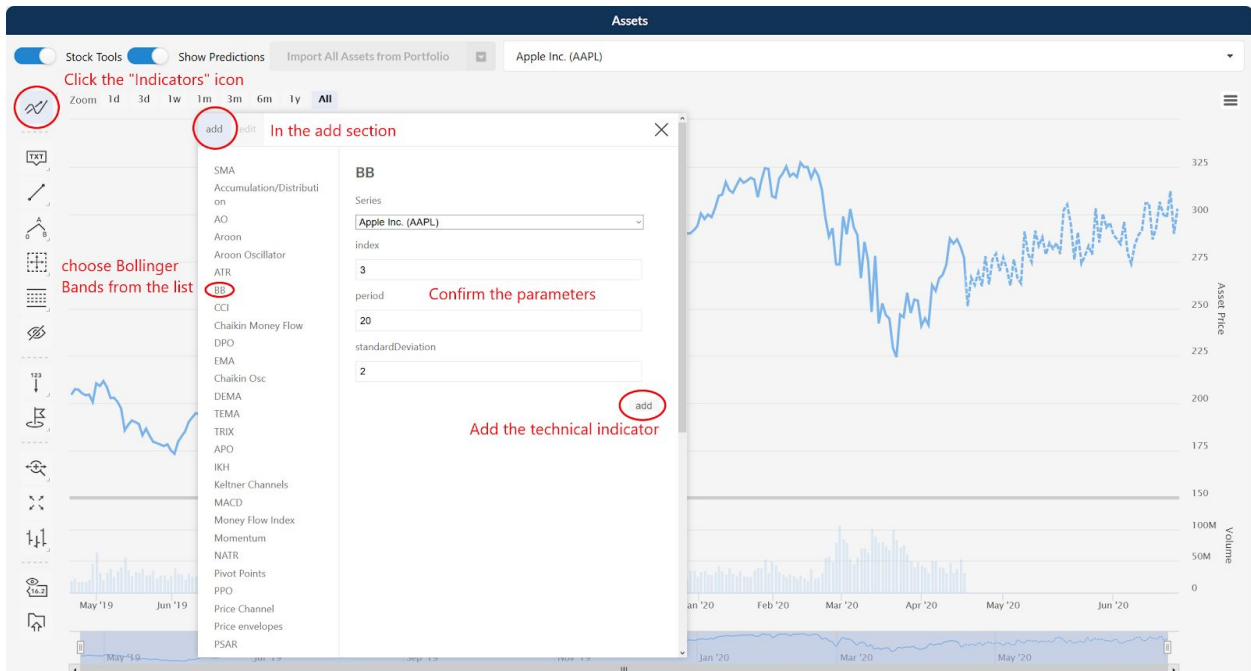


Figure 16. Analysis view of the Stock Analysis page trying to add Bollinger Bands

Technical Indicators will be drawn on the chart in real-time, and users could also move their cursor in order to check the value of the stock price and technical indicators from the tooltip.



Figure 17. Analysis view of the Stock Analysis page after adding Bollinger Bands

Our application allows users to put their own markers and make use of a multitude of tools which will facilitate their analysis process. Users could zoom in and out, work in fullscreen mode, add annotations for making notes and check the basic statistics in a self-defined window.



Figure 18. Analysis view of the Stock Analysis page with Bollinger Bands (BB), Volume Weighted Average Price (VWAP), Relative Strength Index (RSI) and Moving Average Convergence Divergence (MACD)

At the end, users could save their work and the data through clicking the accessibility icon. They could export the data to spreadsheets format or save an image for the notes made.



Figure 19. The accessibility button at the top right corner displays a list of options the user can render the graphical information as.

2.1.1.2.3 Strategy Page

The strategy page comprises two main blocks: the strategy list display block, and the add/edit strategy block.

Strategy Display - This component renders the list of strategies in a readable format. In the left corner, the user can choose to organize their list of strategies by: asset, sector, or due date (closer or further by clicking). In the bottom right corner is an add strategy button, which allows the user to add a strategy, thus emptying the add strategy component fields for a new strategy (discussed further below).

Furthermore, in the list view, the user can toggle the strategy on/off, as well as whether the strategy repeats. We put this feature here so that users from a “strategy list” overall point of view can turn the strategies on/off, and whether it repeats, without needing to click on the strategy then moving to the edit component to toggle these features.

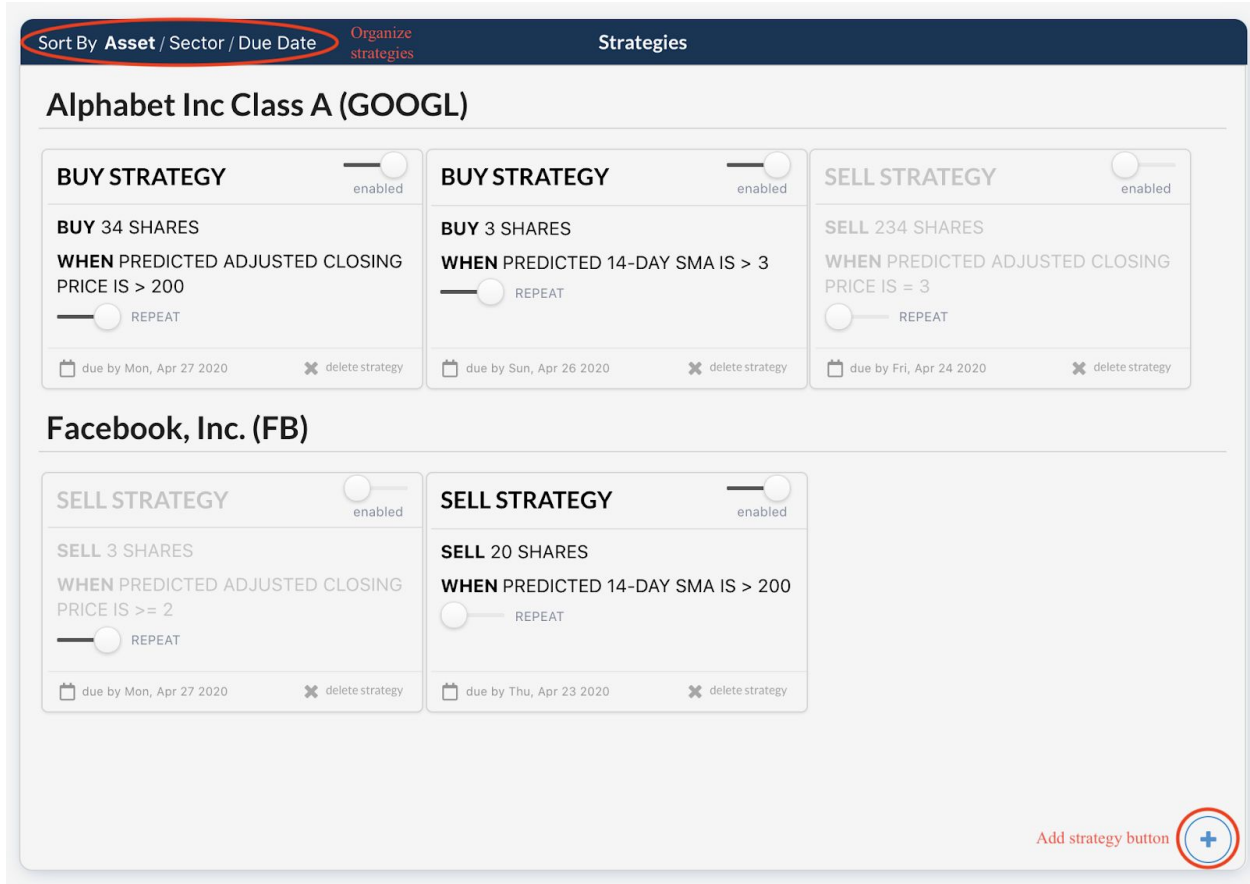


Figure 20. In the upper left corner, the user can choose how they wish to organize their strategies. The due date button if pressed multiple times, toggles between sorting by closest due date, and most future due dates. The add button empties the add/edit strategy component for a brand new strategy.

The overall goal of displaying the strategies card is inspired by IFTTT-like programming, where the user can read the condition & action in layman terms, and toggle the condition on/off with the click of a button.

Each “strategy card” is click-able and sets the strategy as the “active strategy”. The active strategy’s card will be underlined in blue, while the information can be edited in the edit component. Each card displays the essential information of the strategy: the asset to trade, whether it’s buy/sell, the action (e.g. “buy 34 shares”), the condition (e.g. “when predicted adjusted closing price is > 200”), the due date, and a “delete strategy” button. The body of the card (action, condition, and repeat), renders the strategy in a readable format, departing from the coding/programming interface one might see in other e-trading platforms.

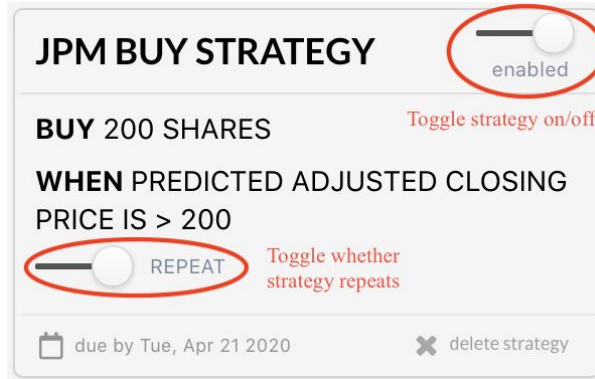


Figure 21. Strategy card. The user can toggle the strategy on/off, and repeat/non-repeat from this view. The essential information is presented in a compact, friendly way, and the condition & action are displayed in colloquial/layman terms to avoid confusion and easy UX

Enabled strategies have the words colored in black, while disabled strategies are faded to grey. For example, please refer to the strategy list pictured before (figure 20). This is designed so that the user can determine which strategies are enabled/disabled from the overall strategy list point of view.

Clicking the “delete strategy” button opens a confirmation modal, displaying a minimized strategy card (pictured below) to be deleted, and whether the user is sure to delete the selected strategy. This is done to minimize user errors.

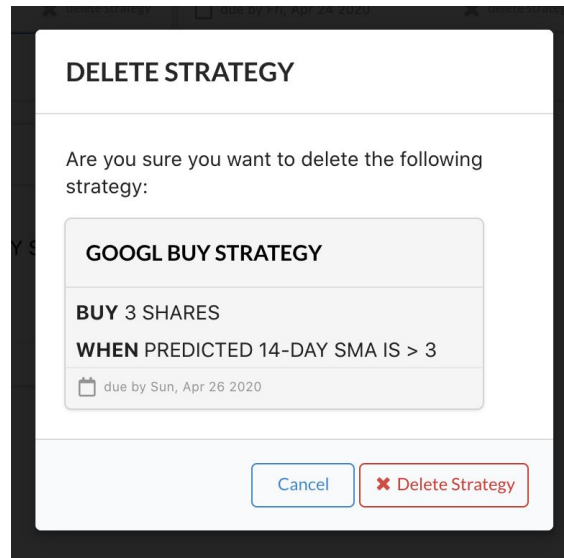


Figure 22. Delete strategy modal. Here, the site prompts the user for confirmation of deletion, and displays the essential information of the strategy to avoid deleting the wrong strategy.

Add/Edit Strategy - Moving on, the second component of the strategy page is the add/edit strategy component. This component can be accessed either by clicking on the component itself, clicking the ‘add strategy’ button, or setting a strategy card as active, which will fill the add/edit component with the active strategy’s information. The add/edit strategy component has 4 main

sections: the add/edit condition sections, add/edit action section, add/edit settings section, and lastly, the save changes/add strategy buttons. Each section has its corresponding fields.

If the user were to click on an existing strategy, the sections' headers would change correspondingly (e.g. "add condition" becomes "edit condition"), and the fields would be pre-filled with the existing strategy's information. Pictured below is an example of what the component would look like when editing:

The screenshot shows a web form titled "Add/Edit Strategy". It is divided into three main sections:

- EDIT CONDITION:** Contains a dropdown menu for "IF" with the value "Alphabet Inc Class A (GOOGL)", a dropdown menu for "14-day SMA", a dropdown menu for "IS" with the value ">", the word "THAN", and a text input field containing the number "3".
- EDIT ACTION:** Contains a dropdown menu for "BUY", a text input field containing the number "3", and the label "SHARES".
- EDIT SETTINGS:** Contains two toggle switches labeled "ENBALED" and "REPEAT", both of which are currently turned off. Below them is a text input field for "DUE DATE" containing the date "April 26, 2020".

At the bottom right of the form, there are two buttons: "Save Changes" (disabled) and "Delete Strategy" (active).

Figure 23. Editing an existing strategy; the fields are already pre-filled with the active strategy, and until any changes are made, the 'Save Changes' button will be disabled. The 'Delete Strategy' opens up the delete strategy modal as per mentioned above.

If the user selects an existing strategy, the buttons at the bottom are configured as 'Save Changes' and 'Delete Strategy'. The delete strategy button opens up the "delete strategy" modal mentioned before to reduce user mistakes, and the 'Save Changes' is disabled until the user has modified any of the fields. This feature is designed to reduce traffic between frontend and backend; the strategy edit can only be registered if the user has made any actual changes.

Upon edit, if the user were to go back and edit a field to an invalid value, the field will turn red and the user will be notified of the error underneath the field. Furthermore, upon any error, the 'Save Changes' button will be disabled.

The screenshot shows a form titled "Add/Edit Strategy" with three main sections: "EDIT CONDITION", "EDIT ACTION", and "EDIT SETTINGS".

- EDIT CONDITION:**
 - IF: Alphabet Inc Class A (GOOGL)
 - ML Adjusted Closing Price
 - IS: =
 - THAN: abc (highlighted in red with error message "Input must be a number")
- EDIT ACTION:**
 - SELL
 - 234
 - SHARES
- EDIT SETTINGS:**
 - ENBALED:
 - REPEAT:
 - DUE DATE: April 24, 2020

At the bottom of the form, the text "Disabled until all fields are valid" is shown in red. The "Save Changes" button is also highlighted in red, while the "Delete Strategy" button is not.

Figure 24. Editing a strategy: here, the user wishes to change the technical value but instead inputs a string “abc”. This is an invalid value for the technical value, so the field is highlighted in red, along with the error message.

Upon clicking the circular add strategy button from the strategy list display component, the Add/Edit Strategy component will reflect a blank strategy. Headers in each section will be changed to “Add” instead of “Edit”, and all the fields except the first will be disabled.

The screenshot shows a form titled "Add/Edit Strategy" with three main sections:

- ADD CONDITION:** Contains a dropdown menu for "IF" with "Asset" selected (highlighted in blue), a dropdown for "PREDICTION" with "Prediction" selected, a dropdown for "IS" with "Operator" selected, and a dropdown for "THAN" with "Val" selected.
- ADD ACTION:** Contains a dropdown for "Buy/Sell" and a text input field for "Amount" (circled in red), followed by "SHARES".
- ADD SETTINGS:** Contains two toggle switches for "ENBALED" and "REPEAT", and a text input field for "DUE DATE" with the value "April 21, 2020".

Red text annotations indicate that the "Asset" field is the active field, while other fields are disabled. A blue button labeled "Add New Strategy" is located at the bottom right.

Figure 25. The component when adding a new strategy. The active field, in blue, is currently 'asset', and until the 'asset' field is filled and valid, all the other fields will be invalid.

This component has many features to add to great user experience, firstly by ensuring correct inputs, without any missing fields or invalid values. Secondly, the process is designed to reflect how an user might add a strategy in layman's terms - set the condition first, then set the action, and finally fine tune the settings. The form is interlaced with english text, so when the user is adding a strategy, it mimics the process of filling in the blanks of a sentence, thus departing from the daunting interface of regular trading/programming languages.

Each field can be edited only if the previous fields are filled and valid. Until the user enters a valid value in the current active field, the later fields are disabled. The active field is highlighted in blue, and if any of the previous fields were to be edited invalid, the active field will be jumped to that edited field. An example is shown below:

The screenshot shows a web form titled "Add/Edit Strategy". It is divided into two main sections: "ADD CONDITION" and "ADD ACTION".

- ADD CONDITION:** This section is active. It contains:
 - IF:** A dropdown menu with "Apple Inc. (AAPL)" selected.
 - 14-day SMA:** A dropdown menu with "14-day SMA" selected.
 - IS:** A dropdown menu with ">=" selected.
 - THAN:** A text input field containing the letter "a". Below it is a red error message: "Input must be a number".
- ADD ACTION:** This section is disabled. It contains:
 - BUY:** A dropdown menu with "BUY" selected, circled in red.
 - Amount:** A text input field with "Amount" as a placeholder and "Required" below it, circled in red.
 - SHARES:** A label to the right of the "Amount" field.

A red message "Previously filled fields are now disabled" is positioned to the right of the "ADD ACTION" section.

Figure 26. Although the fields in 'Add Action' section are already filled, if the user were to go back and change a previous valid field's value to an invalid value, the invalid field would be red, and the later fields would be again disabled.

The image shows three examples of the "Amount" input field with different error messages:

- An empty field with the error message "Required".
- A field containing "-3.0" with the error message "Must be a positive number".
- A field containing "3.2" with the error message "Must be an integer".

In all examples, the field is highlighted in red and the error message is in red text.

Figure 27. Examples of invalid field error messages. The messages are added so that the user knows how to correct his/her input. In this case of the 'amount of shares' field, the field cannot be empty, must be a positive number, and must be an integer.

Furthermore, when the add/edit strategy component is adding a new strategy, the buttons at the bottom are changed to "Submit Strategy" only. This button is disabled until all the required fields are filled.

2.1.1.2.4 Settings Page

Users can go into the Settings page to configure the source of news they are receiving and the machine learning model parameters to be used in predictions. When users click the “Settings” section from the navigation bar, a modal will pop up and the following screen can be seen.



Figure 28. Settings modal

Users can click on the “News Sources” section and pick the source(s) they wanted to include in the “Financial News” component in the “Portfolio” page.



Figure 29. “News Sources” section in Settings modal

Users are required to configure the “Machine Learning Profiles” before they can make use of their own machine learning model to generate predictions. Assuming users' machine learning model files are already on the system (as the functionality to do file upload and download is out of our project scope). There are a maximum of 10 profiles users can configure in the current design. There are a limited number of choices available for machine learning framework and min max scaler interface (again, as a Proof of Concept project, if it works for a few sets of interface and frameworks, it could be applied to other popular ones too).

To configure a specific profile, users would have to specify the framework used to generate the machine learning model (Tensorflow Keras[13] in this case), the machine learning model file, min max scaler interface (scikit-learn[14] in this case), the min max scaler file, the input and output time length with dimension, a list of assets which this profile will be applied to, and the input parameters of the machine learning model (Adjusted Closing Price in this case).

▼ Machine Learning Profiles

Machine Learning Profile Profile 1 ▼

What machine learning framework did you use to build the model? Tensorflow - Keras ▼

Which machine learning model do you want to use to generate predictions? aapl_lstm_exact_pricing_60in_60out.tf ▼

Which min max scaler interface did you use? Scikit Learn ▼

Which min max scaler you would like to use? aapl_lstm_exact_pricing_60in_60out_mm_scaler.save ▼

Please specify your model input time length and dimension (e.g. day, minutes) 60 Day(s) ▼

Please specify your model output time length and dimension (e.g. day, minutes) 60 Day(s) ▼

Targeted Asset(s)
JPMorgan Chase & Co. (JPM) ✕ ▼
Apple Inc. (AAPL) ✕ Facebook, Inc. (FB) ✕

Number of Input Parameters for Machine Learning Model 1 ▼

Input Parameter 1 Adjusted Close ▼

Delete this profile or Save this profile

Figure 30. "Machine Learning Profile" section in Settings modal

After specifying all the parameters, users just have to click save this profile and a confirmation screen will be shown. They can click the "Confirm" button and finish the configurations.

Which min max scaler you would like to use? aapl_lstm_exact_pricing_60in_60out_mm_scaler.save ▼

! You are going to overwrite/update the profile 1

Please specify your model input time length and dimension (e.g. day, minutes) 60 Day(s) ▼

Are you sure you want to overwrite/update it? This action cannot be undone!

Please specify your model output time length and dimension (e.g. day, minutes) 60 Day(s) ▼

✕ Cancel ✔ Confirm

Figure 31. Confirmation page after configuring "Machine Learning Profile"

2.1.1.3 Designing the Backend

The backend is a server that handles all the requests from the frontend interface and returns the data required. The backend server is responsible for collecting requests in JSON⁸ format through RESTful Protocol in the appropriate endpoints⁹ from the frontend.

The screenshot shows a REST client interface for a POST request to the endpoint `/ts/eod_daily`. The request body is a JSON object with the following structure:

```

{
  "tickers": [
    "amzn",
    "aapl"
  ],
  "startDate": "2019-09-20",
  "endDate": "2019-09-20",
  "columns": [
    "adjClose",
    "adjVolume"
  ]
}
    
```

Figure 32. Example of a POST Request schema for historical market data endpoint

The screenshot shows the server response for the POST request. The response is a 200 status code with the following JSON body:

```

{
  "columns": [
    "date",
    "adjClose",
    "adjVolume"
  ],
  "data": {
    "AAPL": [
      [
        1568937600000,
        217.0856798616,
        57977094
      ]
    ],
    "AMZN": [
      [
        1568937600000,
        1794.16,
        5555839
      ]
    ]
  ]
}
    
```

Figure 32. The corresponding API response

The backend server also serves another purpose of providing the necessary system features and functionalities for the following features: 1) predictive insights, 2) intuitive and customizable automated trading strategies and 3) systematic management of portfolios. Furthermore, it keeps track of the monitoring metrics from different models proposed by different users, to fit the

⁸ JavaScript Object Notation

⁹ Endpoints documentation attached at the Appendix

users' needs more by using their own machine learning model instead of a centralized and generic model. The system architecture is also able to digest the commands from the users in a graphical way and convert it into actual trading parameters in their trading strategies. We will explain the backend task flows below:

Delta Trading Backend API ^{1.0}
{ Base URL: /api }
http://fyp.martinwongtk.com:8080/api/swagger.json

the backend api endpoints for frontend

Server Status status of the backend services

- POST /server/fetch_market_data
- GET /server/health
- GET /server/test

News API main page news headlines

- POST /news/news_headline
- POST /news/news_headline_offset

Time Series API time series data points

- POST /ts/eod_daily
- POST /ts/ts_pred

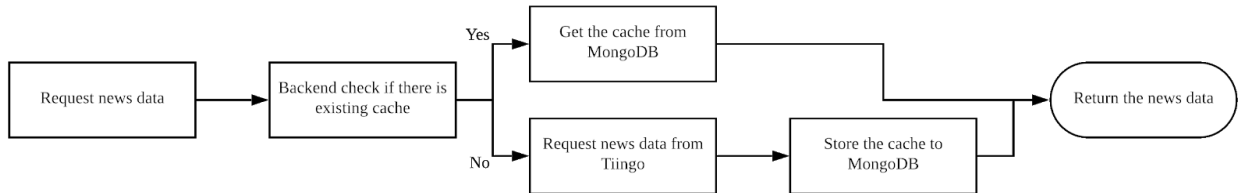
User API manage all users information / positions / actions / settings

- GET /user/ifttt
- PUT /user/ifttt
- GET /user/orders
- POST /user/orders
- GET /user/orders_history
- GET /user/positions
- DELETE /user/positions
- POST /user/quote
- GET /user/user_configs
- DELETE /user/user_configs_machine_learning_profiles
- PUT /user/user_configs_machine_learning_profiles
- PUT /user/user_configs_news_sources

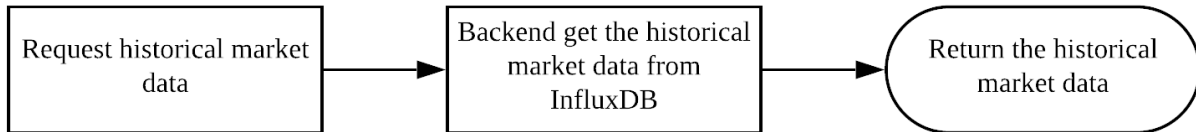
Figure 33. Backend server API documentation page using Swagger

1. Fetch news data

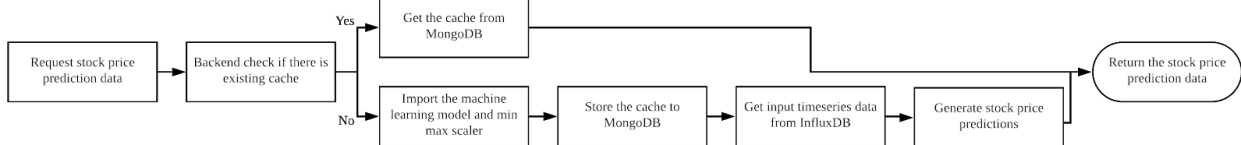
When the “News Feed” component is mounted on to the frontend’s domain tree, a fetch command is triggered to the backend. The backend system then checks whether there is existing news feed in the MongoDB cache. If so, the system directly returns the news data. Otherwise, another fetch is made to the third-party server Tiingo, cached in MongoDB, and finally returns the news data.



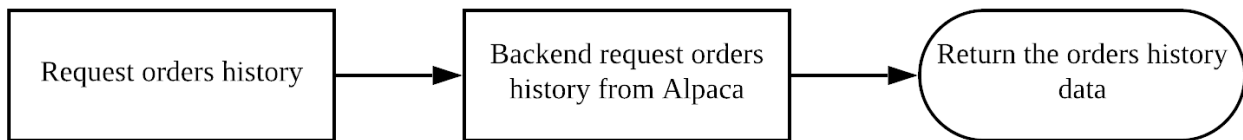
2. Fetch historical market data



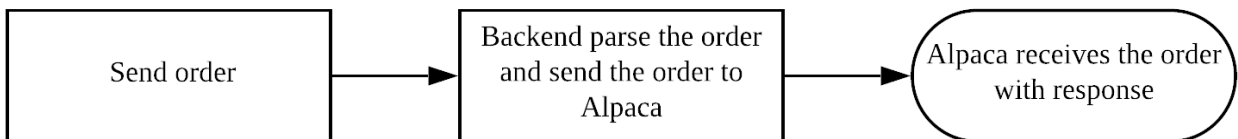
3. Fetch stock price prediction data



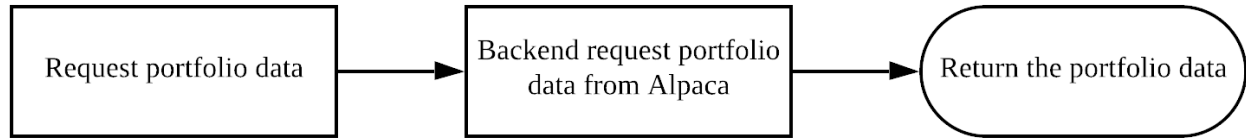
4. Fetch users’ orders history



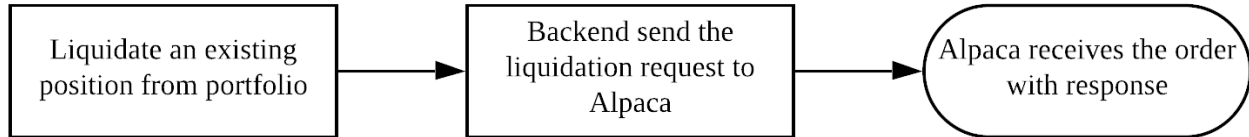
5. Make trades



6. Fetch portfolio data



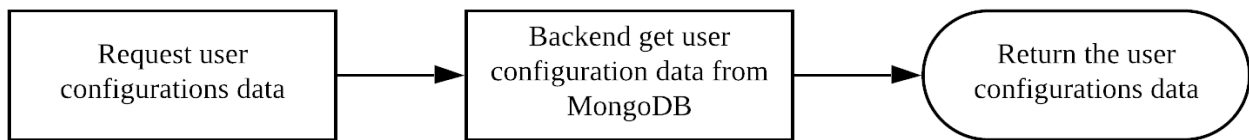
7. Liquidate an asset in portfolio



8. Fetch real time quote price



9. Fetch users' configurations data



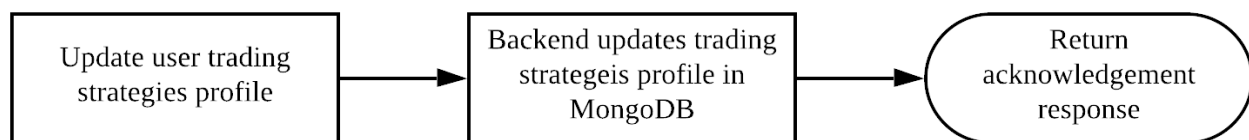
10. Update users' configurations



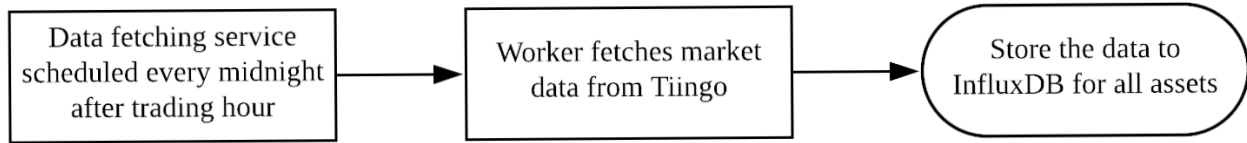
11. Fetch users' trading strategies profile



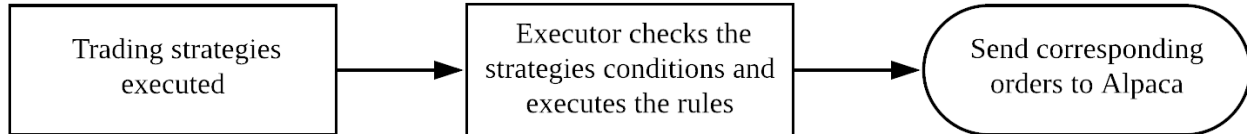
12. Update users' trading strategies profile



14. Fetch market data at scheduled time



15. Execute users' trading strategies



2.1.1.4 Designing the Data Pipeline

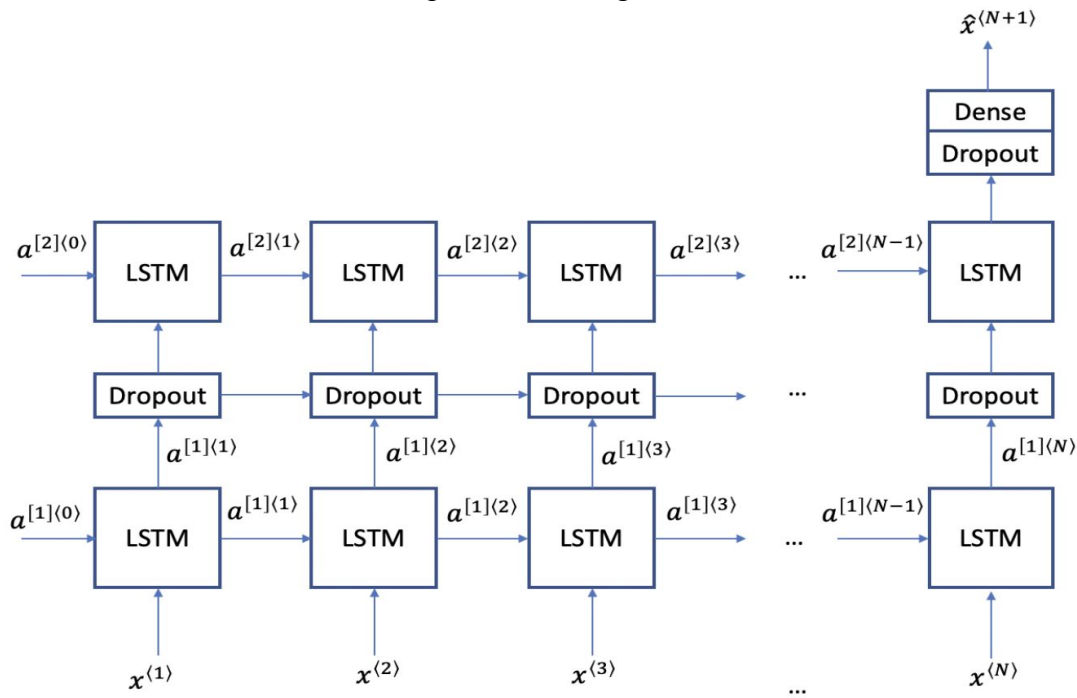
A data pipeline is essential in our system to ensure all the features are working. Since all of these features rely on the input from the market data and news data. For example, the trading strategies are based on the historical positions and predictive insights using past data. That is why a persistent and robust data pipeline is essential to our system so as to 1) fetch the data from our data provider, 2) store and retrieve data in our high performance database and 3) stream to all the services modules for our users such as trading strategies executions, machine learning model prediction and data presentation in the frontend.

2.1.2 Designing the ML Prediction Model

To design the ML prediction model, we have broken the process down into three main steps:

1. Designing the model - Our initial objective for this project was to develop a Long Short-Term Memory (LSTM) ML model that would be worthy of retrofitting in the system for financial gain. We took a step by step approach to implementing this using different types of technical indicators. We tested and played around with the following methods and tested the effectiveness of our models using error calculation metrics [7].
 - a. Last value - This method assumes the prediction to be the value of the last observed closing value. It serves as a computation based cost-effective benchmark to compare with other sophisticated models that we implement.
 - b. Simple Moving Averages - This method calculates the prediction based on the mean of the previous N observed closing values. We tune the hyperparameter N based on the root mean squared error calculations. This method tends to be heavily localized on the market situation in the past days and is a slow indicator of a market that is undergoing a change in direction of price movement.
 - c. Linear Regression - This method calculates the prediction based on the linear relationship between the dependent variables and independent variables. We use the past N days to train the model and predict based on the linear relationship that is defined. This method doesn't give us a trend and misses out on the quadratic relationship between various indicators.

- d. XGBoost - Extreme Gradient Boosting was used to minimize the errors through iterative training of the boosted tree algorithms. We tune the hyperparameters such as differences between the high and low prices for the day and the difference between the closing and opening prices and perform feature engineering on the adjusted closing prices. This method is prone to overfitting and requires extensive scaling of different datasets as it has vanishing gradient problems.
- e. LSTM - Our final and main design aspect was to develop a LSTM that would be able to give state of the art predictions on the datasets. We did multiple iterations on different parameters and tried different layers. Following is the design of the LSTM model that we plan to be using in our model.



LSTM network architecture.

Figure 34. LSTM network architecture diagram. Here our model has two LSTM layers, a dropout layer and a dense layer in our design.

2. Evaluating and choosing the data source - We first picked a data source and analyzed the credibility of the chosen data source. To gauge the data source's credibility, we plan to compare its stock data against the data from a credible news source (such as Thomson Reuters or Bloomberg). The comparison will help us in determining the validity of the data. We have reviewed several online sources, and have decided to use Tiingo¹⁰ as our tentative data source; it is the most optimal data provider in both quality and pricing. Although Yahoo Finance allows us to download intraday data free of cost, it may not be robust enough for us to display as part of our product.
3. Designing the data storage structure and choosing the web scraping methodology - We will scrape the news data and market data programmatically using microservices

¹⁰ A financial research platform dedicated to creating innovative financial tools for all, <https://www.tiingo.com/>.

developed in Python. The microservices will periodically scrape market data and news data from our data provider, Tiingo.

2.2 Implementation

In this project, we made use of a variety of toolsets. For IDEs, we used JetBrains IntelliJ IDEA and VSCode. We wrote our frontend web interface using ReactJS with Typescript support and Python 3.7 for our backend web server. We've also utilized GitLab for version control.

2.2.1 Implementing the System

We will follow the backend designs formulated earlier to implement the system and the data pipeline. Our strategy for implementing the system was to first get the market data from our data provider to build the data pipeline, then continuously refine our system architecture and schema according to the market data schema and user requirements.

2.2.1.1 System Architecture

We have already set up the infrastructure for the system on Digital Ocean¹¹. We allocate a linux server running CoreOS to host all our applications and databases. Given the immense memory usage for the Time Series Database, performance tuning was done to enhance the service availability. In order to achieve modularity, we containerized different modules (frontend user interface, Flask backend server, MongoDB and InfluxDB) in our systems by leveraging Docker¹² and Application Container¹³ technology. We have also set up Flask on Web Server Gateway Interface (WSGI) and made it available to users with a web browser. We hosted our code on Gitlab¹⁴ which could help us in source code versioning and set up a CI/CD¹⁵ pipeline for automated checking and deployment. We have fully completed the implementation of the system architecture at this stage.

2.2.1.2 The Platform

We originally wrote the code in HTML/CSS/JavaScript with the React framework. For styling, we used a combination of CSS and the Semantic UI React library. We designed the frontend architecture with React framework, for its speed and scalability. Namely, its ability to change data without completely reloading the page. Furthermore, the coding practices of React framework were simple, as seen with its virtual document object model, which is similar to that

¹¹ DigitalOcean provides developers cloud services that help to deploy and scale applications that run simultaneously on multiple computers.

¹² Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers.

¹³ Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels.

¹⁴ GitLab is a web-based DevOps lifecycle tool that provides a Git-repository manager

¹⁵ Continuous Integration and Continuous Delivery

of object-oriented programming, and its single-way data flow. We eventually made the shift from JavaScript to TypeScript, as TypeScript’s structural nature made the code easier to read and debug.

To further facilitate the strict data flow of the React framework, we also implemented Redux, which is a library built on the idea of the flux architectural pattern, in which components on the bottom of the domain tree cannot directly modify the data of those above in the tree.

In order to visualize the time series data and distribution of assets in a portfolio, we made use of Highchart¹⁶ and Highstock JavaScript library to draw informative charts, allow the use of a variety of annotations in Stock Analysis view, and generate technical indicators in real-time.

Portfolio Page

The portfolio page, or the home page, first began as a “general monitoring” page consisting of 2 components: the live news feed, and a graph of market data. However, after we researched the UIs of other platforms, we realized the home page should instead place emphasis on the portfolio performance aspect, so we moved the general market data graph to the stock analysis page.

Below, we will explain the technical functionalities of the portfolio page:

1. Rendering “Your Stocks Overview” component -



Figure 35. The “Your Stocks Overview” is conceptually divided into three rows

The top row of “Your Stocks Overview” component has two menus to choose from: the asset menu, and the timeframe menu. With these 2 user inputs in place, a fetch request with three parameters: the chosen asset, chosen timeframe, and desired indicators as an array (open, close, ... etc.) is sent to the backend. If successful, a response is received in the form of an array with length dependent upon the chosen timeframe. Each element in the response array is an array

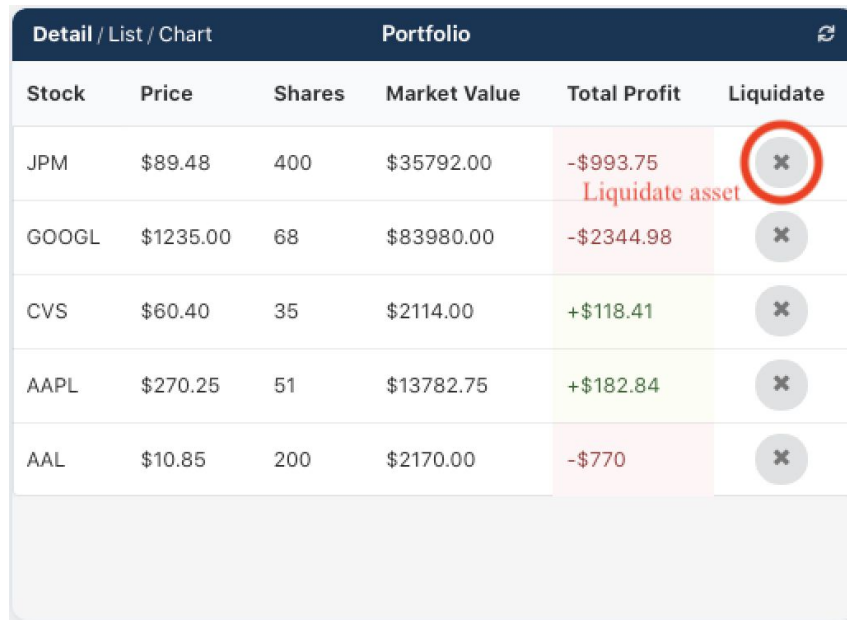
¹⁶ Highcharts JS is a JavaScript charting library based on SVG.

itself, indicating the values corresponding to the desired indicators specified in the fetch parameter of a particular day. An example of this fetch request can be seen in figure 33 in the backend design portion of this report.

Once the response is successfully received, the response is loaded into the “middle row report” and “bottom row graph”. Both of these components are responsible for rendering pertinent information from the fetch response. The middle row renders the current stock price, and the difference in the asset’s stock price for the given timeframe, both in monetary amount and percentage. The bottom row graph displays the price movement across the given timeframe.

As with all requests made in our project, we utilized the JavaScript built-in promise object to handle the asynchronous nature of AJAX communication. The promise object allows for better code readability, as well as better flow of handling asynchronous logic and errors. If an error were to occur, the error is either logged onto the console, or displayed as a popup box on the frontend to notify the user.

2. Liquidating assets



Detail / List / Chart		Portfolio				
Stock	Price	Shares	Market Value	Total Profit	Liquidate	
JPM	\$89.48	400	\$35792.00	-\$993.75 Liquidate asset	<input checked="" type="checkbox"/>	
GOOGL	\$1235.00	68	\$83980.00	-\$2344.98	<input type="checkbox"/>	
CVS	\$60.40	35	\$2114.00	+\$118.41	<input type="checkbox"/>	
AAPL	\$270.25	51	\$13782.75	+\$182.84	<input type="checkbox"/>	
AAL	\$10.85	200	\$2170.00	-\$770	<input type="checkbox"/>	

Figure 36. The option to liquidate stocks

In the frontend, when the user chooses on the liquidate stocks option by clicking on the ‘X’ as shown in the figure above, a POST request is called on which initiates the change in the overall makeup to the portfolio. In the backend, this results in the calling of a function that appends the list of portfolios maintained using the Alpaca Trading API and updates the change on their servers. Once this process is completed, a success or failure is returned on the frontend.

3. Making orders

Figure 37. View of “Make Order” component

When the user presses the submit button, a POST method call is made by the frontend which leads to a function in the backend adding the BUY/SELL request to the list of orders that are maintained with the Alpaca Trading API. Initially, the orders are submitted as ACCEPTED OR REJECTED in the ‘Recent Activity’ tab which is replaced by the FILL or PARTIAL FILL command during trading hours.

Stock Analysis Page

The main functionalities and their implementations of the stock analysis page are detailed below:

1. Choosing the assets

On the frontend, the “Choose asset” selection bar/dropdown component is implemented via the Semantic UI React library. If the user were to click the ‘Import All from Portfolio’, the list of assets kept in the user portfolio is retrieved from the Redux store and set as default value for the dropdown component.

Frontend logic is implemented to limit the asset selection to one asset when the ‘stock tools’ feature is toggled. This is done so to avoid overcrowding when there are several indicators for multiple assets.

2. Rendering the assets’ past performance

Much like the “bottom row graph” implementation discussed in the portfolio page implementation section, the stock analysis graph is rendered via the Highstocks component from the Highstocks library. The list of selected assets is rendered into an array of strings, and

formatted as one of the parameters for the fetch request. The exact fetch method and URL is listed under the “Time Series API” portion in figure 33.

On successful retrieval of market data for the given assets list, the data is formatted to the specifications of the Highstock component [15]. The Highstocks component then renders asset(s) price movement across a timeframe on the graph.

3. Rendering the predictions

Once the user toggles the ‘Show Prediction’ option on the strategy page, a fetch request with parameters list is sent to the backend. Once the fetch response is received, we receive a response with the format of a time-series data

4. Stock tools

The stock tools is an optional set of tools provided by the Highstocks library for better analysis. The user can draw custom markers, such lines, shapes, or labels, and data can be rendered in the form of lines/candlesticks/OHLC (open high low closing) through the ‘type change’ tool.

The Highstocks component also provides an ‘accessibility’ feature (pictured in figure 19), which allows the user to print the chart, download the chart as an image file, or render it as csv/xls files. These features’ logic is handled by the Highstocks library.

Strategy Page

After designing the strategy page interface, we focused on connecting the strategies to the user’s portfolio so that changes can be reflected in the Portfolio page in real time. There are three key features that we implemented which allow for the seamless maintenance of information and strategies across the whole system

1. Adding strategy tasks

After a user uses our IFTTT based Add/Edit strategy tab to create new strategies, these are stored as a JSON object list in our backend. To minimize the risk of accidental error in creating strategies, we use error checking and limit the users to a drop-down list of pre-checked data items such as assets, prediction indicators and relational operators.

2. Editing strategy tasks

Users have the option to choose one of the pre-existing strategies from among the list of Strategies and change the parameters for how the strategy is implemented. We provide the users with a real-time access to the stored strategies in our servers and allow them to mutate options based on their preference

3. Scheduling strategy tasks

Once the user has created a strategy, we use python's Advanced Python Scheduler¹⁷ (APScheduler) which allows us to run strategy jobs in the background. We set the trigger for this action to execute the due strategy every night at midnight and then disable them if they are not to be repeated in the future.

2.2.1.3 Data Pipeline

The data pipeline is designed such that the system is able to download data from data providers, store it into our high performance database, InfluxDB, and be retrieved by the users for analysis. We were able to automate this process through using a background task scheduler APScheduler along with cron¹⁸. The data fetching feature is automated to fetch data from Tiingo every midnight after trading hours. While implementing this feature, we initially thought of using Celery¹⁹, a Distributed Task Queue, as our task scheduler. It provides better software application support and functionalities to the system. Later on however, we found its requirement of setting up a message queue system like that of Redis to be overly complex. After evaluation of Celery's complexity and the project schedule, we decided not to implement this solution as it would consume a lot of time and it is not the main focus of our project.

2.2.2 Implementing the ML Prediction Model

As previously discussed in our design stage, we took a step by step approach to implement the various methods for stock price prediction. The following graph shows the accuracy for the predictions generated using the last value, moving average, linear regression, XGBoost and LSTM models methods. We observed that the LSTM gave the highest accuracy, and the lowest root mean squared error (the index for measuring errors). In the future, we plan to implement the ARIMA and triple smoothing methods as mentioned in the paper [7].

¹⁷ <https://github.com/agronholm/apscheduler>

¹⁸ <https://en.wikipedia.org/wiki/Cron>

¹⁹ <http://www.celeryproject.org/>

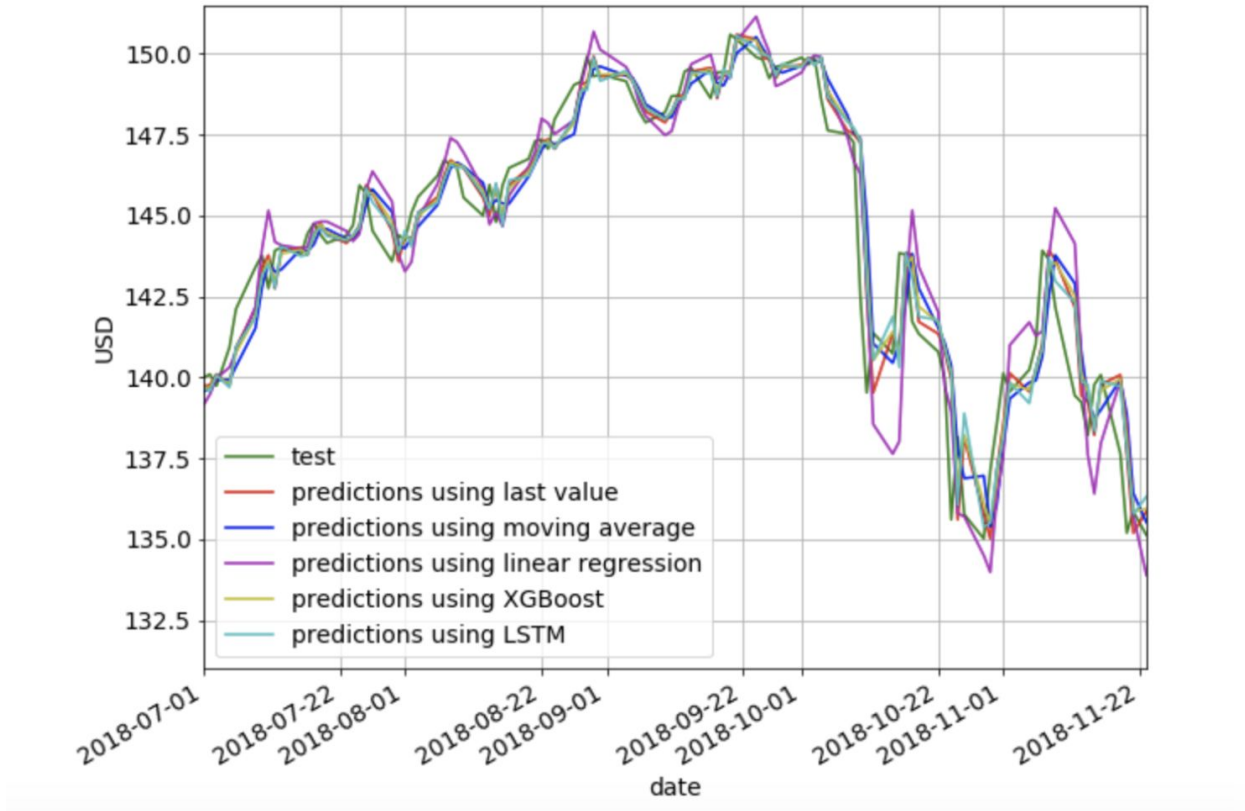


Figure 38. Graphical Representation of the accuracy of the various prediction method on the VTI ETF dataset based on the article [7]

After realizing the result above, we decided to implement a stock price prediction model using LSTM. We have implemented two machine learning models:

1. LSTM model for Apple Inc. (AAPL) Stock, input 60 days, output 60 days

Layer	Hyperparameters
LSTM 1	Units = 60, Return sequence = True, Input shape = (60,1)
Dropout 1	Rate = 0.15
LSTM 2	Units = 60, Return sequence = True
Dropout 2	Rate = 0.15
LSTM 3	Units = 60, Return sequence = True
Dropout 3	Rate = 0.15
LSTM 4	Units = 60
Dropout 4	Rate = 0.15
Dense 1	Units = 60
Function	Hyperparameters
Optimizer	Adam
Loss	MSE (Mean Squared Error)
Epochs	500
Batch size	1024

Layer (type)	Output Shape	Param #
lstm_8 (LSTM)	(None, 60, 40)	6720
dropout_8 (Dropout)	(None, 60, 40)	0
lstm_9 (LSTM)	(None, 60, 40)	12960
dropout_9 (Dropout)	(None, 60, 40)	0
lstm_10 (LSTM)	(None, 60, 40)	12960
dropout_10 (Dropout)	(None, 60, 40)	0
lstm_11 (LSTM)	(None, 40)	12960
dropout_11 (Dropout)	(None, 40)	0
dense_2 (Dense)	(None, 60)	2460
Total params: 48,060		
Trainable params: 48,060		
Non-trainable params: 0		

Figure 39. Model 1 summary extracted from Tensorflow

2. LSTM model for Apple Inc. (AAPL) Stock, input 60 days, output 1 day

Layer	Hyperparameters
LSTM 1	Units = 60, Return sequence = True, Input shape = (60,1)
Dropout 1	Rate = 0.15
LSTM 2	Units = 60, Return sequence = True
Dropout 2	Rate = 0.15
LSTM 3	Units = 60, Return sequence = True
Dropout 3	Rate = 0.15
LSTM 4	Units = 60
Dropout 4	Rate = 0.15
Dense 1	Units = 1
Function	Hyperparameters
Optimizer	Adam
Loss	MSE (Mean Squared Error)
Epochs	500
Batch size	1024

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 60, 40)	6720
dropout (Dropout)	(None, 60, 40)	0
lstm_1 (LSTM)	(None, 60, 40)	12960
dropout_1 (Dropout)	(None, 60, 40)	0
lstm_2 (LSTM)	(None, 60, 40)	12960
dropout_2 (Dropout)	(None, 60, 40)	0
lstm_3 (LSTM)	(None, 40)	12960
dropout_3 (Dropout)	(None, 40)	0
dense (Dense)	(None, 1)	41
=====		
Total params: 45,641		
Trainable params: 45,641		
Non-trainable params: 0		

Figure 40. Model 2 summary extracted from Tensorflow

2.3 Testing

2.3.1 User Experience

We conducted a poll on the user experience satisfaction on the various pages. We received 34 responses, and below are the results.

We asked the users to rate the effectiveness of the portfolio page on communicating the user's portfolio performance. Below is the spread of ratings on a scale of 1 - 10.

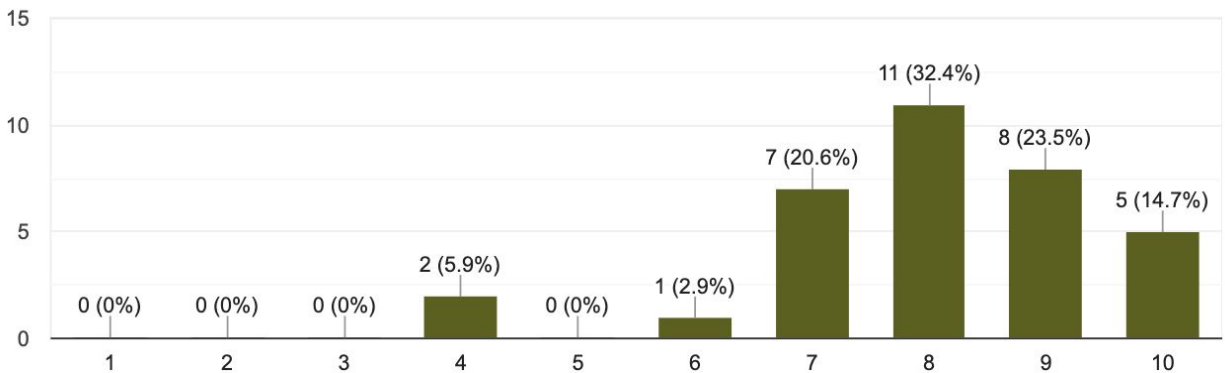


Figure 41. Spread of user feedback on the portfolio page's effectiveness, with the X axis as the rating, and Y as the number of users who gave that score

There was a non-required question placed at the end of the poll, asking users for any additional feedback on what features they enjoyed, or what features they thought were missing, or could be improved upon. Of the 9 responses, we received 1 response pertaining to the portfolio page. The response suggested a "personal value" located clearly on the page, which is available on other e-trading platforms.

We suspect the 2 reviews that gave a marking of 4 out of 10 on the effectiveness of the portfolio page could be attributed to the lack of a clear signifier on the overall portfolio's performance. While we did include such performance in the "portfolio makeup" component on the portfolio page, perhaps it would have benefited the user greater if it had its own individual component, or be placed somewhere more obvious.

Moving on, the second question asked the users to review the stock analysis page's importance (namely, the ML generated predictions) in the overall process of trading. We asked the users to place the importance of this feature when trading on a scale of 1 - 10, with 1 being "Not essential", to 10 being "Very essential/Would use it every time". The spread of ratings can be seen on the graph below:

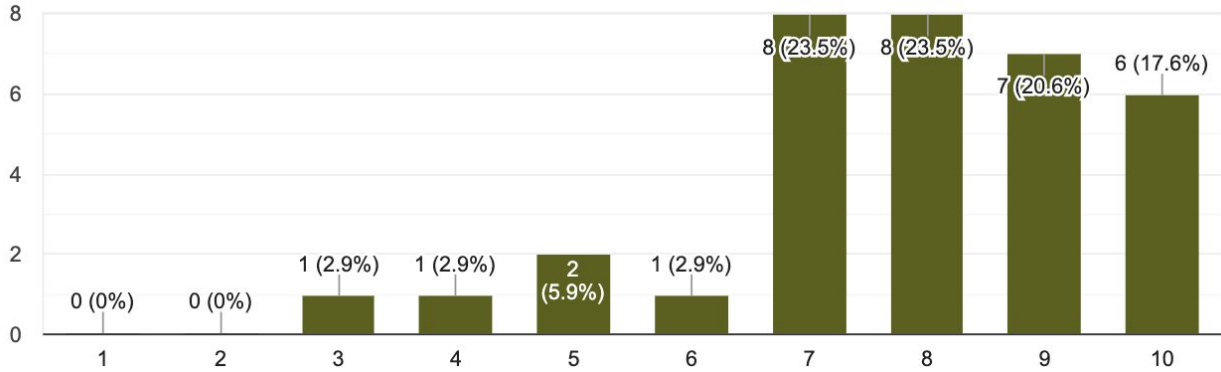


Figure 42. Spread of user feedback on the stock analysis page's importance

As seen by the graph, the stock analysis/prediction page was perceived generally as an essential tool in stock trading. We received 1 written feedback on the stock analysis page: “[I would like to see] more transparency/explanation of the ML parts of the product for better understanding and trust from the layman.”

The third question we asked the users was to rate the usefulness of the user settings modal, which allows the user to customize news sources, and different machine learning profiles. Users were asked to rate the usefulness on a scale of 1- 10, with 1 being “Not useful/Don’t understand it”, to 10 being “Very useful”. Below is a graph representing the spread of ratings:

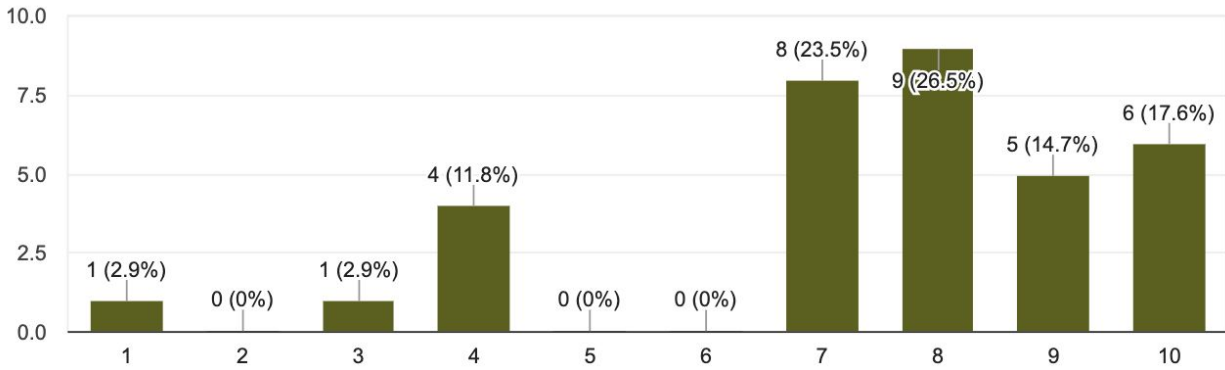


Figure 43. Spread of user feedback on the user settings page's usefulness

A minority of users (6 users, 17.65%) of users found the settings page not useful/not understandable, with the rest of the users rating it 7 or above. We expected this page to be the least well received, as it employed the most ML/financial jargon out of the pages, which users from a non-machine learning background would be rather clueless on how to utilize this page.

Next, we asked the users to rate the usefulness of the strategy page on a scale of 1 - 10, with 1 being “Not effective/would not use it”, and 10 being “Very effective/would use it”. The spread of ratings can be seen below:

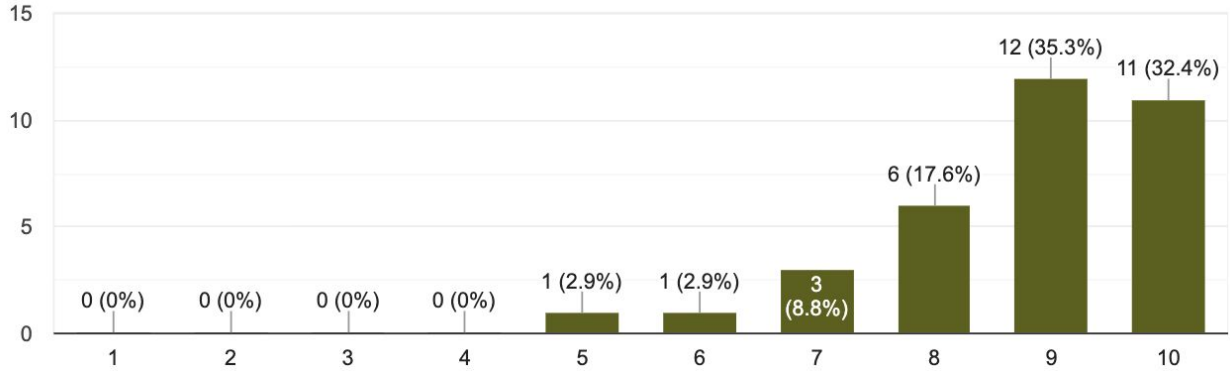


Figure 44. Spread of user feedback on the user strategy page's usefulness

This was the most well received page in our application, with only 5 users (14.71%) rating it 7 or below. One user commented he/she greatly appreciated the execution of this page.

Lastly, we polled the users on the overall user experience/the usefulness of the overall application. On a scale of 1-10, 28 users (82.35%) gave the application a rating of 8 or above, with the rest of the users rating it a score of 6 or 7. The spread can be seen below:

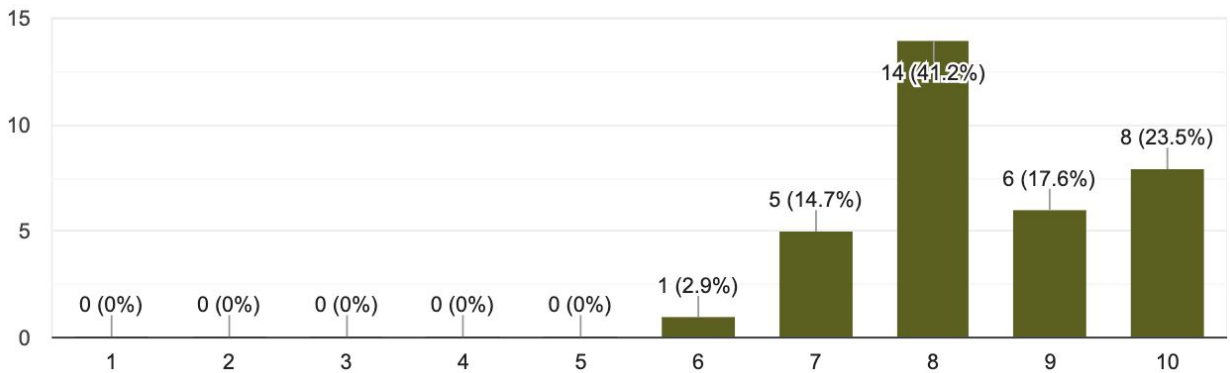


Figure 45. Spread of user feedback on the overall user experience

Overall, our application was generally well received, but none of the pages had a majority of users giving it a 10/10 on effectiveness/usefulness. This may be due to the varying degrees of user knowledge on machine learning/financial trading, as well as the lack of instructions on how to fully utilize the platform, leaving users to have to make connections between the components by themselves.

2.3.2 Browser Loading Test

To measure the network analysis and show how well optimised our platform is, we perform a browser loading test using the developer tools provided by chrome. We will focus on the HTTP requests and the time required to get the data-packets from the digital ocean server. Prerequisites to performing this include disabling cache to prevent loading from browser cache.

Stage 1 DOMContentLoaded [0 to 3160ms] - Our test results showed that the platform takes 3s to load up scripting and stylesheet elements for our web browser. This value is standard with the industry benchmark values as confirmed by a test which yielded 2500ms for Google and 4000ms for Facebook

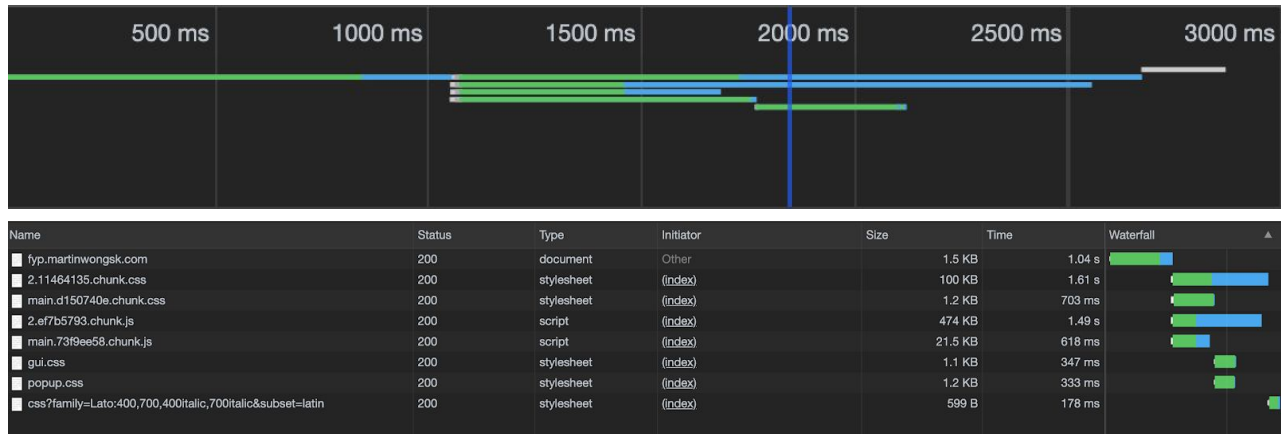


Figure 46. DOMContentLoaded time series graph for loading of our website

Stage 2 Load [3160 to 7840ms] - Our platform shows up to be quite robust in the test for network latency by allowing HTTP based fetch for multiple pages and various data heavy values such as (time-series, graphing, user-configs, ifttt-strategies) in 4.5s as shown by the graph below.

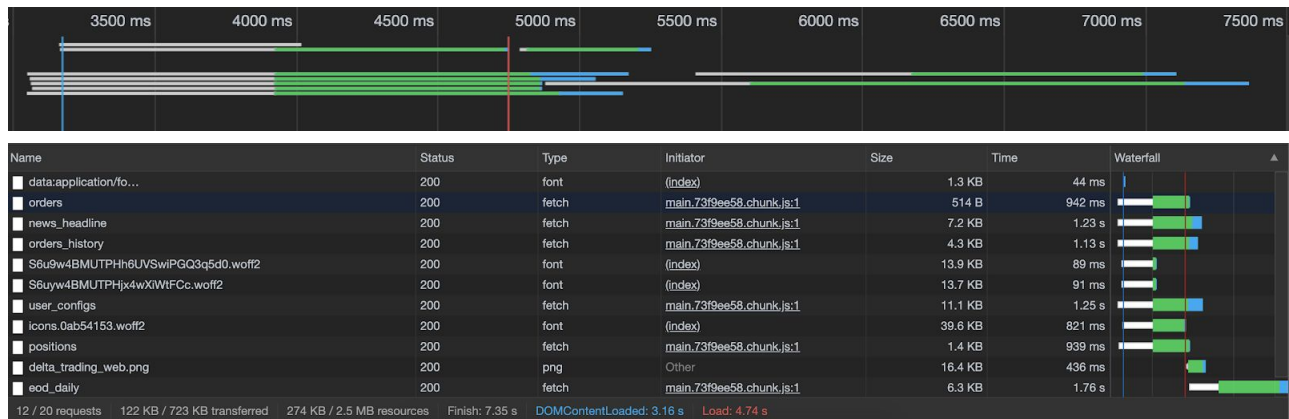


Figure 47. Load time series graph for loading of our assets and UX toolkits

Overall our platform performs 20 successful requests with 2.5 MB resources transferred in 7.35s which provides users with a nominal loading time that maintains a great user-experience on using the application.

2.3.3 Further Testing

Regression testing for every change - There would be changes to our pipeline, and to ensure our code works correctly most of the time, we will conduct regression testing for every change, and adopt the Test-Driven Development (TDD) practice.

Unit testing for most scripts - We will do unit testing on our pipeline. In order to easily conduct the unit testing, we will try to modularize and follow the “Single Responsibility” design pattern. We will write test cases for the majority of the modules, and make sure it has a high code coverage.

Use-Case testing - To perform Use-Case testing, we selected certain parameters for a user to test out if the specified functionality of our platform aligns with the functionality that they can access. All the test cases resulted in success and the detailed analysis can be found in Appendix 10 of our report.

2.4 Evaluation

After a successful completion of the criterias mentioned by us in our overview, we would do the following evaluation of how we fared -

1. We were successful in providing our users a holistic all-in-one platform where they could perform a strategic analysis of their assets and formulate trading strategies. Our Stock Analysis page and Strategy page are dedicated to fulfilling this criteria and provide an integrated approach to solving this major roadblock for our client.
2. We were partially successful in providing our users various stock prediction models based on technical indicators. We provide users an opportunity to incorporate various self-made models that can further be incorporated with our pre-existing set of models. However, one limitation is the advanced technical know-how required by the users to be able to achieve this feat.
3. We were successful in allowing our users to incorporate the ML predictions as signals in their trading algorithms. This is shown by our strategy page where we allow users to choose ‘ML Adjusted Closing Price’ as a key indicator for comparison while generating strategies.
4. We were partially successful in allowing our users with an IF/THEN-like trading strategy page where they could integrate all essential features of our platform. We lag behind in making the GUI more aligned with the IF/THEN standard and providing users a more detailed access to predictions through displaying detailed parameters while they work with the IF/THEN part of strategy.
5. We were generally successful in displaying the predictions in a clear, graphical way. Of the 34 responses we received, roughly 85% of users rated 8 or above out of 10 on how likely they would use the stock analysis page as part of their trading. 82% of users rated their overall user experience as 8 or above, and see themselves using Delta Trading Platform regularly.

Overall, we believe that we were successful in making this platform align with the goals and criterias that we had set for ourselves. We further believe that the platform stands out to be a model solution for the trading needs of our users.

3. Discussion

3.1 Shift of Focus from Research to Application

Our initial approach to this project was a research-based approach on ML prediction of stock price movement. However, as we realized the task of ML prediction was out of our time scope, we shifted the project to a software engineer exercise on what an e-trading platform might look like if it were integrated with ML generated predictions. Our primary focus became replicating the standard features of an e-trading platform, generating predictions, displaying the predictions in a clear readable format, and providing strategy implementation based on predictions.

3.2 Challenges in Stock Price Prediction

As we spent more time on developing an application with a great UX, we had to sacrifice the development of a highly-accurate machine learning model. Users should be aware that technical analysis never guarantees price movement; it merely indicates a possible trend. The signals generated from market movement often lag, as it often may not be the cause of price movement, but rather the effects of movement beyond numbers and statistics.

Similar concerns are shared with sentiment analysis. Pulling news feeds and giving it a sentiment score never guarantees 100% prediction accuracy. Firstly, as mentioned above, news headlines are very prone to lag, as more often than not it merely responds to market movement, rather than be the cause of market movement. Furthermore, inherent limitations in sentiment analysis/natural language processing prevent perfect predictions; at the current state of natural language processing, it is difficult for the analyzer to detect sarcasm, or understand personal opinions from mere headlines. With these limitations in mind, the current state of our project's predictions should be taken as suggested trends, rather than 100% accurate predictions.

3.3 Challenges in Application Development

On the application side, we achieved our goal in developing a clean, user friendly GUI to communicate the user's portfolio performance, stock price predictions, as well as the user's list of strategies. The main challenge we encountered during this process was always keeping the user in mind when developing the application. An example can be seen in the stock analysis page: if users wished to compare how past predictions perform against real market data, we would have had to store the past predictions in another dataset and determine a way to display the comparison. While this feature would be essential for users with more finance background, we decided this was not essential to the everyday user and focused on other features instead.

The user feedback indicates that, for a minority of users, the predictions were not viewed as essential/trustworthy. We suspect the lower rated responses to be from a lack of understanding/trust from machine generated predictions. For the average consumer without a strong background in ML, he/she may not realize the workings of ML and/or the limitations in the task of stock-price prediction, thus a sense of distrust. Perhaps a tutorial page on the fundamental workings & inherent limitations of ML predictions would have been useful, or a graphical display of how accurate our models have performed against real data. Such features could be implemented for the user to build a visceral understanding of how reliable our predictive models are, and thus higher usage of the predictions/stock analysis page.

An additional feature we would have implemented in retrospect in the strategy page would be a component with a search bar, displaying the chosen asset's price movement in a graph, along with its generated predictions. When making strategies with predictions as part of the condition, the user could have benefited from a component on the strategy page which would allow the user to search an asset, and see the predictions on display. While the user can still replicate this process by switching back to the 'stock analysis' page/opening a new 'stock analysis' tab to make form a strategy, the user experience would have greatly benefited from placing the essential features in the correct spots.

The strategy page was successful in providing a friendly GUI with an intuitive approach to algorithmic trading. However, within our given timescope, we could only implement simple strategies, such as "if predicted adjusted closing price is greater than value X, then sell Y amount of shares." However, in standard trading practices, often times users wish to program nuanced instructions, or have more specifications on the buying/selling of stocks (e.g. an user might only want to sell 20% of shares if predicted adjusted closing price crosses a value of 200, and sell 40% of the same shares if the same prediction indicator crosses a value of 300). Currently, if we wished to implement this feature we would have to make the "add strategy" component from the frontend more dynamic, and implement more detailed logic in the backend.

4. Conclusion

On the machine learning side of things, we learned that the task of prediction is an extremely multi-faceted, wide topic beyond our initial understandings. Technical and sentiment analysis, while proven to be somewhat accurate, more often than not are merely reflections of trends rather than indicators of trends.

In fact, our initial focus was trying to research the methodology to generate stock price prediction based on historical market data and news data. However, the difficulty and complexity to understand and derive a new algorithm which could possibly lead to a new optimization problem is beyond our capability and time frame. So we later on only attempted to generate simple stock price generating models as examples. Though we eventually shifted our focus, we learnt the process of developing a new machine learning algorithm from reading a number of research papers. We have also learnt different machine learning techniques from literature surveys, which were instrumental to our learning.

As our project had to be completed in a relatively short time frame, we were more preoccupied with fleshing out the essential frontend components, rather than improving the overall UX. This flaw in system design extends to the backend, where we could have implemented more efficient data fetching/storing strategies to improve loading time. In retrospect, we would have conducted further investigation into regular stock trading practices to have a better understanding of a trader's UX flow, and consequently better backend data flow.

5. References

- [1] “Under the Hood - Robin Hood” <https://blog.robinhood.com/> (accessed 17-Oct-2019).
- [2] “Bloomberg Professional Services” Bloomberg Blog <https://www.bloomberg.com/professional/blog/> (accessed 17-Oct-2019).
- [3] “Trading on NinjaTrader using NinjaScript” <https://ninjatrade.com/support/helpGuides/nt8/?ninjaSCRIPT.htm> (accessed 20-Apr-2020).
- [4] “Investment Knowledge Basics” Learn Investing using E*trade <https://us.etrade.com/knowledge/investing-basics> (accessed 17-Oct-2019).
- [5] “Trading Platform at TD Ameritrade” <https://www.tdameritrade.com/tools-and-platforms/web-platform/features.page> (accessed 20-Apr-2020).
- [6] A. Singh. “*Predicting the Stock Market Using Machine Learning and Deep Learning.*” *Analytics Vidhya.* <https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningn-d-deep-learning-techniques-python/> (accessed 17 Oct, 2019).
- [7] “Machine Learning Techniques applied to Stock Price Prediction” Towards Data Science. <https://towardsdatascience.com/machine-learning-techniques-applied-to-stock-price-prediction-6c1994da8001>. (accessed: 17-Oct-2019).
- [8] A. Edell. “*I spent 20 minutes trying to predict the stock market with AI — these are my results.*” *Hacker Noon.* <https://hackernoon.com/i-spent-20-minutes-trying-to-predict-the-stock-market-with-ai-these-are-my-results-59d48c7a388a> (accessed 17 Oct, 2019).
- [9] R. Fuhmann. “Stock Exchanges Around The World.” Investopedia.com. <https://www.investopedia.com/financial-edge/1212/stock-exchanges-around-the-world.aspx> (accessed 17 Oct, 2019).
- [10] D. Caplinger. “*The 10 Biggest Blue Chip Stocks,*” *The Motley Fool.* <https://www.fool.com/investing/the-10-biggest-blue-chip-stocks.aspx> (accessed 17 Oct, 2019).
- [11] Joshi, Kalyani & N, Bharathi & Rao, Jyothi. (2016). *Stock Trend Prediction Using News Sentiment Analysis.* International Journal of Computer Science and Information Technology. 8. 67-76. 10.5121/ijcsit.2016.8306.

- [12] “The Stanford NLP Group,” The Stanford Natural Language Processing Group. <https://nlp.stanford.edu/pubs/>. (accessed: 17-Oct-2019).
- [13] “Tensorflow Keras documentation” TensorFlow Core Keras <https://www.tensorflow.org/guide/keras> (accessed 20-Apr-2020).
- [14] “Scikit-learn documentation” User guide - Scikit-learn https://scikit-learn.org/stable/user_guide.html (accessed 20-Apr-2020).
- [15] “High Stocks API” <https://api.highcharts.com/highstock/chart> (accessed 20-Apr-2020).

6. Appendix A - Meeting Minutes

6.1 Minutes of the 1st meeting

Date - 15th September 2019

Time - 4:30 PM

Location - Prof Rossiter's Office

Members present - Prof Rossiter, Izen, Martin, Tanay

1. Approval of minutes

As this is the first in-person meeting after finalizing our team, there were no past meeting minutes to approve.

2. Discussion items

- We have to decide if we want to switch from the pre-decided trading simulator system or create something more of a research based version
- Possible non-technical indicators for Stock Price Prediction using Sentiment/word analysis on news/social media
 - Given a certain field to predict, the “predictor” would give prediction rates on “rise” or “drop”
- Other possible applications/ directions of our final product
 - Templates for the incorporation of company
 - Streamline the process the approach professionals
 - Teaches students about trading
 - “Automatic Headhunter” for businesses -- Tinder for businessmen
 - Client Risk Profile Automation
 - SMEs cash flow, industry trend & operations management tool
 - Risk assessment for mental health email

3. Goals for the coming week

Learn about Data Mining applications

Analysis (AI/ML)

Final Delivered Product

Think of possible setbacks and prepare backups for those

4. Meeting adjournment and next meeting

- The meeting was adjourned at 5:30 PM.
- The next meeting will be on 23/10/2019 at 3:30 PM HKT

6.2 Minutes of the 2nd meeting

Date - 23rd October 2019

Time - 3:30 PM HKT

Location - Prof Rossiter's Office

Members present - Prof Rossiter, Izen, Martin, Tanay

1. Approval of minutes

All members approved the past meeting minutes.

2. Report on progress

- All members finished the learning items discussed last time and came up with novel ideas discussed below.

3. Discussion items

3.1. We decided to train a stock predictor using information from within the sector (e.g. A change in Yahoo stocks would be a good indicator of a change in Google stocks.) To go even further, we can use information across different sectors to predict (e.g. automobile industry is strongly correlated to the performance of oil companies)

3.2. We discussed who our core user would be

- Discussion led us to find that retail clients and researchers would serve as good clients. Our product would be of effective use to them.
- Our main goal should be something tangible, but not necessarily a commercial product. We can still go towards research direction, but the final "program" does not have to be polished.

3.3 We discussed possible machine learning models that could be used well with the time-series stock data to predict prices.

4. Goals for the coming sprint

- Figure out how to crawl data from different social media (proposed: bloomberg - *need to pay*, twitter, facebook).
- Push the code for the crawler to github (https://github.com/bedman3/2019_FYP_Stock_Prediction)
- Post images on the data crawled in this document
- Do exploratory research on various machine learning models and frameworks that can be used for stock prediction

5. Meeting adjournment and next meeting

- The meeting was adjourned at 4:30 PM.
- The next meeting will be on 10/11/2019 at 4:30 PM HKT

6.3 Minutes of the 3rd meeting

Date - 10th November 2019

Time - 4:30 PM HKT

Location - Prof Rossiter's Office

Members present - Prof Rossiter, Izen, Martin, Tanay

1. Approval of minutes

All members approved the past meeting minutes.

2. Report on progress

- Finished first model for sentiment analysis using the Stanford NLP library.
- Scraped data from the Reuters terminal using a python script.
- Finished the first LSTM model that predicts stock models with benchmarked accuracy.

3. Discussion items

3.1. The data available is only for 10 years and has historical gaps in the time-frame needing much cleaning and pre-processing. Discussed possible shift to using Bloomberg for its better and voluminous data pool.

3.2. Discussed the need for back-testing to see if our machine learning model is merely following the past data. Dividing the dataframes into chunks to prepare and perform mini-batch algorithms

3.3 We further discussed other possible machine learning models that could be used well with the time-series stock data to predict prices.

4. Goals for the coming sprint

- Find use case for the sentiment analysis and tie it with the stock in a uniform and real-time method
- Perform tests and evaluation on the machine learning model to get better predictions and to test if we are not overfitting

5. Meeting adjournment and next meeting

- The meeting was adjourned at 4:30 PM.
- Future meeting dates will be set based on the situation in Hong Kong and our availability during the midterm season.

6.4 Minutes of the 4th meeting

Date - 29th January 2020

Time - 10:30 AM HKT

Location - Skype Call

Members present - Izen, Martin, Tanay

1. Approval of minutes

All members approved the past meeting minutes.

2. Report on progress

- We convened after the winter break and did a recap of where we had left off before the holidays

3. Discussion items

3.1. Selecting a source for getting stock data that is both economical and reliable in the long run

- We shortlisted <https://www.tiingo.com/> as one of the most feasible options based on budget and the quality of data.

3.2. Shift our code base from GitHub to GitLab as it provides asynchronous CI/CD (Continuous Integration/ Continuous Delivery) pipeline

- Martin raised the point that we should shift to GitLab which provides us with an easier way to check the changes that we deploy multiple times a day without having to take huge pains on our ends. It will help us track, test and evaluate the changes we make.

3.3 We need to update Prof Rossiter to set a meeting either in person or through Skype/ Zoom call

4. Goals for the coming sprint

- Send Prof Rossiter an email to discuss the progress and ask advice about possible future steps
- Need to set the structure of the backend and the frontend. We need to figure out the various API tools that go with the dashboard.
- Figure out the structure of the progress report as the deadline is approaching soon

5. Meeting adjournment and next meeting

- The meeting was adjourned at 12:00 PM.
- The next meeting will be on 01/02/2020 at 10:30 PM HKT

6.5 Minutes of the 5th meeting

Date - 1st February 2020

Time - 10:30 AM HKT

Location - Skype Call

Members present - Izen, Martin, Tanay

1. Approval of minutes

All members approved the past meeting minutes.

2. Report on progress

- Izen and Martin worked on removing errors in the frontend and pushed the current version to our testing environment
- Izen worked on removing the bugs in the frontend News API section
- Martin compiled research and did comparative analysis on the HighStocks API for our frontend dashboard
- Martin and Tanay worked on finalizing the influxdb storage environment

3. Discussion items

3.1. Primary discussion on finding a suitable name for the trading platform.

- We decided that our current project title isn't interesting to grasp user attention
- Came up with clearer objectives about what our system does to draw us closer to a better project title
- "An intelligent all-in-one platform for all your trading needs, featuring: customized ML stock price prediction, IFTTT automated trading, and portfolio rebalancing."

3.2. Finalizing the frontend dashboard design

- After receiving feedback from Prof Rossiter in the last meeting about having a clearer and well defined structure, we came up with the newer design for the frontend

4. Goals for the coming sprint (7/2/2020 - 12/2/2020)

- Compile and standardize meeting minutes for discussion with the communication tutor
- Complete frontend HighStocks API and fix issues with the landing dashboard pages.
- Work on the report structure to show to Prof Rossiter during the meeting on 12th February.

5. Meeting adjournment and next meeting

- The meeting was adjourned at 11:30 AM.
- The next meeting will be on 12/02/2020 at 12:30 PM HKT with Prof David Rossiter.
- Zoom Meeting invite has been sent by Izen.

6.6 Minutes of the 6th meeting

Date - 6th February 2020

Time - 10:30 AM HKT

Location - Zoom Video Call

Members present - *Prof David Rossiter*, Izen, Martin, Tanay

1. Approval of minutes

All members approved the past meeting minutes.

2. Report on progress

- As this was our first meeting after the winter holidays, we did a recap and reiteration of our project objectives for Prof Rossiter.

3. Discussion items

- Prof Rossiter suggested that we change our project title as it doesn't grasp user attention
- Suggestions on the frontend structure to make it more lucrative and useful for our clients
 - Adding a portfolio page along with the features for portfolio rebalancing
 - Adding and highlighting the machine learning model features as that is the technologically advanced part of our project.
- Prof Rossiter requested us to present to him a rough outline of our progress report for us to get some feedback about it

4. Goals for the coming meeting

- Individually think of useful ideas that can be consolidated for our project
- Regroup and discuss what can be done and discuss in the meeting tomorrow

5. Meeting adjournment and next meeting

- The meeting was adjourned at 11:30 AM.
- We will meet at 10:30 am tomorrow (7th February) to discuss and prepare for the next sprint.

6.7 Minutes of the 7th meeting

Date - 7th February 2020

Time - 10:30 AM HKT

Location - Skype Call

Members present - Izen, Martin, Tanay

1. Approval of minutes

All members approved the past meeting minutes.

2. Report on progress

- Izen and Martin worked on removing errors in the frontend and pushed the current version to our testing environment
- Izen worked on removing the bugs in the frontend News API section
- Martin compiled research and did comparative analysis on the HighStocks API for our frontend dashboard
- Martin and Tanay worked on finalizing the influxdb storage environment

3. Discussion items

3.1. Primary discussion on finding a suitable name for the trading platform.

- We decided that our current project title isn't interesting to grasp user attention
- Came up with clearer objectives about what our system does to draw us closer to a better project title
- "An intelligent all-in-one platform for all your trading needs, featuring: customized ML stock price prediction, IFTTT automated trading, and portfolio rebalancing."

3.2. Finalizing the frontend dashboard design

- After receiving feedback from Prof Rossiter in the last meeting about having a clearer and well defined structure, we came up with the newer design for the frontend

4. Goals for the coming sprint (7/2/2020 - 12/2/2020)

- Compile and standardize meeting minutes for discussion with the communication tutor
- Complete frontend HighStocks API and fix issues with the landing dashboard pages.
- Work on the report structure to show to Prof Rossiter during the meeting on 12th February.

5. Meeting adjournment and next meeting

- The meeting was adjourned at 11:30 AM.
- The next meeting will be on 12/02/2020 at 12:30 PM HKT with Prof David Rossiter.
- Zoom Meeting invite has been sent by Izen.

6.8 Minutes of the 8th meeting

Date - 12th February 2020

Time - 12:30 PM HKT

Location - Zoom Call

Members present - Prof Rossiter, Izen, Martin, Tanay

1. Approval of minutes

All members approved the past meeting minutes.

2. Report on progress

- Izen walked us through the introduction, title of our project. We explained to Prof Rossiter how our project would be a all-in-one platform would serve a bigger purpose for our clients
- There are different levels of our project, where level 0 is the historical data, level 1 is the prediction model, level 2 is the customizable ML model and level 3 would be a GUI based

3. Discussion items

- Discussion on IFTTT and how this concept can be used to make our system work with
- Martin explained the concept of using a modularized Machine Learning model
- Discussion on technical indicator & specificity of using ML models as future indicators for our project.

4. Goals for the coming sprint (12/2/2020 - 16/2/2020)

- Submit the first draft of the report by 16/02/2020
- Provide a working prediction and an architecture that supports ML model modularization
- Early goal is do tech analysis such as Simple Moving Average, integrate into frontend
- Send an email requesting a meeting to Prof Rossiter

5. Meeting adjournment and next meeting

- The meeting was adjourned at 1:00 PM.
- The next meeting will be on 16/02/2020 at 12:30 PM HKT

6.9 Minutes of the 9th meeting

Date - 27th February 2020

Time - 12:30 PM HKT

Location - Zoom Call

Members present - Ted (Communication Tutor), Izen, Tanay,

1. Approval of minutes

All members approved the past meeting minutes. Since Ted is an external member, he was not shown or asked to approve previous meeting minutes.

2. Report on progress

- Received feedback on the progress report and some clarity about the direction in which our project is moving towards.

3. Discussion items

- Discuss indentation and capitalization for the title of the project. Discussion about keeping it shorter in-order to attract more user attention.
- Citation in-text needs to be updated to the correct format
- Grammatical fixes on our progress report, with advice about increasing the scope of our Gantt Chart to incorporate date change in the FYP.
- Indexing of the meeting minutes

4. Goals for the coming sprint (27/2/2020 - 28/2/2020)

- Fix the progress report and submit it before the deadline tomorrow

5. Meeting adjournment and next meeting

- We will keep in touch with Ted through Zoom and emails for further help with our final report

6.10 Minutes of the 10th meeting

Date - 9th March 2020

Time - 3:00 PM HKT

Location - Zoom Call

Members present - Prof Wang, Martin, Tanay

1. Approval of minutes

All members approved the past meeting minutes. Since Prof Wang is an external member, he was not shown or asked to approve previous meeting minutes

2. Report on progress

- Tanay walked Prof Wang through our project and we fielded his questions. We explained to Prof Wang how our project would be a all-in-one platform would serve a bigger purpose for our clients by incorporating ML predictions and an easy to use trading software

3. Discussion items

- Prof Wang and we discussed that our current direction towards a more software technology approach is better compared to a pure machine learning based model approach.
- He asked us to first start working on the strategy page before we go along perfecting the other two pages so that we have something to show for all three pages
- He emphasised the need for us to focus on finishing the parts and to document the challenges that we face and writing them down in the final report

4. Goals for the coming sprint (9/3/2020 - 20/3/2020)

- Tanay would learn more about the React framework and start working on the strategy page interface
- Izen would make changes to the portfolio page and also fix the messages that the user receives
- Martin would add endpoints in the backend to generate predictions, plot the predictions in the frontend chart and add a scheduler.

5. Meeting adjournment and next meeting

- The meeting was adjourned at 3:30 PM.
- The next meeting will be on 20/02/2020 at 3:00 PM HKT

6.11 Minutes of the 11th meeting

Date - 20th March 2020

Time - 3:00 PM HKT

Location - Zoom Call

Members present - Izen, Martin, Tanay

1. Approval of minutes

All members approved the past meeting minutes.

2. Report on progress

- Izen and Tanay finished the first draft of the prediction page
- Martin finalized the pipeline for generating and storing predictions. He further finished revamping the portfolio page.

3. Discussion items

- We discussed the direction of our project at the moment and decided to move forward with the progress that we had made. Based on the current progress, we planned to revamp the direction further.
- We discussed going for the President's cup but decided ultimately to not go for it due to the incremental amount of time that it takes to fill the application and the low return ration

4. Goals for the coming sprint (20/3/2020 - 02/04/2020)

- Tanay would work on creating a cache system on mongodb for storing the ML predictions so that we don't work our servers and we have access to data.
- Izen would debug the issue when prediction is loaded and weird things happen when add/delete stock choices happen in the highstock API
- Martin would add endpoints in the backend to generate predictions, plot the predictions in the frontend chart. He would work on the stock analysis page

5. Meeting adjournment and next meeting

- The meeting was adjourned at 3:30 PM.
- The next meeting will be on 02/04/2020 at 3:00 PM HKT

6.12 Minutes of the 12th meeting

Date - 2nd April 2020

Time - 3:00 PM HKT

Location - Zoom Call

Members present - Izen, Martin, Tanay

1. Approval of minutes

All members approved the past meeting minutes.

2. Report on progress

- Finalized the view of the strategy page and future steps to finish it to completion
- Structure of the report and the upcoming deadlines for the month of April

3. Discussion items

- Decided to split the strategy page into two main parts - laymen view and the strategy list view.
- We discussed splitting the strategy into IF-THEN blocks and limiting user errors by introducing

4. Goals for the coming sprint (02/04/2020 - 20/04/2020)

- Martin - Set up backend for storing the strategy profiles
- Tanay - Create a scheduler task logic for implementing the strategy page
- Izen - Fix add strategy tab, add enabled button, date picker and beautify the page

5. Meeting adjournment and next meeting

- The meeting was adjourned at 3:30 PM.
- The next meeting will be on 20/04/2020 at 3:00 PM HKT

6.13 Minutes of the 13th meeting

Date - 20th April 2020

Time - 3:00 PM HKT

Location - Zoom Call

Members present - Prof Rossiter, Izen, Martin, Tanay

1. Approval of minutes

All members approved the past meeting minutes.

2. Report on progress

- Showcased our final design to Prof Rossiter
- Started working on the final report
- Setup date and time matching for the oral presentation on 16th May

3. Discussion items

- We limited the scope of the project so that there is no project creep
- We discussed about possible directions that the final report could follow

4. Goals for the coming sprint (20/04/2020 - 29/04/2020)

- Finalize and submit the Final Report
- Have a meeting with the communication tutor to finalize the report

5. Meeting adjournment and next meeting

- The meeting was adjourned at 3:30 PM.
- The next meeting will be on 29/04/2020 at 3:00 PM HKT

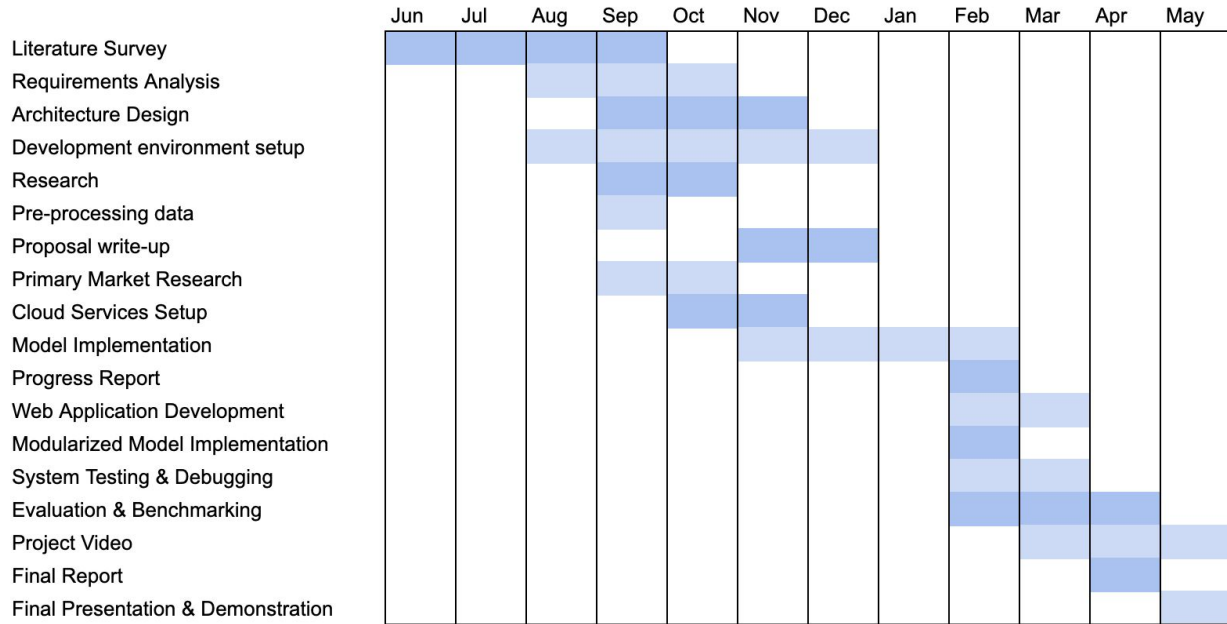
7. Appendix B - Project Planning

7.1 Division of Work²⁰

S No	Task List	Sub Tasks	Individuals Incharge		
			Izen	Martin	Tanay
1	Literature Survey	<i>Financial Markets</i>			x
		<i>News Sources & APIs</i>		x	
		<i>Existing Prediction Models</i>	L		
2	Requirements Analysis		L		
3	Architecture Design			L	
4	Development environment setup			L	
5	Research	<i>Machine Learning Models</i>		x	
		<i>Broker Decision Algorithms</i>			L
		<i>News Sentiment Analysis</i>	x		
6	Pre-processing data	<i>Contiguous Stock Data to Confidence Level</i>			x
		<i>News Headlines for Insights & Trigger Words</i>		L	
7	Proposal write-up		x	x	x
8	Primary Market Research		x		
9	Cloud Services Setup			x	
10	Model Implementation		x	L	x
11	Progress Report		x	x	x
12	System Testing & Debugging	<i>Use-case testing</i>			L
		<i>Unit testing</i>	x		
13	Web Application	Frontend Development	x		x
14		Backend Development		L	
15		API Integration	x		
16	Evaluation & Benchmarking			L	
17	Project Video		x	x	L
18	Final Report		x	x	x
19	Final Presentation & Demonstration		L	x	x

²⁰ Uppercase 'L' denotes leader for the task and lowercase 'x' is for other members who worked on the task

7.2 Gantt Chart



8. Appendix C - Required Software and Hardware

8.1 Hardware

Cloud Server	Digital Ocean (DO)
Computational Specifications	4vCPU and 8GB RAM
Storage Specifications	50GB SSD
Operating System	CoreOS 2303.3.0

We are now using the **DO** cloud server, running our front-end dashboard, backend server and database on **DO** droplets. It runs CoreOS and only is able to ssh in with rsa key authentication.

8.2 Software

Front User-Interface	ReactJS with Typescript
Backend Server	Flask (a micro web framework) and Python scripts
Backend Task Scheduler	APScheduler
Database	MongoDB, InfluxDB

The 3 main applications (frontend dashboard, backend server, database) are now running in containerized environments on the droplet server, bash scripts are used to update the application.

9. Appendix D - Miscellaneous Items

9.1 A list of supported Technical Indicators

- Absolute Price Oscillator
- Acceleration Bands
- A/D (Accumulation/Distribution)
- Aroon
- Aroon Oscillator
- ATR (Average True Range)
- Awesome Oscillator
- Bollinger Bands
- CCI (Commodity Channel Index)
- Chaikin
- CMF (Chaikin Money Flow)
- Detrended price
- Double EMA (Exponential Moving Average)
- EMA (Exponential Moving Average)
- Ichimoku Kinko Hyo
- Keltner Channels
- Linear Regression
- Linear Regression Angle
- Linear Regression Intercept
- Linear Regression Slope
- MACD (Moving Average Convergence Divergence)
- MFI (Money Flow Index)
- Momentum
- NATR (Normalized Average True Range)
- Percentage Price oscillator
- Pivot Points
- Price Channel
- Price Envelopes
- PSAR (Parabolic SAR)
- RoC (Rate of Change)
- RSI (Relative Strength Index)
- SMA (Simple Moving Average)
- Slow Stochastic
- Stochastic
- Super Trend
- Triple EMA (Exponential Moving Average)
- TRIX
- VbP (Volume by Price)
- VWAP (Volume Weighted Average Price)
- Williams %R
- WMA (Weighted Moving Average)
- Zig Zag

10 Appendix E - User Testing

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
TU01	User can access the domain	User opens his preferred choice of browser User types in the URL - https://fyp.martinwongsk.com/	Url - https://fyp.martinwongsk.com/	User lands on the 'Delta Trading Platform' - Portfolio page	As Expected	Pass
TU02A	User can 'Make Buy Order'	User finds the 'Make Order' component User chooses 'BUY' User selects Assets from dropdown of assets User specifies quantity of shares User presses the 'Submit Order' button	side: 'Buy', asset: 'GOOG', quantity: 100	A fill status in the Recent Activity	As Expected	Pass
TU02B	User can 'Make Sell Order'	User finds the 'Make Order' component User chooses 'SELL' User selects Assets from dropdown of assets User specifies quantity of shares User presses the 'Submit Order' button	side: 'SELL', asset: 'FB', quantity: 100	A fill status in the Recent Activity	As Expected	Pass

TU04	User can see predictions for selected stock	User navigates to Stock Analysis Page User clicks on the toggle button labelled 'Show Predictions'	N/A	Dotted line showing predictions for the stocks that are selected by user	As Expected	Pass
TU05	User can see stock tools for the selected assets	User navigates to Stock Analysis Page User clicks on the toggle button labelled 'Stock Tools'	N/A	Dropdown list of items that can be used for stock analysis	As Expected	Pass
TU06	User can add new strategy to the list	User navigates to Strategy Page User fills in the IF and THEN conditions for the 'Add/Edit Strategy' Components	"ticker": "aapl", "prediction_indicator": "ML Closing Price", "relational_operator": ">=", "technical_value": 200.12, "due_date": "2020-04-13", "side": "buy", "amount": 102.34, "enabled": true, "repeat": true	A new generated strategy in the list of strategy	As Expected	Pass