Make Portfolios Smart Again

Final Report

Group name:	PCKK
Project:	Stock Portfolio Management System (SPMS)
Scrum master:	Kenneth Tsang (z5184717@student.unsw.edu.au)
Team members:	Christopher Sheaffe (z5177424@student.unsw.edu.au)
	Zhuowei Gan (z5180689@student.unsw.edu.au)
	Ahmed Hannan (z3445136@student.unsw.edu.au)
Project tutor:	Ali Darejeh

Introduction

As of 2020, there are a variety of stock portfolios management systems (SPMS) on the market, with examples including Yahoo Finance, Sharesight...etc. Back then, sites such as Yahoo Finance (introduced in 1997) were designed to allow users to view and organize their stock investments at one place and acquire relevant information needed to make sound investment decisions. It served as an information aggregation platform to improve the efficiency of investors.

However, the market has changed and technology has rapidly advanced since 1997. With recent development in areas such as deep learning and computational hardware, traditional SPMS can be seen as lacking three key features:

- **Traditional portfolio tools lack integration with recent progress in deep learning.**
    - Deep learning has been adopted by many investment entities to predict and optimize their investments. However, sites like Yahoo Finance have not yet integrated such features.

- **Traditional portfolio tools lack support for advanced portfolio optimization algorithms.**
    - Modern portfolio theory has developed over the years since its debut by Nobel Prize Winner Harry Markowitz and has been adopted by investment banks and hedge funds to optimize their asset distribution and maximize their revenue (Harry, 1990).
    - However, such portfolio optimization tools are not normally accessible to the vast majority of casual investors.
    - The current portfolio management systems such as Yahoo Finance do not provide portfolio optimization features.

- **Traditional portfolio tools lack support for casual investors.**
    - Traditional portfolio management tools are often designed for professional financial investors and the usage of modern stock analysis tools often require a strong mathematical background. Yet the vast majority of stock investors are casual investors who cannot utilise such complex tools. The market cries for a solution that offers a “one-click” optimisation button for portfolio management.

To sum up, traditional SPMS has been an efficient information **aggregation** platform. However, they are not “SMART” enough at the age of AI. Our system, **MPSA (Make Portfolios Smart Again)** is introduced to solve the above-mentioned problem as an information **analysis** platform**.**

MPSA's is a revolutionary system that stands out from its peers for being "smart". To ensure it is smart we adopted the following strategies.

- **S**: **Simple** yet **Sophisticated**. Take the portfolio optimisation function as an example, while giving the One-click optimization option for casual investors, we also display the Efficient Frontier with the help of Monte Carlo Simulation to help the professional users to devise their own portfolio optimization strategy. To keep it simple, the fluent design is adopted to provide an elegant and intuitive interface for customers.
- **M**: **Markowitz** model and von Neumann–Morgenstern utility function are used for portfolio optimization, i.e. to maximizes factors such as expected return and minimizes costs like financial risk for the user (Neumann, 1953).
- **A**: "**Attention** is all you need", the 2017 paper, that introduced the state of the art Transformer architecture, is the quintessential aspect of our prediction model. We have adopted this new technology to massively enhance prediction accuracy (Vaswani, 2017).
- **R**: **Retro** style. We did not forget the key features of classic SPMS systems while innovating, but we also implement them with a **Retrospective** mindset. We strive to provide all features that current SPMS such as Yahoo Finance provides, with a constant retrospection on what can be further improved upon.
- **T**: **Twitter** sentiment analysis was further included in our prediction model to enhance accuracy.

SPMA was made with the following goals:

- Provides a sophisticated portfolio management tools with detailed information with respect to stocks within a user's portfolio
- Provides customizable algorithms and multifaceted parameters for machine learning in the stock recommendations
- Provides portfolio optimization tools for both professional and casual investors
- Provides twitter sentiment analysis to enhance prediction model
- Provides an elegant, intuitive interface that adheres to modern Fluent Design philosophy

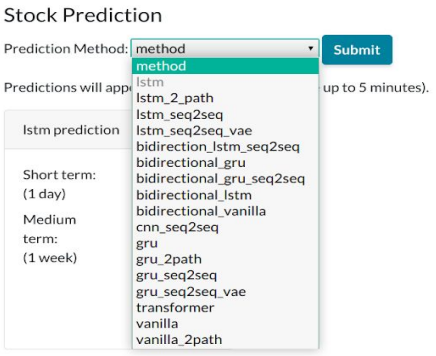
On the next page, we have provided a detailed comparison of our system with respect to traditional stock portfolios.

Comparison to Traditional Stock Portfolios:

Traditional portfolio tools lack integration with recent progress in deep learning.

- Deep learning is rapidly on the rise and permeates every aspect of life.
- As of 2020, Transformers(Attention is all you need) has become state of the art in deep learning (Vaswani, 2017).
- It features high accuracy and low epoch time.
- Albeit transformers are widely adopted in areas such as conversational agents such as Google's BERT, it is still considered novel to many in the financial sector.

SPMA has integrated with 17 latest deep learning algorithms that allow for user customization



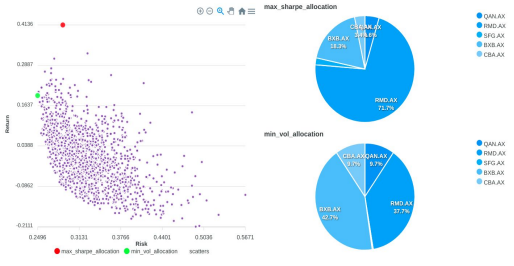
- The state of the art deep learning algorithm, "Transformers" has also been included.

- It is worth noting that we did not use any third-party APIs for any of the above algorithms. Our team is involved in every line of the ML algorithms to make sure its correctness and robustness.

Traditional portfolio tools lack support for advanced portfolio optimization algorithms.

- Modern portfolio theory has developed over the years since its debut by Nobel Prize Winner Harry Markowitz and has been adopted by investment banks and hedge funds to optimize their asset distribution and maximize their revenue (Harry, 1990).
- However, such portfolio optimization tools are not normally accessible to the vast majority of casual investors.

SPMA can give users the optimal budget allocation based on the Markowitz model with only one-click.



- Customizable portfolio optimization strategy based on either maximum return or minimal volatility.
- Detailed graphs of efficient frontier for professional users.

Traditional portfolio tools lack support for casual investors.

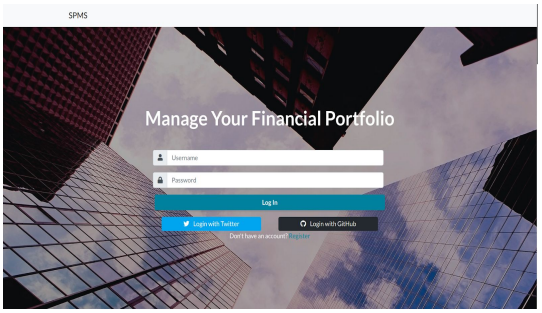
- Traditional portfolio management tools are often designed for professional financial investors and the usage of modern stock analysis tools often require a strong mathematical background. Yet the vast majority of stock investors are casual investors who cannot utilise such complex tools. The market cries for a solution that offers a “one-click” optimisation button for portfolio management.

SPMA has a tailored user interface to allow for the most casual user to navigate through it easily.

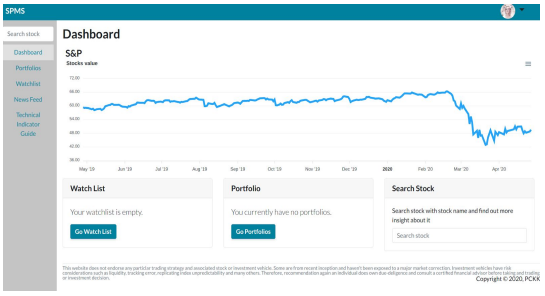
- We strive to make the system feel like an approachable and friendly personal financial assistant for the user, instead of the traditional cold and complex data aggregation notice board.
- We have provided a guide for stock indicators to allow for user’s self-education.
- We have offered the option for one-click optimisation

Functionalities Walkthrough

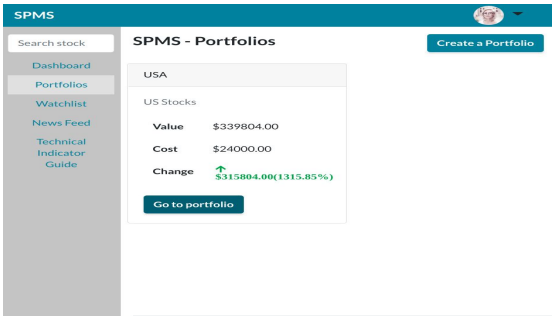
- Upon visiting the homepage, users will be prompted with a login window. They can also choose to register a new account or log in with either Twitter or Github.



- Upon login, users will be greeted by the homepage.
- Users can view an overview of the stock value of the market.
- The homepage has a sidebar that allows users to switch different interfaces.

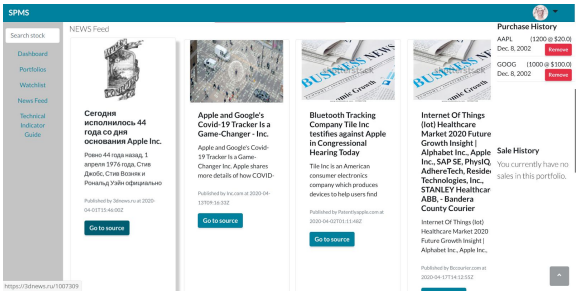


- At the portfolio interface, users can click to create a new portfolio or to go to their existing portfolios.

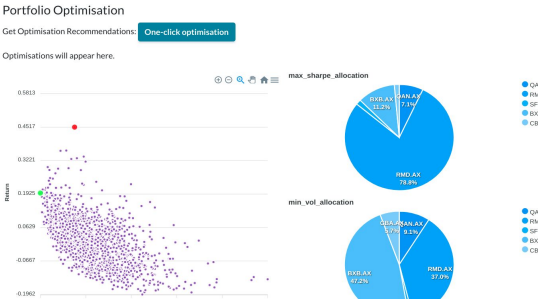


- In the portfolio interface, users can add stocks, or add or remove a purchase history.

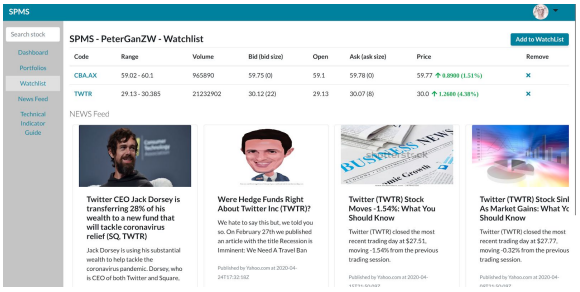
- Scrolling down, users can view relevant news feeds about the portfolio.



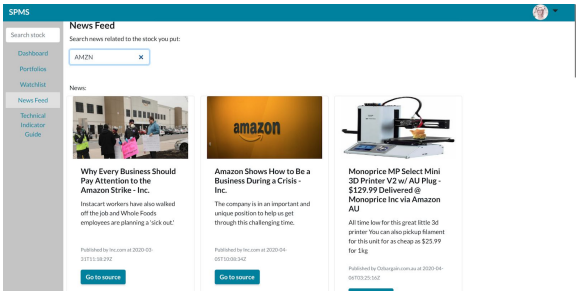
- Scrolling down further, users can click on the optimization button to get a detailed analysis of their portfolio and suggest optimal budget allocation based on the Markowitz model.



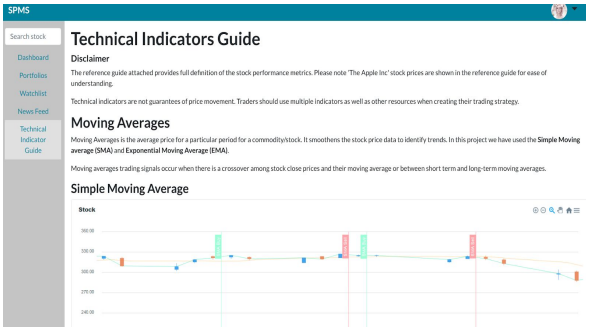
- At the watchlist interface, the user can choose a customized list of stocks to keep track of. They can also view relevant news about the listed stocks.



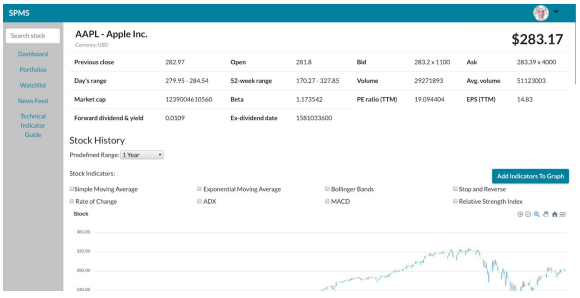
- The news feed section allows users to track relevant news with regard to a particular stock.



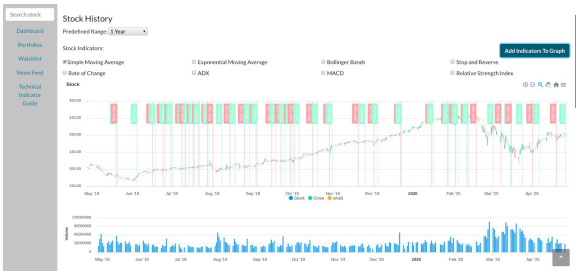
- The stock indicator guide allows users to understand relevant stock indicators that they can use in the stock page.



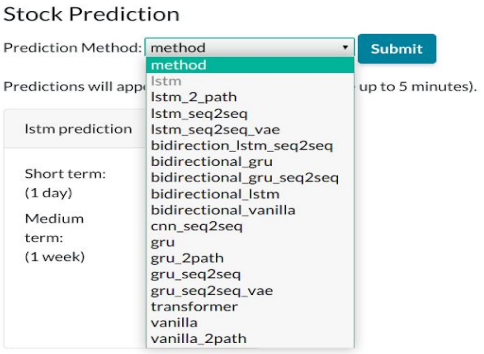
- Users can access a particular stock's page through the search bar, or clicking on the stock in the portfolio interface or watchlist interface



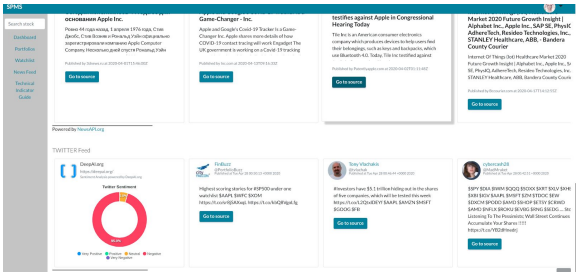
- For a stock's graph, the user can choose from a variety of indicators to be displayed on the chart. They can also customize the range of dates that the chart covers.
- The user can also zoom in/out or save the graph as an offline file through the toolkits offered on the top right corner at each chart.



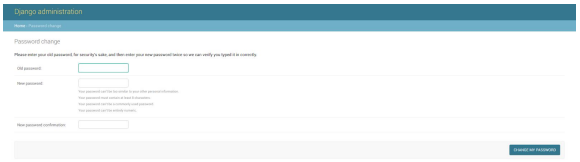
- Users can select from one of the 17 methods for stock prediction.



- Other than viewing news feeds with regard to a particular stock, users can also see the twitter sentiment analysis about a particular stock and relevant tweets.



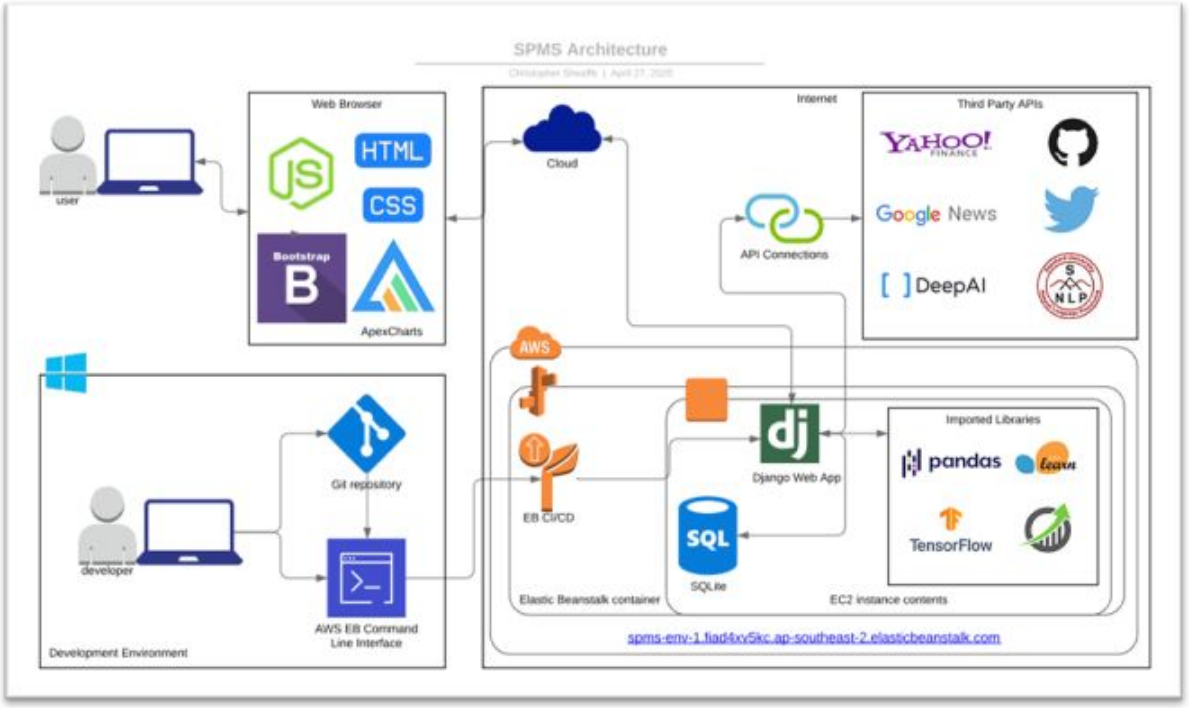
- Users can manage their account password by clicking on their profile icon on the top right-hand corner.



Software Architecture

Our project SPMS is a financial portfolio website. There are many different layers including User Interface Layer, Business Layer, API Layer, Database Layer and Infrastructure layer. In User Interface Layer, html5 and bootstrap are used for styles and ApexChart is used for

making charts. In Business Layer, Django is used for server-side. Pandas, Sklearn and TensorFlow are used for data manipulation. In API Layer, we used yahoo finance API, Google News API, DeepAI API, Stanford NLP API, Twitter API and GitHub API. For Database Layer, we used SQLite. For Infrastructure, we used Beanstalk to run a Linux EC2 to host the server.



Business Layer

Django Web App

The SPMS webapp is fundamentally built on a Django Web framework.
The default Django webapp will consist of the following items:

Template (Custom .html files that Django variables and tags),
App logic (additional .py files for custom libraries and code),
URL decoder (urls.py, that matches a url to its app and view)
View Logic (views.py, this handles requests and connects to backend)
Model (models.py, ORM [Object Relational Mapping], that handles database operations.
Database (where the ORM is finally stored).
More information on Django Architecture can be found on their official website.

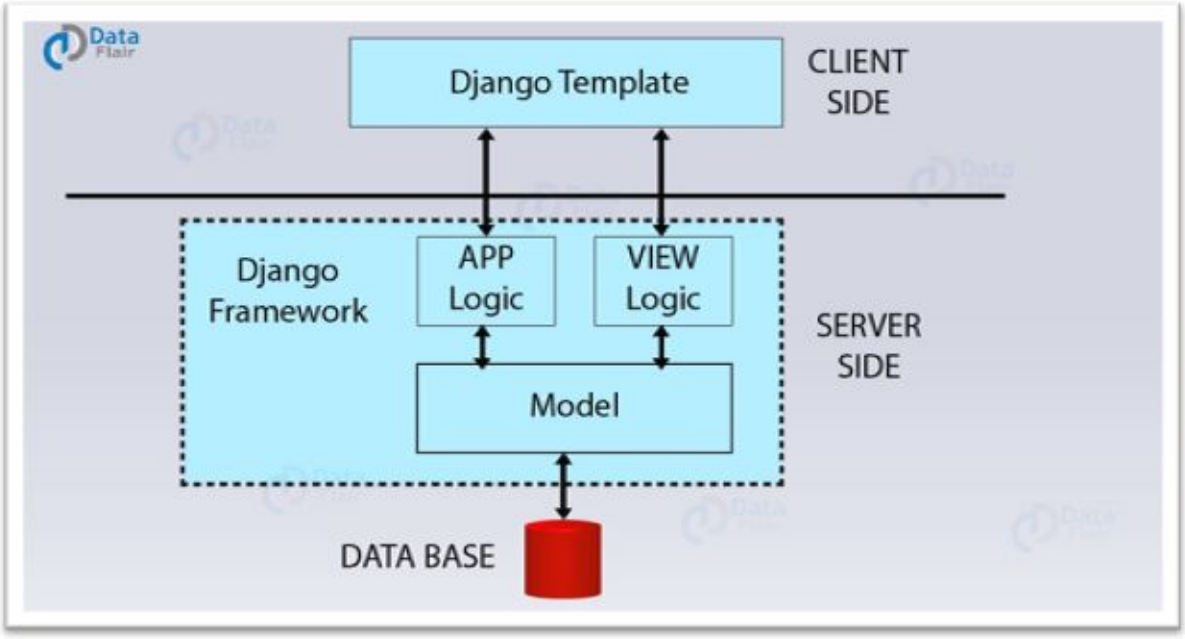


Image Courtesy Towards Data Science

Django Apps

Additional 'Apps' can be created for Django. Apps can be imported from plugins or created from scratch. Each app will have the same structure as seen in the previous Django diagram.

SPMS has created two apps in addition to the default app.

Default App

- SPMS: This app controls the default configuration of the Django web app. It contains the app configuration file (settings.py), as well as the root level URL decoder. The configuration file is where the following is set:
    - Location of static files (non template html, javascript, css, images etc)
    - API keys (in a real production project these keys would be stored in a separate secure file and loaded in via environmental variables. However as this is a short term university project, this was considered to be overkill.
    - Allowed hosts (which IP address and domains are allowed to run the Django app).
    - Additionally installed apps and plugins
    - Middleware used
    - Authentication backends (for user logins)
    - Database configuration
    - Other administrative settings

Custom built Apps

- Portfolio: This app includes all urls view and templates related to stocks, and portfolio management.
- Accounts: This app includes all urls, views and templates related to user logins and registration.

Additional Apps (plugins)

- widget_tweaks (allows extra front end customization of Django forms)
- django_extensions (adds commands to expand Django administration functionality)
- social_django (allows for integration of Twitter and Github authentications)
- bootstrap_modal_forms (allows Django forms to be delivered via a modal instead of a unique page)

The following screenshot is of a typical url pattern in urls.py

```python
app_name = 'portfolio'
urlpatterns = [

    # ************ CORE PAGES **************
    path('portfolios/', views.portfolios, name='portfolios'),
    path('watchlist/', views.watchlist, name='watchlist'),
    path('dashboard/', views.dashboard, name='dashboard'),
    path('news_feed/', views.news_feed, name='news_feed'),
    path('stock/<str:stock_id>', views.stock, name='stock'),

    # ************ Watchlist Functions ************
    path('remove_stock_watchlist/<str:stock_id>/', views.remove_stock_watchlist, name='remove_stock_watchlist'),
    path('add_stock_watchlist/<str:stock_code>/', views.add_stock_watchlist, name='add_stock_watchlist'),

    # ************ Portfolio Functions ************
    # view single portfolio
    path('<int:portfolio_id>/', views.portfolio, name='portfolio_detail'),
    # add and remove stock purchases to portfolio
    path('add_stock_purchase/<int:portfolio_id>', views.add_stock_purchase, name='add_stock_purchase'),
    path('remove_stock_purchase/<int:portfolio_id>/<int:stock_purchase_id>/', views.remove_stock_purchase, name='remove_stock_purchase'),
    # add and remove stock sales to portfolio
    path('add_stock_sale/<int:portfolio_id>', views.add_stock_sale, name='add_stock_sale'),
    path('remove_stock_sale/<int:portfolio_id>/<int:stock_sale_id>/', views.remove_stock_sale, name='remove_stock_sale'),
    # create and delete a portfolio
    path('create_portfolio/', login_required(views.create_portfolio.as_view()), name='create_portfolio'),
    path('delete_portfolio/<int:pk>/delete/', login_required(views.delete_portfolio.as_view()), name='delete_portfolio'),

    # ********** API FUNCTIONS ***************
    url(r'^api/yahoo/search/', views.yahoo_stock_search, name='yahoo_stock_search'),
    url(r'^api/google/news/business/', views.google_news_top, name='google_news_top'),
    url(r'^api/google/news/stock_list/', views.google_news_stocks, name='google_news_stocks'),
    url(r'^api/twitter/topics/', views.twitter_topic_search, name='twitter_topic_search'),

    url(r'^api/historical_data/', views.historical_data, name='historical_data'),
    url(r'^api/ml_prediction/', views.ml_prediction, name='ml_prediction'),
    url(r'^api/stock_indicators/', views.stock_indicators, name='stock_indicators'),
    url(r'^api/portfolio_optimisation/', views.portfolio_optimisation, name='portfolio_optimisation'),

    # ********** ULRs That are functional but are not linked to on front end ************
    path('stock/yahoo/search/', TemplateView.as_view(template_name='portfolio/stock_search.html'), name='stock_search'),
    path('search_stock_watchlist/', login_required(TemplateView.as_view(template_name='portfolio/search_stock_watchlist.html')), name='search_stock_watchlist'),
]
```

The following screenshot is a typical view function within views.py

```python
@login_required
def portfolio(request, portfolio_id):
    """

    Parameters
    ----------
    request : Django Request
        The Django Request Object
    portfolio_id: int
        ID of a portfolio

    Returns
        portfolio/portfolio.html template with details of the given portfolio.
        If portfolio does not belong to logged in user then 403 error is raised.
    -------

    """
    tempUser = request.user
    portfolios = Portfolio.objects.filter(user=tempUser)
    portfolio = get_object_or_404(Portfolio, pk=portfolio_id)
    holdings = get_portfolio_holdings(portfolio)

    purchases = portfolio.stock_purchases.all()
    heldStocks = Stock.objects.distinct().filter(stockpurchase__in=purchases)
    stock_list = [str(stock.code) for stock in heldStocks]
    holdings["stock_name_list"] = [h["name"] for h in holdings["stock_holdings"]]

    if portfolio.user != tempUser:
        raise PermissionDenied
    return render(request, 'portfolio/portfolio.html', {'portfolios':portfolios, 'portfolio':portfolio, 'holdings':holdings, 'stock_list':json.dumps(stock_list)})
```

API Layer

Finance APIs

Our financial data is provided by Yahoo Finance. We chose Yahoo Finance as it is considered the gold standard of free financial data. However, Yahoo ended their official finance API support a few years ago. As such we had to choose whether to look for another data source or for work arounds. We chose to look for existing libraries that made use of Yahoo Finance. We found Pandas-Datareader and yFinance.

We encountered a severe bug in yFinance that would crash the server when looking for detail for stocks that did not have institutional investors associated with it. As there was no fix available, we decided to download the library, bug fix and patch it then store it locally within the Portfolio app.

Screenshot of yFinance data:

Pandas-Datareader was able to quickly provide historical data for stock prices, while yFinance was able to provide current financial information related to a stock.

Screenshot of Pandas Datareader data:

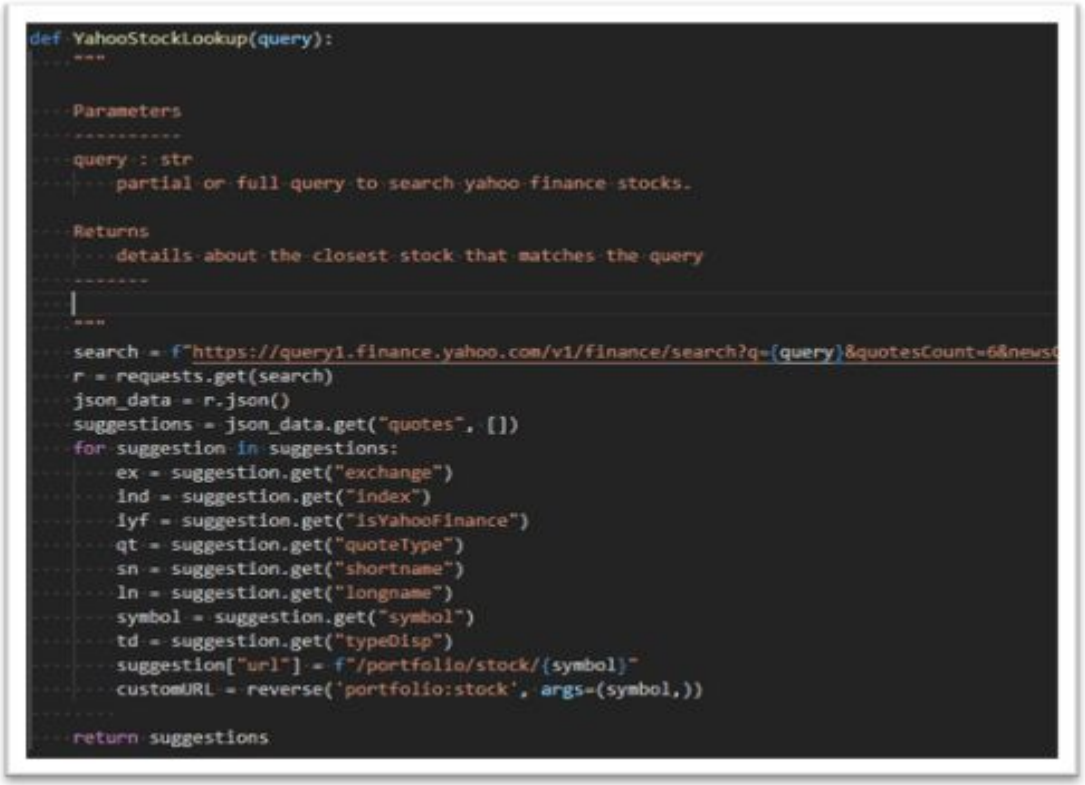
We also reverse engineered the Yahoo Finance stock search and integrated it into our stock search functionality.

Screenshot of search functionality:

The following screenshot demonstrates the use of Pandas Datareader(pdr). The input parameters are a stock code (eg “cba.ax”) and a historical range key (this is passed to a helper function to calculate appropriate datetimes).

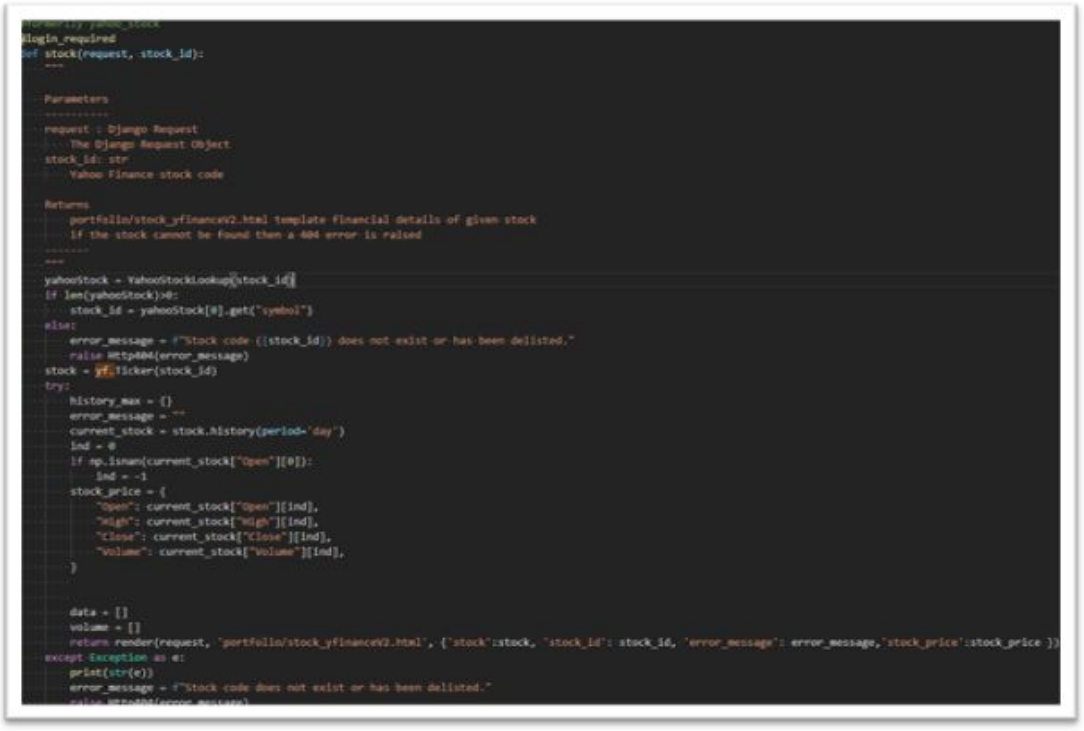
```
def GetHistoricalData(stock_code, historical_range):
    now = datetime.datetime.now()
    start = CalculateStartTime(now, historical_range)
    end = now.strftime("%Y-%m-%d")
    hist_data = pdr.get_data_yahoo(stock_code,start, end)
    return hist_data
```

The following screenshot demonstrates how the Yahoo Finance stock search is used.



```
def YahooStockLookup(query):
    """

    Parameters
    ----------
    query : str
        partial or full query to search yahoo finance stocks.

    Returns
        details about the closest stock that matches the query
    -------

    """
    search = f"https://query1.finance.yahoo.com/v1/finance/search?q={query}&quotesCount=6&news
    r = requests.get(search)
    json_data = r.json()
    suggestions = json_data.get("quotes", [])
    for suggestion in suggestions:
        ex = suggestion.get("exchange")
        ind = suggestion.get("index")
        iyf = suggestion.get("isYahooFinance")
        qt = suggestion.get("quoteType")
        sn = suggestion.get("shortname")
        ln = suggestion.get("longname")
        symbol = suggestion.get("symbol")
        td = suggestion.get("typeDisp")
        suggestion["url"] = f"/portfolio/stock/{symbol}"
        customURL = reverse('portfolio:stock', args=(symbol,))

    return suggestions
```

The following screenshot demonstrates how the yFinance library is used. First a stock is checked against the Yahoo Finance stock search to ensure it exists, then a call to yFinance is made.



```
@login_required
def stock(request, stock_id):
    """

    Parameters
    ----------
    request : Django Request
        The Django Request Object
    stock_id: str
        Yahoo Finance stock code

    Returns
        portfolio/stock_yfinanceV2.html template Financial details of given stock
        If the stock cannot be found then a 404 error is raised
    -------
    """
    yahooStock = YahooStockLookup(stock_id)
    if len(yahooStock)>0:
        stock_id = yahooStock[0].get("symbol")
    else:
        error_message = f"Stock code ({stock_id}) does not exist or has been delisted."
        raise Http404(error_message)
    stock = yf.Ticker(stock_id)
    try:
        history_max = {}
        error_message = ""
        current_stock = stock.history(period='day')
        ind = 0
        if np.isnan(current_stock["Open"][0]):
            ind = -1
        stock_price = {
            "Open": current_stock["Open"][ind],
            "High": current_stock["High"][ind],
            "Close": current_stock["Close"][ind],
            "Volume": current_stock["Volume"][ind],
        }

        data = []
        volume = []
        return render(request, 'portfolio/stock_yfinanceV2.html', {'stock':stock, 'stock_id': stock_id, 'error_message': error_message,'stock_price':stock_price })
    except Exception as e:
        print(str(e))
        error_message = f"Stock code does not exist or has been delisted."
        raise Http404(error_message)
```

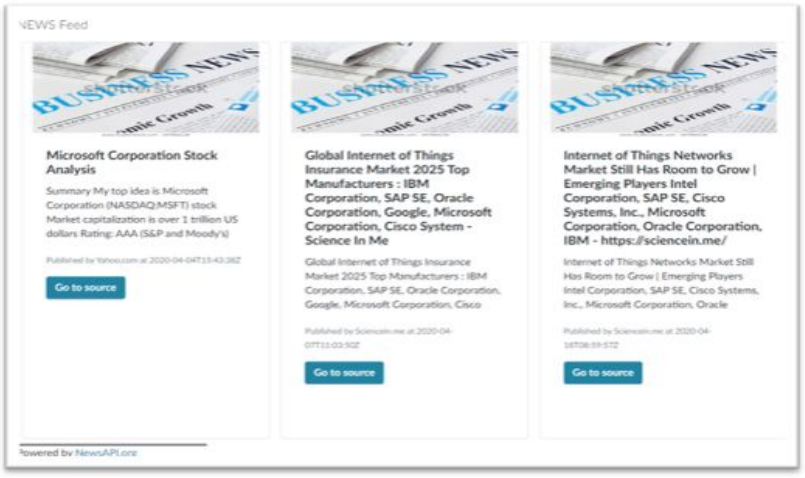
News Feed

News feeds are provided by Google News API.
An api key was required before calls were authorised. The project makes use of two API calls.

1. Top Business headlines
2. Articles filtered by subject and publish period

Code can be found within the GoogleNewsAPI.py file.

Screenshot of news feed:



The primary challenge of the news feed was that we needed to get all news related to multiple companies to show in the news feed of a portfolio. The issue was that the API had no functionality to call for a partial match of this company OR a partial match of this company etc. It could only search for an EXACT match of this company OR an EXACT match of that company. Our solution was to strip as much of the company's name so that it would increase the chances that it would appear exactly in a news article. i.e. We would remove punctuation, remove '.com' or Ltd. Or Limited so that the more casual reference of the company remained. This appears to have worked well.

The following screenshot demonstrated a news API call. It takes a list of company names, cleans the text, joins all companies into a single query, sets the date period to the last month, constructs the query URL then calls the API.

Twitter Feed and Sentiment Analysis

Twitter feeds are provided from Twitter through a library called Tweepy.

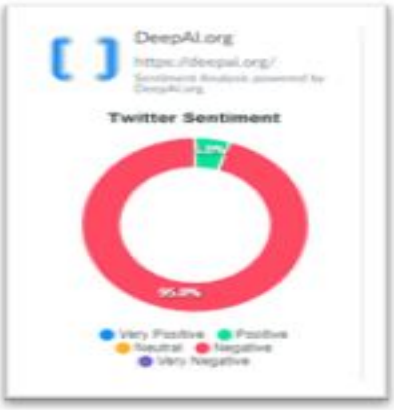
Sentiment analysis was provided via a DeepAI API which was developed by Stanford Natural Language Processing Group.

In order to use the API we required an access key. Before we could get a key we first had to register the application with Twitter. We were required to provide our url that hosted the application as well as a description of when we were going to use Twitter's data. This meant that we first had to host and deploy the site on Amazon before we could start developing anything to do with Twitter.

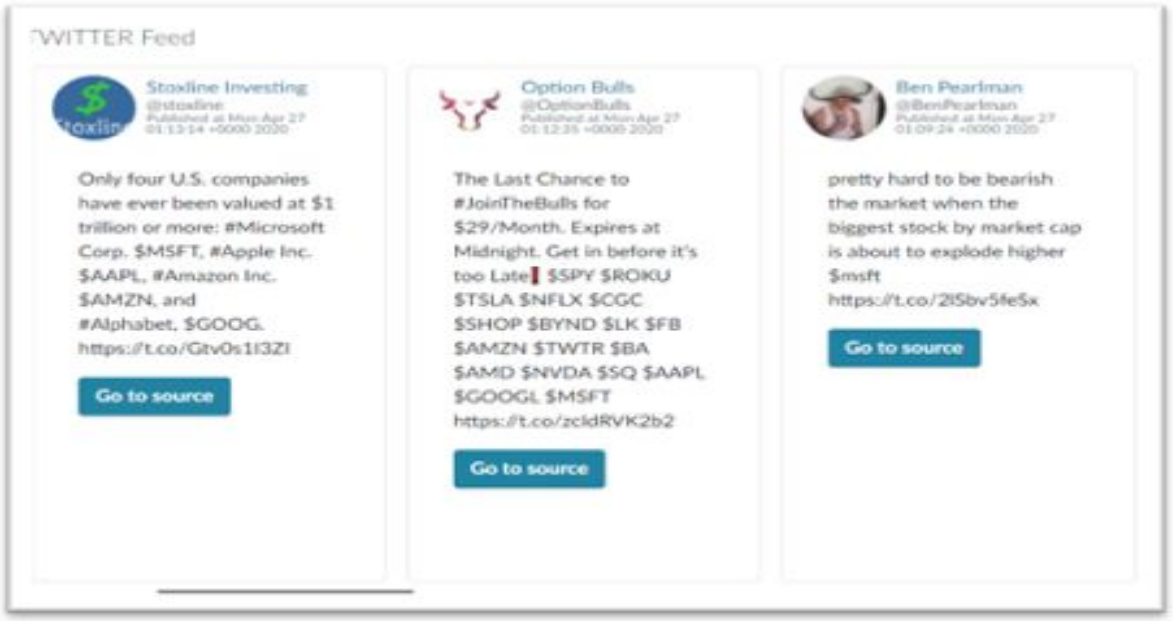
In order to use the sentiment analysis api, tweets had to first be collected then cleaned. All tweets were stripped of punctuation, and converted to lowercase.

We also found that most financial tweets skewed negative. This may have been heavily impacted due to the recent oil crisis and Covid-19 pandemic.

Screenshot of Sentiment Analysis:



Screenshot of Twitter Feed:



Code can be found in the TwitterFeedAPI.py file.

The following screenshot shows a Twitter feed api call that gets all tweets (and sentiment) related to stocks in the provided stocklist. As Twitter is more informal than news outlets, stock codes are used in addition to the stock business names when creating a query.

```
def GetFinancialTweets(stock_list, stock_name_list, no_retweets=True):
    if stock_list and stock_name_list:
        queryList = QuerySplit(stock_list)
        queryList = AppendHashAndDollar(queryList)
        cleaned_names = ["("+s.replace(",", "").replace(".","").lower()+")" for s in stock_name_list]
        combined_queryList = queryList + cleaned_names
        query      = QueryListToQuery(combined_queryList)
        if no_retweets:
            query += twitter_filters["no_retweets"]
        tweets     = list(SearchTwitter(query))
        tweets = [tweet._json for tweet in tweets]
        cleaned_tweets = [re.sub(r'[^A-Za-z0-9 ]+', '', tweet.get("full_text","")).strip() for tweet in tweets]
        tweetText = ".\n".join(cleaned_tweets)
        if len(tweetText) > 0:
            sentiments = GetSentimentOfText(tweetText)
            return tweets, sentiments
    return [],[]
```

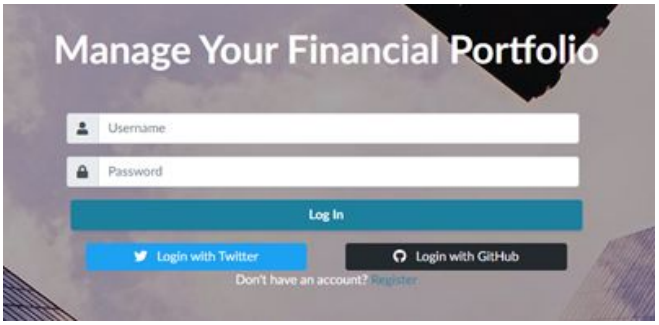
Social Media Authentication Integration

Twitter and GitHub APIs have been integrated to allow users to log into SPMS via using their social media accounts. Please note that this integration will only work on the LIVE server hosted on AWS here
http://spms-env-1.fiad4xv5kc.ap-southeast-2.elasticbeanstalk.com/portfolio/dashboard/

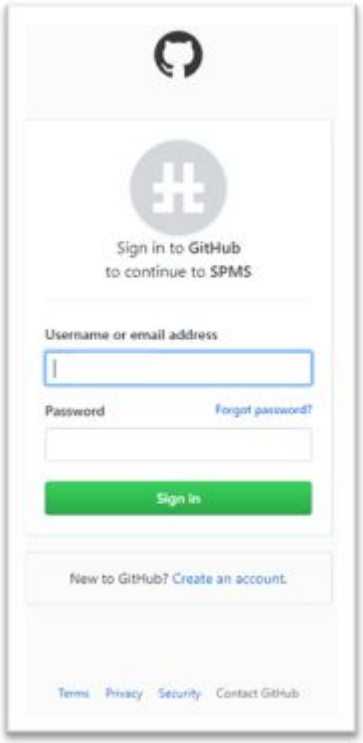
This is because Twitter and GitHub require developers to provide an IP address and/or domain that their app will run one. This means that the APIs will BLOCK attempts to login when running SPMS in a local development server (localhost).

GitHub API keys can be generated from within the Developer's page of *your GitHub account.*
Twitter API keys can be generated from within the Developer's page of your *Twitter account.*
Django Authentication Integration is provided by the Django Social Auth plugin.

Social Media Login Integration Screenshot:



GitHub and Twitter authentication screens:

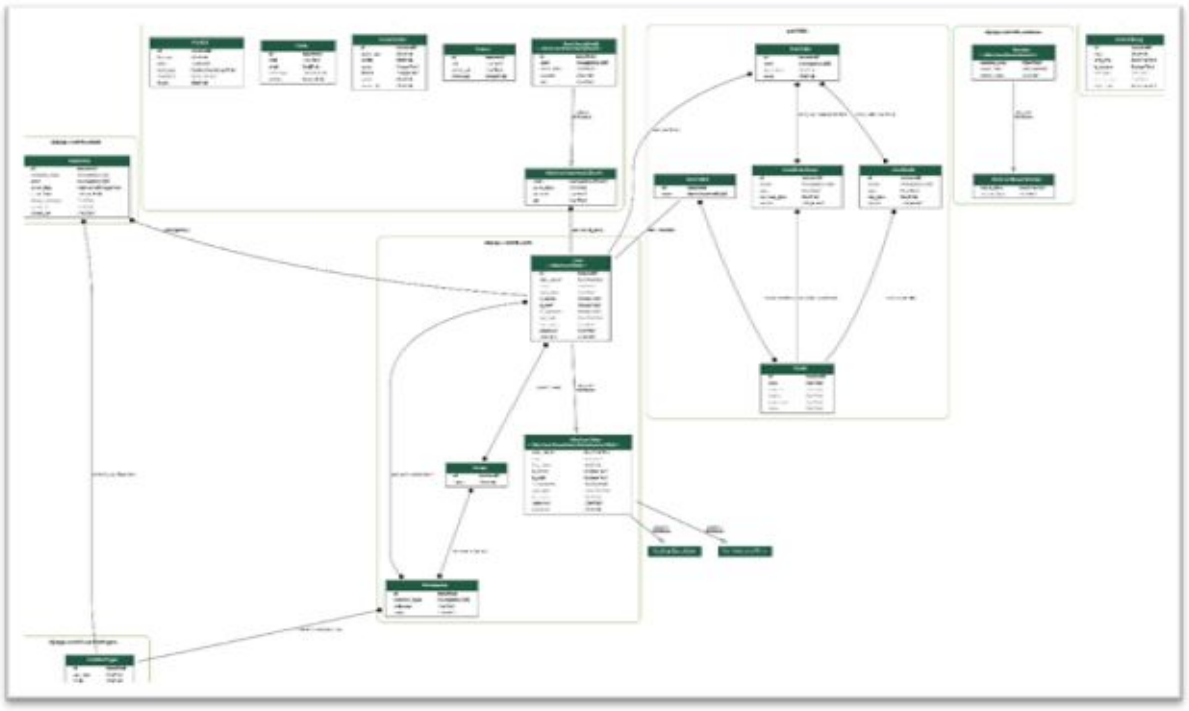


Once API keys have been registered, installing into Django is relatively simple. Just add the necessary lines to the configuration file (settings.py) as per the plugin's instructions.

Database Layer

Django uses an Object-relational mapping and allows developed to use the same code regardless of the database backend they use.

We have used the SQLite as the database for this project as it was lightweight and provided a fast development path to a working live prototype.



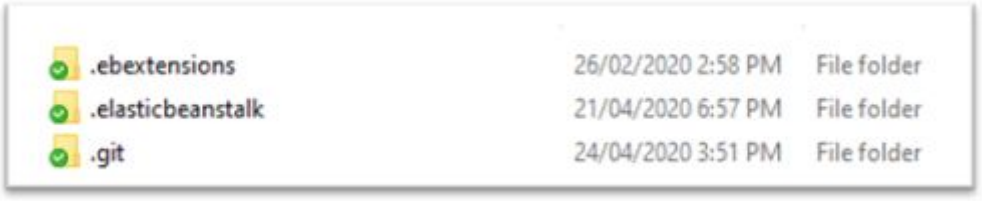
Infrastructure and Framework

The production server is available at this link:
http://spms-env-1.fiad4xv5kc.ap-southeast-2.elasticbeanstalk.com/

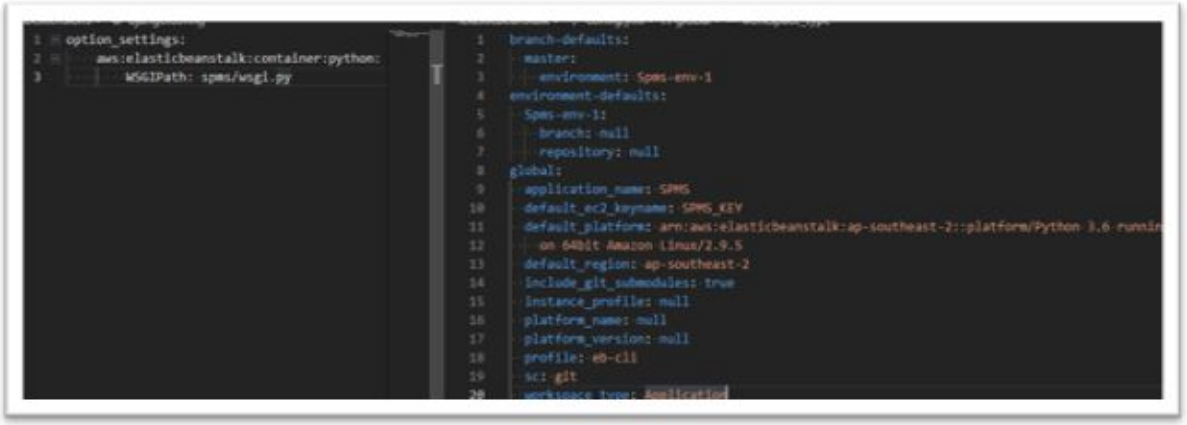
The production server is hosted on Amazon Web Services (AWS) using Elastic Beanstalk (EB).

This application was primarily developed on Windows 10.
The code is deployed to the server using the EB Command Line Interface (EB CLI).

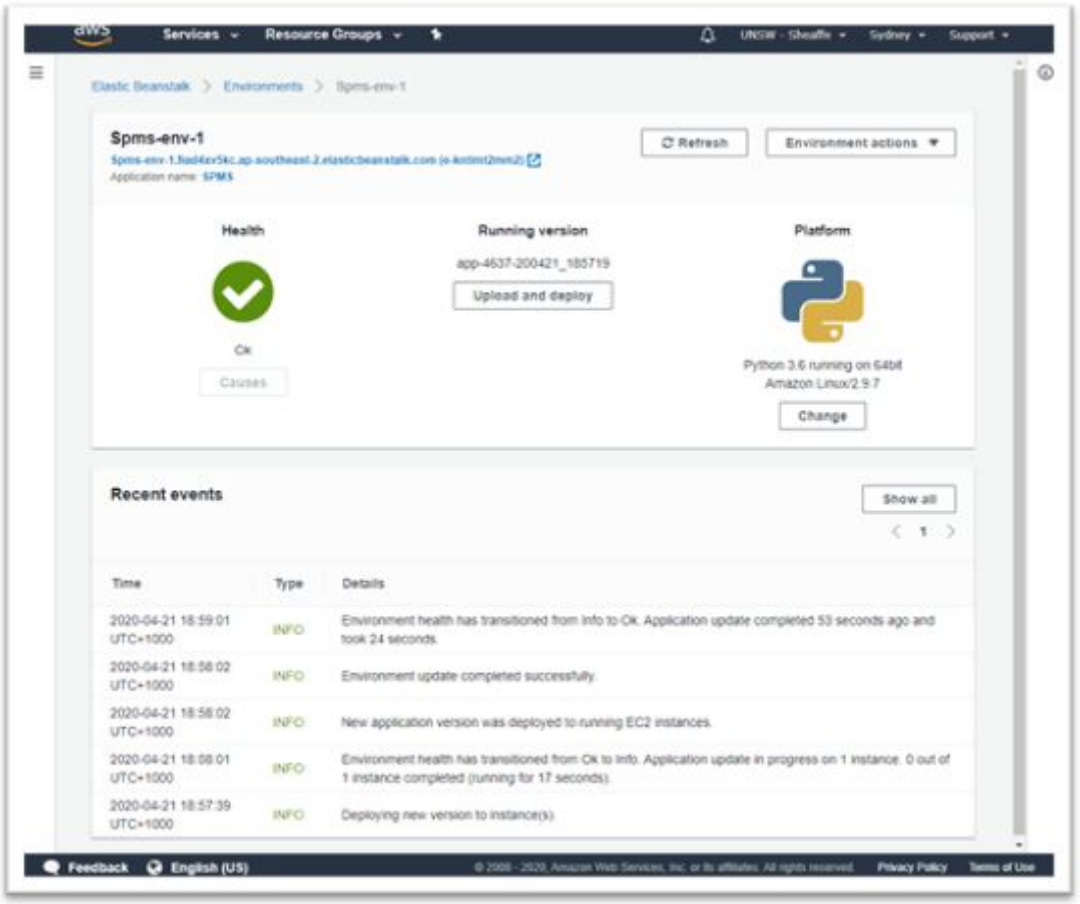
EB CLI collects the most recent Git repo commit, packages it and deploys it to the EB server.
Screenshot of Git and Elastic Beanstalk configuration folders:



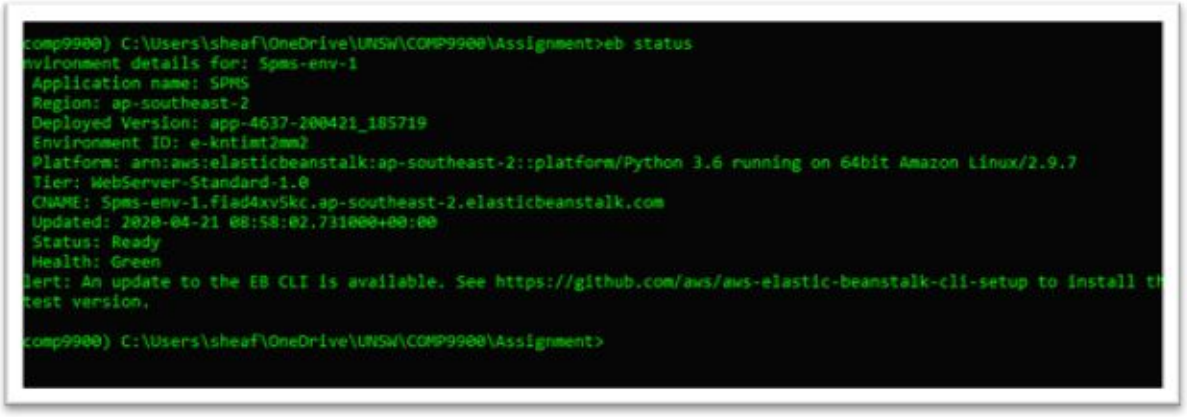
Screenshot of EB configuration files:



Screenshot of EB management console with SPMS app running:



Screenshot of the EB CLI:



Functionalities and Implementation Challenges

Login page:
Functionality:

- A summary of what MPSA is and what we can provide for the user.
- Help user to register and login

Challenges:

- The main challenge for the login page is on how we should design a clean and attractive interface. To meet this challenge, we consulted multiple personnel on their opinion on the interface, including our project tutor Ali, a PhD in Human-Computer Interaction. With their help, we have greatly improved our homepage in terms of cleanliness and user-friendliness.

Dashboard page:
Functionality:

- After logging in, users will reach the dashboard page. Here, users can see the cards in the centre and the sidebar. The cards show a fraction of features including WatchList, Portfolio and Search Stock. In the sidebar, users can navigate to different pages including Portfolio page, Watchlist page, News Feed page and Technical Indicator Guide.

Challenges:

- The main challenge here is the implementation of a search bar. A user might search for a stock using its stock code, such as ‘AAPL’ for Apple.Inc, but another user might search for “APPLE” instead. We designed our search bar such that both queries can be addressed. We also cater for shorthands such as “AAP” and users would be directed to the Apple page since it is the first result among the suggested stocks.

WatchList page:
Functionalities:

- Users can add stocks of interest into the watchList. After that, users can see the information of the stock in the watchList including range, volume, bid price, bid size, open, ask price, ask size, price, price change and price change percentage. Below the information, users can see the news feed related to the chosen stocks. If a stock is not needed any more, users can delete it by pressing the cross button.

Challenges:

- The challenge here is how we should differentiate a watchlist with respect to a portfolio, and understand what the user wants when he/she adds a stock into a watchlist instead of a portfolio. After a careful study on the current watchlist implementation analysis on traditional platforms such as Yahoo Finance, we decided to incorporate news feeds and stock stats onto the watchlist, as the user would be most likely interested in the news that could affect a stock of interest.

News Feed page:
Functionality:

- Users can search a stock here. Then, they can see the news related to the stock. The title, a paragraph and the publication date will be shown. By clicking the news, users can navigate to the source website. This feature is powered by NewsAPI.org.

Challenges:

- The main challenge here is the utilization of news API. We have compared and benchmarked a few APIs on the market and we picked Google News for the final product.

Technical Indicator Guide page:
Functionality:

- Users can view the implications and concepts of different indicators here. The signals given by the indicators are explained here so that users can understand the indicators more.

Challenges:

- All of our team members do not have financial background, hence it was challenging to study all the indicators at once.

Stock page:
Functionalities:

- After searching a stock, users can go to the Stock page. Besides basic information, users can see Stock History, Stock Prediction, News Feed and Twitter Feed for the stock.
- The fundamental function of Stock History is to show the candlestick chart and volume over a defined period of time. On top of that, users can choose different indicators and follow the buy/sell signals given by the indicators. For an in-depth explanation of the indicators, users can go to the Technical Indicator Guide page.
- Users can choose different machine learning prediction models in Stock Prediction. After a brief moment, the short-term behavior and the medium-term behavior of the stock will be predicted with the assigned prediction model.
- The related news will be shown in the News Feed section and the related twitters will be shown in the Twitter Feed section. In addition, the twitter sentiment result given by DeepAi.org is shown. Users can see how positive/negative people's twitters are.

Challenges:

- The implementation of the ML algorithms requires careful tweaking of parameters. However, due to the recent outbreak, the stock trends are much less predictable and less learnable. To address this issue, we decided to predict a trend upwards/downwards in either short-term or medium-term, instead of the exact values. This is because the predicted values sometimes fall into the negatives due to the extremely negative gradients for the majority of stocks.
- The implementation of stock indicator requires study of ApexChart API which took a considerable amount of time.

- Another challenge is that data for twitter feeds and analysis are dirty and unorganized. To utilize them, we needed to clean the data to ensure all the necessary data are present and the format of the data is correct.
- For some backend operations like model training and data fetching, the processing time can be as long as 3 minutes. Without special care, users may face non-reactive user interfaces. To tackle this problem, for some components, loading state is implemented. That way, we can provide our service without impacting user experience too much.
- There are various bugs in Yahoo finance API. We had to fix ourselves.

Portfolio page:
Functionalities:

- Users can create portfolios on this page. After that, users can add/delete stock purchase history and sale history into the portfolio. The portfolio value, portfolio cost and portfolio profit are calculated and shown. For each involved stock, value, cost, price, the number of shares owned, profit and daily change will be displayed.
- Moreover, there is a portfolio optimization feature powered by Markowitz Model. Users can see the return over risk graph with point of max_sharpe_allocation and point of min_vol_allocation indicated. At the same time, the explicit allocations for both types are shown in the pie chart.

Challenges:

- It took a considerable amount of effort to understand the Markowitz model and to implement it. Furthermore, as Apexchart lacks documentation for scatter graphs, it took much trial and error to present the Efficient Frontier graph in the final product.

User Manual:

To use the site please visit the following URL:
http://spms-env-1.fiad4xv5kc.ap-southeast-2.elasticbeanstalk.com/
If you wish to run a local server, please follow the instructions found within the readme.md of the GitHub page found here:
https://github.com/unsw-cse-comp3900-9900/capstone-project-pckk
Instruction on setting up local server:

1. Set up python libraries.

```
pip install -r requirements.txt
```

2 Run the dev server:

```
cd spms
```

```
python manage.py runserver
```

User manual for Stock Technical Indicators in MPSA

Stock Technical Indicators are used to alert on the need to study price action with greater detail, confirm other technical indicators or predict future stock prices direction. There are mainly two types of technical stock indicators:

1) Lagging Stock Technical Indicator
2) Leading Stock Technical Indicator

Lagging Stock Technical Indicator

They are used to analyse the price movements by looking at the uptrends and downtrends. In our website we used below three mostly:

1) Moving averages- comprising of simple and exponential
2) Bollinger Bands (BB)
3) Parabolic Stop and Reverse (SAR)

Moving Averages

Moving Averages is the average price for a particular period for a commodity/stock. It smoothens the stock price data to identify trends. In this project we have used the **Simple Moving average (SMA)** and **Exponential Moving Average (EMA)**. The below formula is used to calculate the moving averages:

1) $Simple\ Moving\ Average_n = \frac{\sum_{i=0}^{n} Close\ Prices}{n}$
2) $Exponential\ Moving\ average_n = \frac{2}{n+1} \times (current\ close\ price - previous\ EMA_n) + previous\ EMA_n$

Where generally $Exponential\ Moving\ average_n = SMA_n\ (n\ periods\ close\ price)$ and $n = no.\ of\ days$

Bollinger Bands

Bollinger Bands (BB) are overlays that identify statistically normal stock price movements. In general, twenty days and two standard deviations are commonly used for its calculation. It's a good indicator to measure whether stocks have been overbought or oversold as their prices are volatile.

- $Middle\ Band = Simple\ Moving\ Average_{20}(Close)$
- $Upper\ Band = Simple\ Moving\ Average_{20}(Close) + 2 \times STD_{20}(Close)$
- $Lower\ Band = Simple\ Moving\ Average_{20}(Close) - 2 \times STD_{20}(Close)$

Parabolic Stop and Reverse Indicator

Parabolic stop and reverse SAR is overall that identifies reversal points between stock prices uptrends and downtrends.

The algorithm for this is as following:

a) At first, it calculates the **Extreme Point (EP)** calculation:

$$EP\ (Uptrend) = Highest\ High\ Current\ Uptrend$$

$$EP\ (Downtrend) = Lowest\ Low\ Current\ Uptrend$$

b) Next, we calculate the **Acceleration Factor (AF)** via the following:

$$AF\ (Uptrend) = if\ (New\ High\ EP\ (Uptrend))\ then$$

$$Current\ AF\ (Uptrend) = Previous\ AF\ (Uptrend) + 0.02$$

$$Else\ Current\ AF\ (Uptrend) = Previous\ AF\ (Uptrend)$$

$$AF\ (Downtrend) = if\ (New\ Low\ in\ EP\ (Downtrend))\ then$$

$$Current\ AF\ (Downtrend) = Previous\ AF\ (Downtrend) + 0.02$$

c) At last we calculate the **Parabolic Stop and Reverse (SAR)** indicator:

$$Current\ SAR\ (Up) = Prev.\ SAR\ (Up) + Prev.\ AF \times (Prev.EP\ (UP) - Prev.SAR(Up))$$

$$Current\ SAR\ (Dn) = Prev.\ SAR\ (Dn) - Prev.\ AF \times (Prev.EP\ (Dn) - Prev.SAR(Dn))$$

Leading Stock Technical Indicator

Leading stock technical indicators are used to lead price movements by identifying their momentum as either the rising prices to rise further or falling prices to fall further.

Most of these are either centred or bounded oscillators and consists of the following which is developed on our website:

1) Average Directional Movement index (ADX)
2) Commodity Channel index (CCI)
3) Moving Averages Convergence/Divergence (MACD)
4) Rate of Change (ROC)
5) Relative Strength Index (RSI)
6) Stochastic Oscillator Full (STO)
7) Williams (%R)

Average Directional Movement index (ADX)

ADX is a bounded oscillator which measures a stock price trend's strength and momentum. Fourteen days are generally used in the calculation.

The algorithm for this is as following:

a) Firstly, we calculate the **true range, positive/negative directional movement** calculation.

$$TR\,(1) = Max\ Between\ (High - Low,\ |High - previous\ Close|,\ Low - Previous\ Close|)$$

$$DM\,(1) = If\ \ (Current\ High - previous\ High > previous\ Low - current\ Low)\ :$$

$$DM\,(1)p = Max\ Between\ (Current\ High - Previous\ High, 0)$$

$$Else\ DM\,(1)p = 0$$

$$DM\,(1)n = If\ \ (Previous\ Low - Current\ Low > Current\ High - previous\ High)\ :$$

$$DM\,(1)n = Max\ Between\ (previous\ Low - current\ Low, 0)$$

$$Else\ DM\,(1)n = 0$$

b) Fourteen days **smoothing with Wilder's techniques**

$$TR\,(14) = Prev\ TR\,(14)\ \times \left(\frac{Prev\ TR(14)}{14}\right) + TR(1)$$

$$initial\ TR\,(14) = SMA_{14}(TR(1))$$

$$DM\,(14)p = Prev\ DM\,(14)p \times \left(\frac{Prev\ DM(14)p}{14}\right) + DM(1)p$$

$$initial\ DM\,(14)p = SMA_{14}(DM(1)p)$$

$$DM\,(14)n = Prev\ DM\,(14)n \times \left(\frac{Prev\ DM(14)n}{14}\right) + DM(1)n$$

$$initial\ DM\,(14)n = SMA_{14}(DM(1)n)$$

c) Fourteen days **positive/negative directional index** calculation:

$$DI\,(14)p = |100 \times \left(\frac{DM(14)p}{TR(14)}\right)|$$

$$DI\,(14)\,n = |100 \times \left(\frac{DM(14)n}{TR(14)}\right)|$$

d) Fourteen days **directional movement index** calculation:

$$DX\,(14) = \left[100 \times \left(\frac{Difference\ DI(14)}{Sum\ DI(14)}\right)\right]$$

$$Difference\ DI\,(14) = |DI\,(14)\,p - DI\,(14)\,n|$$

$$Sum\ DI\,(14) = DI\,(14)\,p + DI\,(14)\,n$$

e) Fourteen days **average directional movement index** calculation:

$$current\ ADX\,(14) = \left[\frac{(previous\ ADX(14)\times 13)+current\ DX(14)}{14}\right]$$

$$initial\ ADX\,(14) = SMA_{14}(DX(14)$$

Commodity Channel index (CCI)

Commodity channel index is a bounded oscillator that measures a stock's price variation from it's statistical mean. Twenty days and constant factors are commonly used to make sure most values fall within bands.

The algorithm for this is as following:

a) **Typical Price** calculation

$$TP = \frac{High+Low+Close}{3}$$

b) **Typical Price Smoothing** calculation

$$SMA_{20}(TP) = \frac{SMA_{20}(TP)}{20}$$

c) **Mean absolute deviation** calculation

$$MAD = \frac{SMA_{20}|SMA_{20}(TP)-TP|}{20}$$

d) **Commodity Channel Index** calculation

$$CCI(20, 0.015) = \frac{Typical\ Price - SMA_{20}\ of\ Typical\ Price}{0.015 \times Mean\ Absolute\ Value\ Deviation}$$

Moving Averages Convergence/Divergence (MACD)

Moving averages convergence/divergence MACD is a centred oscillator that measures a stock's price momentum and identifies trends. Twelve days are commonly used for short term smoothing, twenty-six days for long term smoothing and nine days for a signal.

a) Short term (twelve days) and long term (twenty six days) **smoothing calculation**:

$$current\ EMA_{12} = \frac{2}{12+1} \times (current\ Close - previous\ EMA_{12}) + previous\ EMA_{12}$$

$$initial\ EMA_{12} = SMA_{12}(12\ periods\ Close)$$

$$current\ EMA_{26} = \frac{2}{26+1} \times (current\ Close - previous\ EMA_{26}) + previous\ EMA_{26}$$

$$initial\ EMA_{26} = SMA_{26}(26\ periods\ Close)$$

b) **Moving average convergence/divergence** indicator calculation:

$$MACD\,(12, 26) = EMA_{12}\,(Close) - EMA_{26}(Close)$$

c) **Nine days moving average convergence/divergence** indicator smoothing calculation:

$$Signal\,(9) = EMA_9[MACD(12, 26)]$$

d) Moving average convergence/divergence indicator **histogram calculation**:

$$MACD\ Histogram\,(12, 26, 9) = MACD\,(12, 26) - Signal\,(9)$$

Rate of Change (ROC)

Rate of change ROC is a bounded oscillator that measures a stock's price change speed or momentum. Two hundred and fifty-two days are commonly used for one business year calculation, one hundred and twenty-six for one semester, sixty-three for one quarter and twenty one for one month.

$$ROC\,(21) = \frac{Current\ Close - Close\ 21\ Days\ Ago}{Close\ 21\ Days\ Ago} \times 100$$

Relative Strength Index (RSI)

Relative strength index RSI is a bounded oscillator that measures a stock price trend's strength or weakness. Fourteen days are commonly used for its calculation.

a) Fourteen days average **gain** and **loss**:

$$AG\,(14) = \frac{Sum\ Gains\ Last\ 14\ Days}{14}$$

$$AL\,(14) = \frac{Sum\ Losses\ Last\ 14\ Days}{14}$$

b) Fourteen days **relative strength** calculation:

$$RS\,(14) = \frac{Average\ Gain}{Average\ Loss}$$

c) Fourteen days **relative strength index** calculation:

$$RSI\,(14) = \frac{100}{1+Relative\ Strength(14)}$$

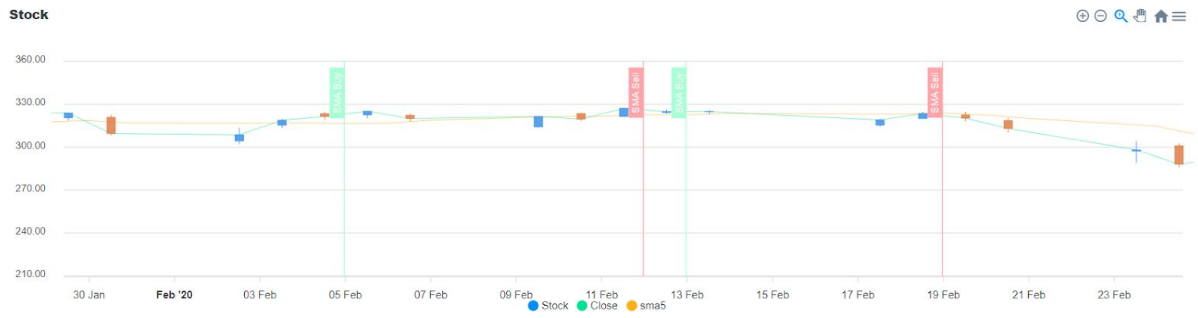
Stock Trading Signals

Moving Averages

Moving averages trading signals occur when there is a crossover among stock close prices and their moving average or between short term and long-term moving averages.

For the graphs below, we have used The 'Apple Inc' stock prices as an example.

Simple Moving Average



For the graph above the technical indicators indicates that if the ***previous close price*** is **less** than the **previous SMA5** and the **Current Close price** is **greater** than the **current SMA5**, then there is a **buy signal.**

On the other hand, if the **previous close price** is **greater** than the **previous SMA5** and the **current close price** is **lower** than the **current SMA5**, then there is a **sell signal.**

Exponential Moving Average

For the graph above the technical indicators indicates that if the ***previous EMA5*** is **less** than the **previous EMA21** and the **Current EMA5** is **greater** than the **current EMA21**, then there is a **buy signal.**

On the other hand if the ***previous EMA5*** is **greater** than the **previous EMA21** and the **Current EMA5** is **lower** than the **current EMA21**, then there is a **sell signal.**

Bollinger Bands Trading Signal

Bollinger bands trading signals occur when there is a crossover among stock close prices and technical indicators' lower and upper bands.

For the graph above the technical indicators indicates that if the ***previous close price*** is **less** than the **previous Lower Band** and the **Current Close price** is **greater** than the **current lower band**, then there is a **buy signal.**

On the other hand, if the **previous close price** is **lower** than the **previous upper band** and the **current close price** is **greater** than the **current upper band**, then there is a **sell signal.**

Parabolic Stop and Reverse Indicator

Parabolic stop and reverse trading signals occur when there is a crossover among stock close prices and corresponding technical indicators.

For the graph above the technical indicators indicates that if the ***previous close price*** is **less** than the **previous SAR** and the **Current Close price** is **greater** than the **current SAR**, then there is a **buy signal.**

On the other hand, if the **previous close price** is **greater** than the **previous SAR** and the **current close price** is **lower** than the **current SAR**, then there is a **sell signal.**

Rate of Change

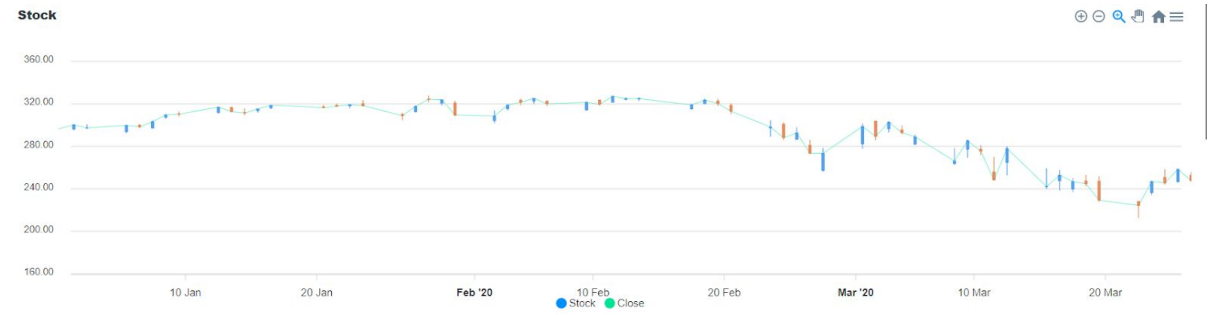
Rate of change trading signals occur when there is a crossover among technical indicators and its upper and lower bands.

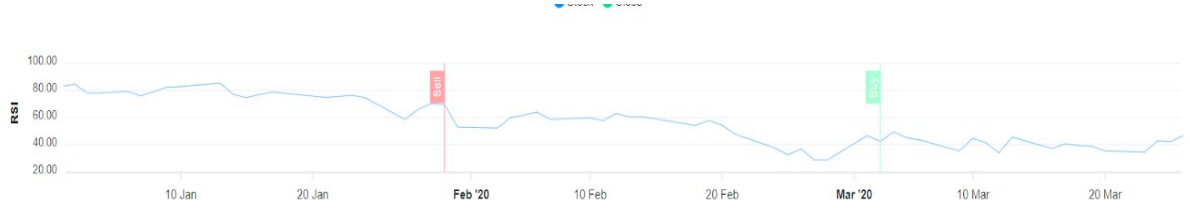
For the graph above the technical indicators indicates that if the **previous ROC of 21 days** is **less** than **-5** and the **Current ROC of 21 days** is **greater** than **-5**, then there is a **buy signal.**

On the other hand if the **previous ROC of 21 days** is **less** than the **+5** and the **Current ROC of 21 days** is **greater** than +5, then there is a **sell signal.**

Relative Strength Index

Relative strength index trading signals occur when there is a crossover among technical indicators and its upper and lower bands.





For the graph above the technical indicators indicates that if the **previous RSI of 14 days** is **less** than **30** and the **Current RSI of 21 days** is **greater** than **30**, then there is a **buy signal.**

On the other hand, if the **previous RSI of 14 days** is **less** than the **70** and the **Current RSI of 14 days** is **greater** than 70, then there is a **sell signal.**

<u>Moving Averages Convergence/Divergence</u>

Moving averages convergence/divergence trading signals occur when there is a crossover among technical indicator and its signal or centreline.

For **Signal Crossover Trading Signals**, if the **previous** MACD for 12 and 26 days is lower than the **Signal of 9 days** and the **current** MACD of 12 and 26 days is greater than the **signal of 9 days**, then there is a **buy** signal. Whereas if the **previous MACD for 12 and 26 days** is **greater** than the **Signal of 9 days** and the **current MACD of 12 and 26 days** is **lower** than the **signal of 9 days**, then there is a **sell signal**.

For **Centerline Crossover Trading Signals**, if the **previous** MACD for 12 and 26 days is **lower** than the **0** and the current MACD of 12 and 26 days is greater than the **0**, then there is a **buy** signal. Whereas if the **previous MACD for 12 and 26 days** is **greater** than the **0** and the **current MACD of 12 and 26 days** is **lower** than the **0**, then there is a **sell signal.**

Reference:

1. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. (2017). Attention is All you Need. NIPS.
2. Harry M., Merton H., William F. (1990). The Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel 1990. (n.d.). Retrieved from https://www.nobelprize.org/prizes/economic-sciences/1990/press-release/
3. Neumann, J. V., & Morgenstern, O. (1953). Theory of games and economic behavior. Princeton: Princeton Univ. Pr.