DL3

FYP Final Report

A System for Predicting Stock Price and Offering Financial Advice

by

Aadhar Agarwal, Jose Luis Amador Jaime, Leung Ka Wing (Michael) and Luk Yin Chai (Dragos)

DL3

Advised by

Prof. Dik-Lun, Lee

Submitted in partial fulfilment

of the requirements for COMP 4981 and CPEG 4901

in the

Department of Computer Science and Engineering

and the

Computer Engineering Program

The Hong Kong University of Science and Technology

2020-2021

Date of submission: April 14, 2021

Abstract

In this Final Year Project, we had the objective of building a system for predicting stock price and offering financial advice. This, as we thought, there was a lack of investment tools that were; insightful to the user, easy to use for an inexperienced investor, and innovative in leveraging machine learning models for stock price prediction.

We created a web application using Django and React web frameworks that leveraged a multivariate CNN model for stock price prediction and optimized portfolio investment allocation through Markowitz a

Through our empirical approach, we found that CNN models were more accurate in capturing stock price trends and making prediction when compared to other models such as RNN-LSTM. Moreover, we found an interesting approach to ingrate stock price predictions with established financial literature on portfolio optimization.

Using the mean-variance optimization technique with the methods of volatility minimization, max sharpe ratio, and Markowitz portfolio optimization with risk aversion, several portfolios were created. These portfolios all outperformed the index in terms of annual returns. Showing the opportunity to make profits.

1 Introduction

1.1 Overview

In the finance business, there exist a variety of tools targeted at traders and investors that aid them in performing analysis and making decisions. For example, the app/website TradingView offers the user access to charting and trading for stocks as well as several tools for technical analysis. Other platforms such as Bloomberg offer products such as their dashboard which not only provides historical data, but also recent news related to the firm. These products are mainly aimed at professionals working in the financial sector.

Consequently, in the field of machine learning and artificial intelligence, stock price prediction is a popular topic of research with several different models currently in development. However, similar to the case with trading platforms, the use of these models is mainly restricted to academia. The lack of integration means that if someone using a platform like TradingView or Bloomberg wishes to use results form a model, they would have to look at the data separately and then go back to the platform. This issue is further exacerbated due to the high level of technical knowledge needed to understand the stock prediction models and use them effectively.

Finally, most of the available trading platforms only display information, and do not provide advice. This is because they are designed with existing investors and traders in mind who will perform the analysis and decide on purchases. However, this results in limited options for people who may be interested in investing but have limited knowledge of financial markets. Platforms such as E-toro allow a user to mimic the decisions of another user and make profits, but the user does not have any agency in the investments and is not investing appropriately to their risk appetite. By combining the data from a stock prediction model with an interface that is able to provide financial advice, the barrier to entry for intelligent investing can be greatly lowered.

This FYP aims to create a system for stock prediction and offering financial advice aimed at people with limited to no experience in investing or trading. It will provide a graphical interface for the stock price history and portfolio management. The financial advice will be provided in the form of weights for the user’s portfolio.

1.2 Objectives

This project aimed to build a web application that allows users to get predictions on stock prices as well as advice on how to maximize their return on investment. This project involves the use of machine learning for prediction, web technologies for the development of the frontend and backend, and implementation of portfolio optimization strategies to provide financial advice. The outcome of the project shall be achieved by reaching the following goals:

1) Achieving a satisfactory level of accuracy on the predictions generated from the machine learning models.
2) Using the predictions to offer investment strategies- including buy, sell, hold, and diversify- that result in a return on investment that is higher than the market return.
3) Ensuring the web application provides an intuitive and personalized interface

The technical challenges involved in reaching a satisfactory level of performance were mainly related to the selection of the algorithms used to make the predictions as well as the training of optimization of the various machine learning models to maximize accuracy and precision across all stocks. This is because it is unfeasible to create a new model for every different stock the user chooses at it would take far too long and be very unfriendly to the end user. In particular, choosing the ideal hyperparameters was a difficult task. This issue was countered by separating the stocks based on their behavior and using different models for the different stock types.

The technical challenges involved in providing financial advice were related to the selection of the portfolio optimization method. Given the intention to design for novice investors, three methods were selected. Maximum Sharpe ratio, Minimum Volatility, and Markowitz Portfolio Optimization with user defined risk aversion. The differences between the methods are elaborated on in section *2.1.4.*

The key challenge for the third goal was ensuring that the results and predictions are provided quickly and smoothly as it is imperative that the frontend and the backend are able to communicate easily. This was handled through the use of light and fast frameworks such as Django for the web development.

1.3 Literature Survey

1.3.1 Recurrent Neural Network

A Recurrent Neural Network is a class of neural networks with its units being recurrently connected. This enables them to have an internal memory to process a sequence of inputs. This allows them to process the inputs and recognize their long-term dependencies in the time-series data of stock prices. A shortcoming is that RNN are not able to hold this memory for an extended period of time, as the Vanishing Gradient descent problem causes that after every iteration in the neural net, the memory and data it has gets vanished going deeper. A way to overcome this is to use Long Short-Term Memory cells instead of the typical neuron-like cells, this to learn more accurately the long-term dependencies in the data.

In [1] it was found that RNN is better at capturing non-linear relationships in stock price data as it was compared against the linear of model ARIMA, and the RNN error percentage was much lower at 3.90% compared to 31.91%. Moreover, it was also found that LSTM performed slightly better due to its ability to hold its memory better. However, the paper also found that CNN performed better as it was able to capture better dynamic changes in trends in the stock price data.

Furthermore, in [2] found that Recurrent Neural Networks (RNN) with Long Short-Term Memory cells performed better than traditional Machine Learning Algorithms, such as regression, support vector machine, random forest, feed forward neural network and backpropagation as RNN-LSTM had much less prediction error when evaluating the models.

1.3.2 Convolutional Neural Networks

Convolutional neural network (CNN) is another class of deep neural network that is being studied for stock price prediction in recent years. Although CNNs are more commonly used to solve for problems related to computer vision, they are recently found to be performing well for time series forecasting problems too. In fact, [3] found that RNNs, which are generally associated with sequential data, could be outperformed by a simple CNN for sequential modeling.

As for the application of CNN to stock price prediction, several studies have been done to examine the performance of CNN, or an ensemble of CNN and other models, against other methods. These studies generally found that CNNs offer a novel direction in tackling the problem of stock price prediction. For example, in [4], a three-dimensional tensor is used to represent the data of the different features of several markets across a period of time, and input into a CNN. Similarly, [5] found that a 1D CNN can predict stock prices as accurately as some LSTM architectures, while only a fraction of the time is needed to train the CNN model.

An alternate category of studies related to applying CNN to the prediction of stock prices are [6] and [7]. Both studies took stock chart images as input to the CNN; for example, [6] experimented with using different kinds of charts, including a candlestick chart, a line chart, a bar chart, or even a fusion of the different charts as images to be input into the model.

In addition to using CNN as a single model, [8] examined the possibility of combining CNN with a bi-directional LSTM. The CNN is used to extract high level features before passing in data to the LSTM; the rationale behind this was that by performing convolutions on the data, the CNN will be able to exploit the time structure of the input data.

1.3.3 Time Series Forecasting

Aside from deep-learning based approaches, traditional regression models have shown great effectiveness in forecasting values for time series data such as stock prices [9]. Traditionally, the method used for times series data regression is the Autoregressive Integrated Moving Average (ARIMA) model [9].This model involves using the forecast outcome and the sum of the forecast errors to provide forecasts that should become more accurate the more data there is available.

However, recent interest in the field has led to the development of a new model for making forecasts using regression called Prophet. Prophet is a time series forecasting model made by Facebook and compiled into open source libraries for use in both Python and R [10]. Prophet is an additive regression model with 4 key components; A linear or logistic growth curve trend, a yearly seasonal component modeled using Fourier series, a weekly seasonal component modeled with dummy variables, and the user provided list of important holidays. The benefit of such a model over a traditional regression model is that it leverages both the objectivity of the model as well as “insights” and “hunches” that industry professionals may have observed in their experiences in their industry.

However, the accuracy and precision of the forecasts created by Prophet is lower than the results obtained from a LSTM model. This results in lower return on investment

(ROI) than LSTM models [11]. Despite this, Prophet models can still be useful as they can be used alongside LSTM models as a part of an ensemble model to reduce errors. In addition, the Prophet model can also be used as an input layer for the LSTM model in order to reduce the impact of random event of the predicted outcomes [12]. For these reasons we will be using a linear regression model – specifically Prophet – alongside our deep learning models.

1.3.4 Summary

In summary, this FYP will aim to use methods from both Deep Learning as well as traditional regression to further extend of existing models and generate accurate predictions. In addition to the above-mentioned methods, we shall also use Natural Language Processing (NLP) to perform sentiment analysis which will then be used to make predictions.

2 Methodology

2.1 Design

The Design Phase of the project started in July and while we are still iterating the design of some of the aspects, we have finalized the designs for the others.

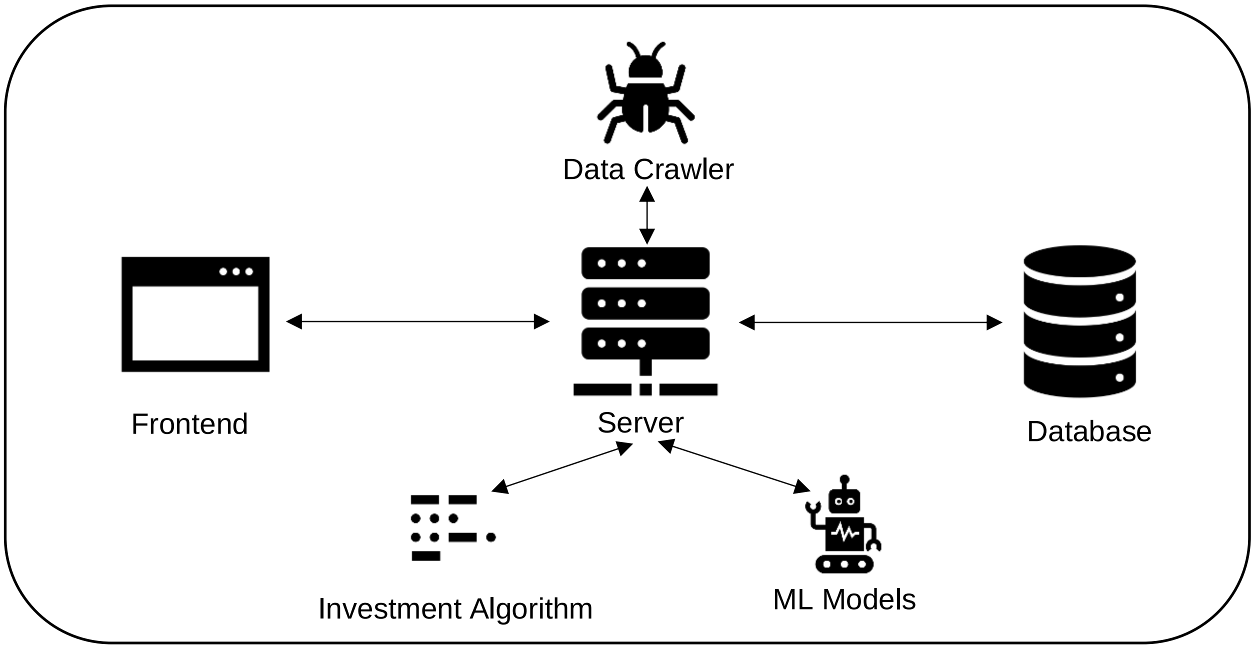


Figure 1. Shows the high-level overview of the different aspects that constitute the project.

2.1.1 Data Crawler Design

We designed a data crawler that periodically crawls data and stores it in a database for each stock on the market, which includes: 1) market indicators, 2) financial statements, and 3) related news.

Stocks traded in the Hong Kong Exchange shall be included.

Market indicators include the following daily data: opening price, closing price, highest price, lowest price, trade volume, turnover, relative strength index (RSI), Simple Moving Average (SMA) and moving average convergence divergence (MACD), etc.

Financial statements include the following yearly data: profit margin, return on assets (ROA), current ratio, etc.

Related news will include the news title, news content, date of publishment, and positive vs negative votes. It is important that all data sources come from credible sources.

2.1.2 Database Design

The data is preprocessed before being stored in the database. Each of the set of data crawled is stored in a separate table, i.e. the market indicators, financial statements, and related news are stored in separate tables.

There is also a user database, where information for a specific user is stored. The table contains the following columns: username, password, and risk aversion level.

Column Name	**Type**	**Description**	**Remarks**
id	Integer	Primary key	
user_id	Integer	Foreign key in the table auth_user	

risk	Integer	The risk level from 0 to 47; higher means a higher risk tolerance	

Figure 2. Shows the different features of the user database as well as their type and description.

For the user to select which stocks to include in their portfolio, another table is used to store the selected stocks. This table has the following columns: stock symbol, number of shares, and average purchase price. After some issues with our initial choice of database, it was decided that the database must also be supported by the frontend to ensure minimal issues.

2.1.3 Model Selection

We conducted experiments to select a model or an ensemble of models for stock price prediction. All the data is used after preprocessing and normalization. Models that we tested include: weighted moving average, linear regression, ADAboost, and LSTM.

1. Weighted Moving Average: The weighted moving average is calculated by taking the close price of the last N days and weighing them by how recent they are. I.e., the closing price of the previous day will have a higher weight than the closing price from a few days ago. By taking the mean of the weighted price, the future price can be predicted. The main issue with this method is that there is a significant lag between an event occurring and the moving average being affected.

2. Linear Regression: Linear regression assumes that there is a linear relationship between the various features and the target value we are trying to predict. It then uses linear algebra to create the best fit linear equation that matches the data. The issue with this is that it is unable to capture non-linear relationships between the variables and assumes that all variables are independent, which they often are not.
3. ADABoost: ADABoost is a gradient booting technique that takes several independent models and combines their outputs to give a final result. It then modifies the hyperparamaters of the various models to ensure the output causes the least errors. The issue with this method is that it is often effective on the test data but performs worse on real world live data. This is because ADABoost models tend to suffer from overfitting.
4. LSTM: The final model that we decided to go for was the LSTM. The LSTM consists of many different layers which are then combined into a final dense layer which provides the output. The input gate and the forget gate of the LSTM model can control some old memories to be dropped and new memories to be added to the model. This allows the model to memorize older data compare to simple RNN model which makes it to be more capable of detecting long-term dependencies. This very advantage of the model makes it suitable for predicting our stocks data because our stocks data contains 5 years data of the stock.

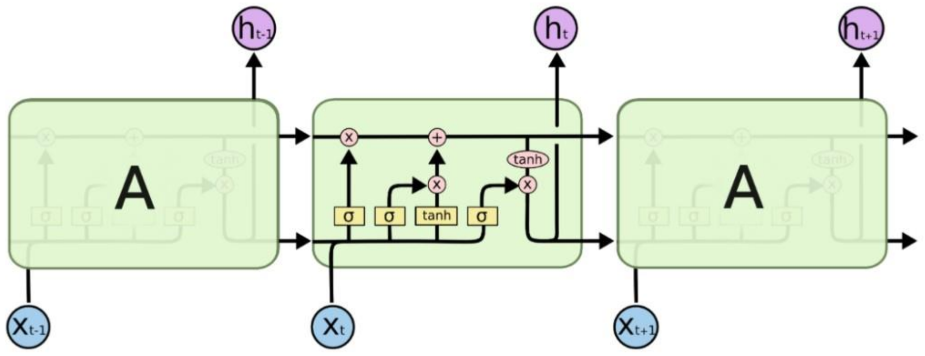


Figure 3. Shows the features and elements of a typical LSTM model.

2.1.4 Portfolio Optimization Algorithm Design

The algorithm is used for suggesting how a user can rebalance their portfolio according to their target risk level. Based on the user’s risk aversion, the algorithm is able to suggest different stocks that the user can add to or remove from their investment portfolio. For example, the algorithm can adjust the overall standard deviation and return of a portfolio according to a user’s preference. Also, by utilizing the prediction results from the models, the algorithm can suggest stocks for the user to invest in.

The design of the portfolio optimization algorithm in relation to the machine learning model is show in the following diagram:

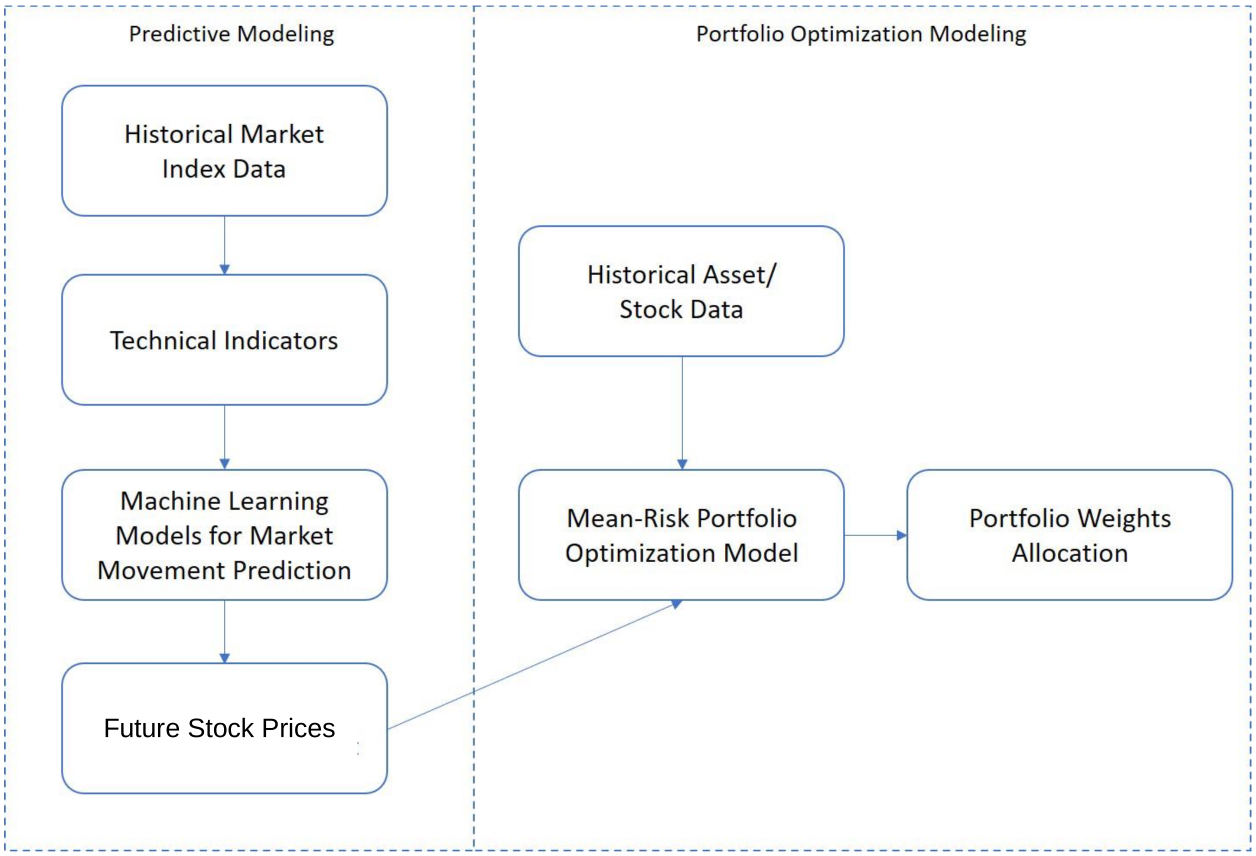

Predictive Modeling
Portfolio Optimization Modeling
Historical Market Index Data
Technical Indicators
Machine Learning Models for Market Movement Prediction
Future Stock Prices
Historical Asset/ Stock Data
Mean-Risk Portfolio Optimization Model
Portfolio Weights Allocation

Figure 4. Shows the relationship between the ML models and the portfolio

The portfolio optimization and allocation algorithm will consist of three parts; Asset retrieval, where the list of possible stocks to allocate is collected; portfolio optimization, where the weights are calculated, and output to database, where the weights are stored for the user.

Given the assumption that our users are unfamiliar with quantitative finance, we chose three relatively simple, but effective portfolio optimization methods based on the classic mean-variance portfolio model (Markowitz (1952),): Minimum Volatility Portfolio, Maximum Sharpe Ratio portfolio, and Markowitz portfolio with risk aversion.

1. **Minimum Volatility Portfolio:** The mean-variance model tries to find the optimal trade off between the reward and risk by using the mean and variance of the

portfolio returns as their measures. In effect, by optimizing for certain constraints, different portfolio weights can be generated. The minimum volatility portfolio is the portfolio that has the least variance with the given stocks. There has been research done that shows that Global Minimum Variance (GMV) portfolios like this can actually outperform mean variance optimized portfolios in certain situations. Mathematically, for N stocks, it can be expressed as finding the solution to:

$$min \sum_{i=1}^{N} \sum_{j=1}^{N} x_i x_j \sigma_{ij}$$

Where x_i and x_j are the mean returns for the stocks and σ is the covariance between the two stocks.

2. **Maximum Sharpe Ratio Portfolio:** Similar to the minimum volatlility portfolio, the Max. Sharpe Ratio is also another optimized version of the mean-variance portfolio. The Sharpe ratio is essentially a measure of the excess return on the portfolio relative to its risk. The maximum Sharpe ratio portfolio has the highest return per unit risk. Mathematically, it can be calculated as the solution to:

$$Max\ Shrape\ Ratio = \max\left(\frac{R_p - R_f}{\sigma_P}\right)$$

Where R_p is the return of the portfolio, R_f if the risk-free rate of return (around 2%), and σ_p is the volatility of the portfolios' excess return.

3. **Markowitz portfolio with risk aversion:** The Markowitz portfolio optimization is the classic mean-variance optimization technique. However, by using the risk aversion of the user as one of the coefficients in the equation, we can account for

their risk appetite and make a recommendation more suited to them. Mathematically, it can be calculated as the solution to:

$$z = \min\left(\sigma_P^2 - \lambda R_P\right)$$

Where σ_P^2 is the volatility of the portfolio, λ is the risk aversion of the investor, and R_P is the return of the portfolio.

By using these three methods, the user is provided with different ways to optimize their portfolio according to their needs.

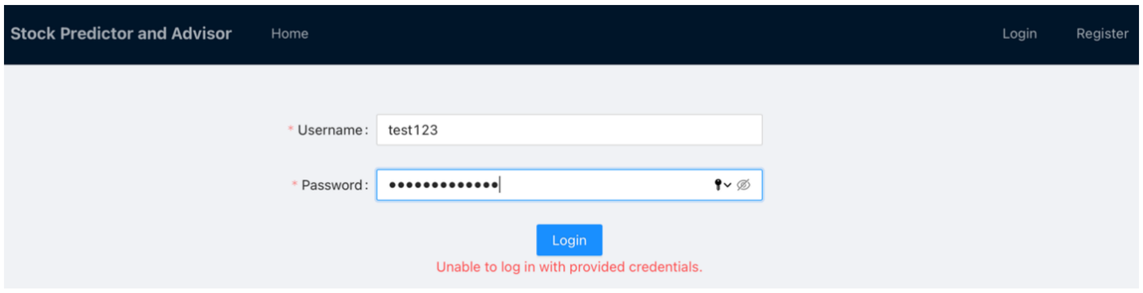
2.1.5 Backend Server Design

The backend consists of the data crawler, the prediction models, the portfolio optimization algorithm, the logic in handling user data, and also databases.

2.1.6 Frontend Design

The frontend will be a web application that can be accessed through a web browser. The application shall contain the following pages:

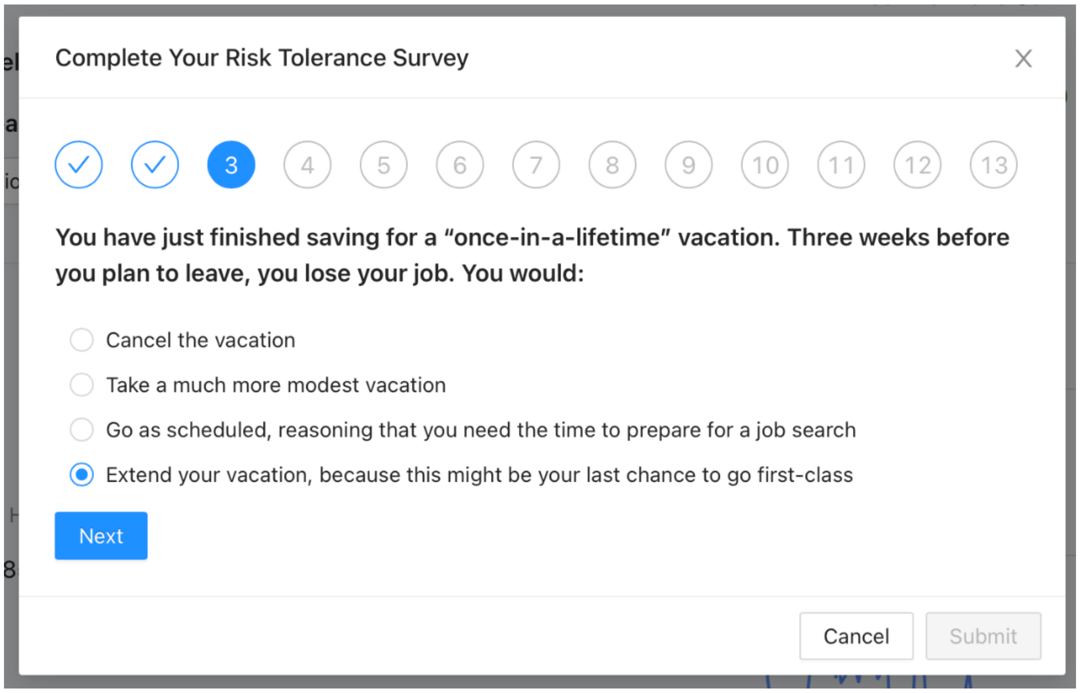
1) A page for stock price prediction that can be accessed by the public
2) A page for the user to login/register and to select their risk aversion level



test123

Unable to log in with provided credentials.

Figure 5. Shows the login page for the user.



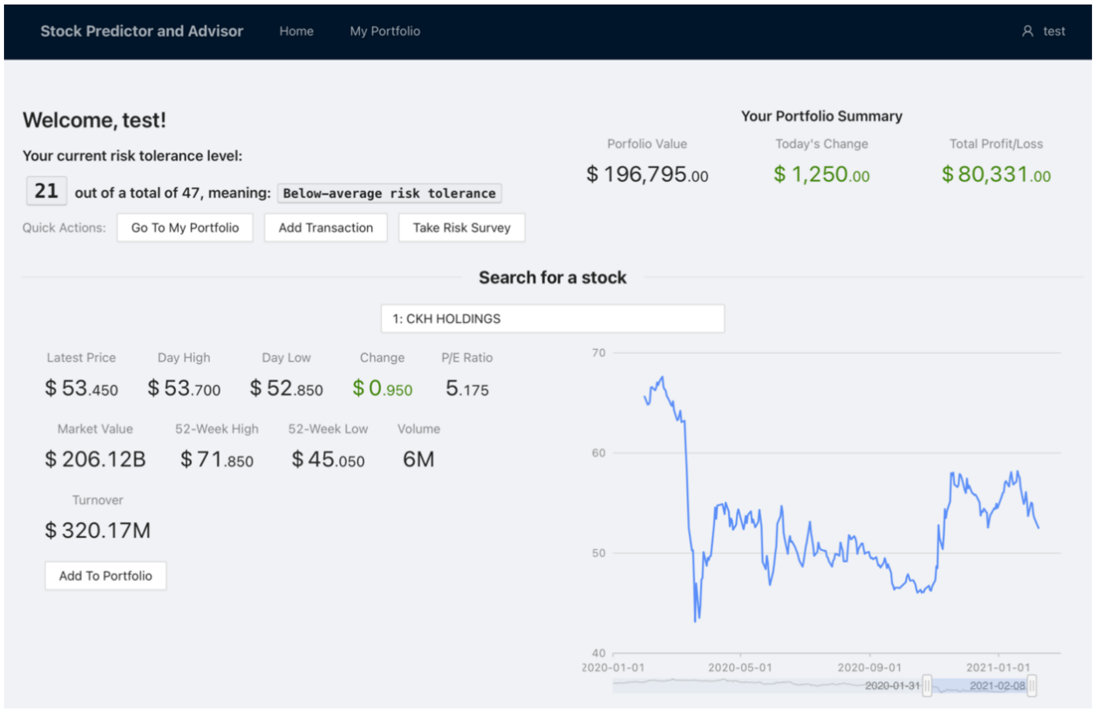
Complete Your Risk Tolerance Survey

You have just finished saving for a "once-in-a-lifetime" vacation. Three weeks before you plan to leave, you lose your job. You would:

- Cancel the vacation
- Take a much more modest vacation
- Go as scheduled, reasoning that you need the time to prepare for a job search
- [✓ Extend your vacation, because this might be your last chance to go first-class]

Figure 6. Shows the risk selection page.

3) A page for the user to view their stock portfolio and to view suggested investment strategies including whether to diversify.



Stock Predictor and Advisor
Home
My Portfolio
test

Welcome, test!

Your current risk tolerance level:

21 out of a total of 47, meaning: `Below-average risk tolerance`

Quick Actions: Go To My Portfolio
Add Transaction
Take Risk Survey

Your Portfolio Summary

Porfolio Value	Today's Change	Total Profit/Loss
$ 196,795.00	$ 1,250.00	$ 80,331.00

Search for a stock

1: CKH HOLDINGS

Latest Price	Day High	Day Low	Change	P/E Ratio
$ 53.450	$ 53.700	$ 52.850	$ 0.950	5.175

Market Value	52-Week High	52-Week Low	Volume
$ 206.12B	$ 71.850	$ 45.050	6M

Turnover

$ 320.17M

Figure 7. Shows the home page of the webapp.

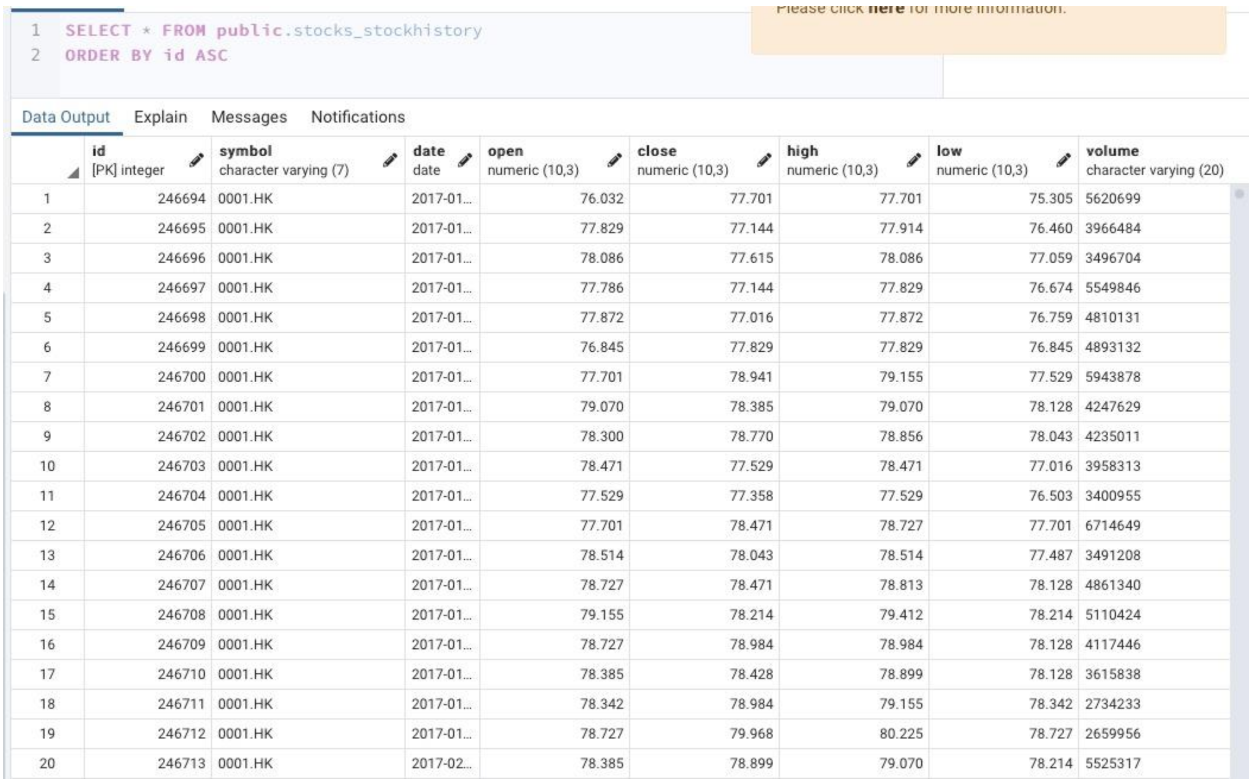
2.2 Implementation

The Implementation Phase includes the following aspects:

2.2.1 Data Crawler Development

We first experimented with the use of the HKEX for our stock data. We were able to retrieve the data, but the data was limited. On the other hand, we found that the yfinance library that scraps data from Yahoo Finance offered more data. Moreover, we found that Alpha Vantage API in addition to price data it offered data on technical indicators such as Simple Moving Average (SMA) and Relative Strength Index (RSI) as well. As such, we decided it would be better to switch to yfinance and Alpha Vantage API instead.

To get the list of all of the stocks listed in the HKEX, we used yfinance [13]. We then used the Python library “pandas” to convert rows in the into a dataframe; after filtering the dataframe for stocks listed in the main market, we inserted the data into a table in our database.



```
SELECT * FROM public.stocks_stockhistory
ORDER BY id ASC
```

	id [PK] integer	symbol character varying (7)	date date	open numeric (10,3)	close numeric (10,3)	high numeric (10,3)	low numeric (10,3)	volume character varying (20)
1	246694	0001.HK	2017-01…	76.032	77.701	77.701	75.305	5620699
2	246695	0001.HK	2017-01…	77.829	77.144	77.914	76.460	3966484
3	246696	0001.HK	2017-01…	78.086	77.615	78.086	77.059	3496704
4	246697	0001.HK	2017-01…	77.786	77.144	77.829	76.674	5549846
5	246698	0001.HK	2017-01…	77.872	77.016	77.872	76.759	4810131
6	246699	0001.HK	2017-01…	76.845	77.829	77.829	76.845	4893132
7	246700	0001.HK	2017-01…	77.701	78.941	79.155	77.529	5943878
8	246701	0001.HK	2017-01…	79.070	78.385	79.070	78.128	4247629
9	246702	0001.HK	2017-01…	78.300	78.770	78.856	78.043	4235011
10	246703	0001.HK	2017-01…	78.471	77.529	78.471	77.016	3958313
11	246704	0001.HK	2017-01…	77.529	77.358	77.529	76.503	3400955
12	246705	0001.HK	2017-01…	77.701	78.471	78.727	77.701	6714649
13	246706	0001.HK	2017-01…	78.514	78.043	78.514	77.487	3491208
14	246707	0001.HK	2017-01…	78.727	78.471	78.813	78.128	4861340
15	246708	0001.HK	2017-01…	79.155	78.214	79.412	78.214	5110424
16	246709	0001.HK	2017-01…	78.727	78.984	78.984	78.128	4117446
17	246710	0001.HK	2017-01…	78.385	78.428	78.899	78.128	3615838
18	246711	0001.HK	2017-01…	78.342	78.984	79.155	78.342	2734233
19	246712	0001.HK	2017-01…	78.727	79.968	80.225	78.727	2659956
20	246713	0001.HK	2017-02…	78.385	78.899	79.070	78.214	5525317

To crawl historical stock price data from Yahoo Finance, we used a Python library called "yfinance". Given a start date and an end date, it gets the daily price data for the requested time period including the open, high, low, and close (OHLC), as well as the volume traded in the format of a dataframe. Similar to how we dealt with the list of stocks listed in HKEX, we stored the data fetched into a table in our database. Currently, we are storing the stock price data of the 52 composite stocks of the Hang Seng Index starting from 2019.

Also, we utilized Alpha Vantage API to get stock data from the Dow Jones Industrial Average Index. For each of the DJIA components, the adjusted close price, the simple moving average (SMA), and the relative strength index (RSI) were obtained. As described previously, this data was stored in the PostgreSQL database and processed with Pandas and NumPy libraries.

The figure below is a visualization of the dataset for the adjusted close price data for each of the 30 components in the Dow Jones Industrial Average ("MMM", "AXP", "AAPL", "BA", "CAT", "CVX", "CSCO", "KO", "DOW", "XOM", "GS", "HD", "INTC", "IBM", "JNJ", "JPM", "MCD", "MRK", "MSFT", "NKE", "PFE", "PG", "RTX", "TSLA", "TRV", "UNH", "VZ", "V", "WBA", "WMT")

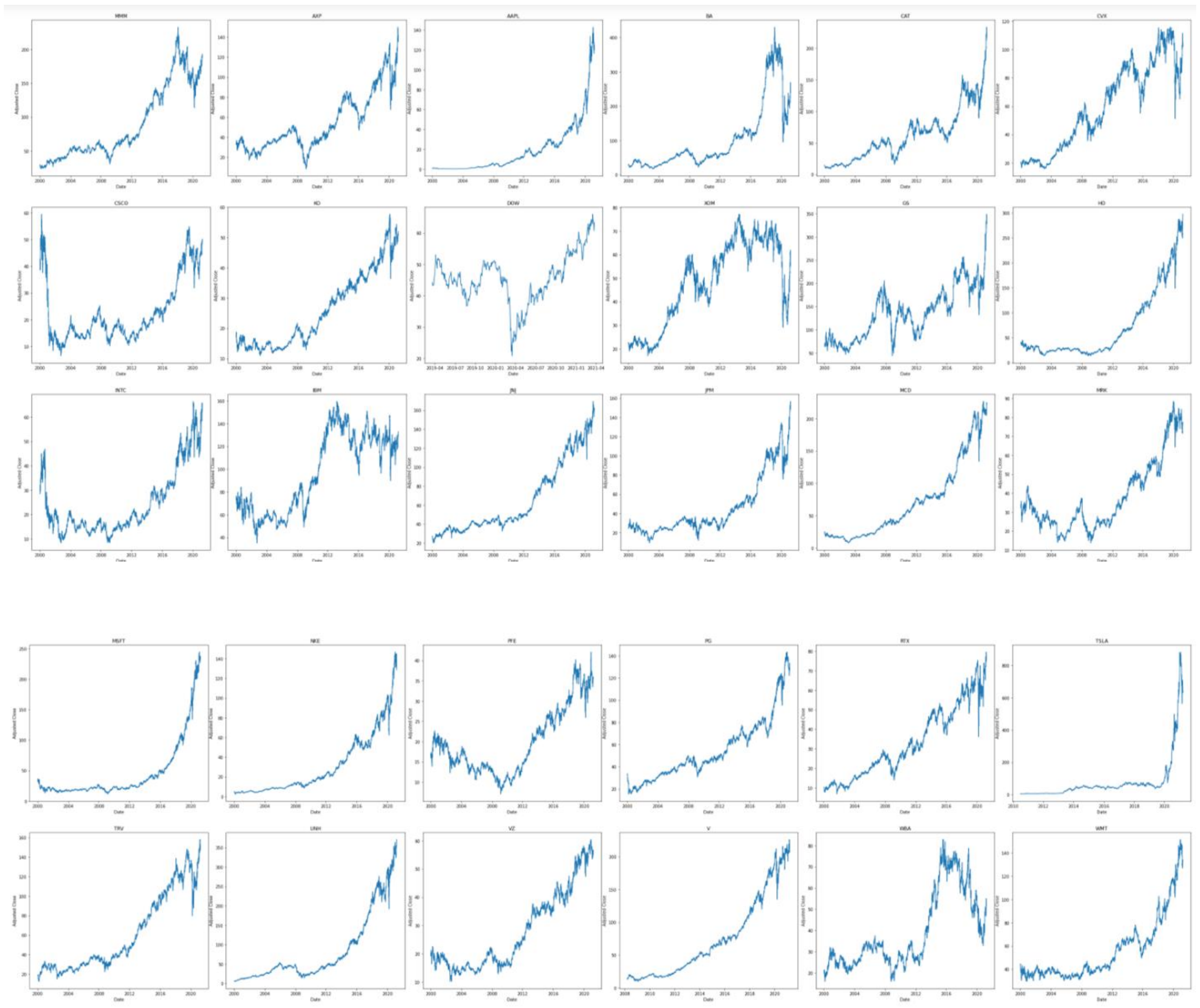


Figure 9. Shows the historical stock data of all the stocks in the DJIA

2.2.2 Database Implementation

By default, Django, the web framework we are using to implement the backend server, uses SQLite as the database. However, given that SQLite is more suitable for lightweight development only, we opted for the first officially supported database by Django, PostgreSQL [14]. During our development phase, we each setup a PostgreSQL database on our local machine and connected to the database via the APIs provided by Django.

To create tables inside the database, we first defined several Python classes with Django. After defining the classes that we need, the tables are automatically created in the database by running the commands “makemigrations” and “migrate” in Django.

Each class, when created, will automatically have a primary key field named “id”. It should also be noted that each class is created under a Django “app”, so in the database, the table names are automatically prepended with the app name in the following format: {appName_className}. The table for the class auth_user is shown below. Tables for all classes can be found in appendix B.

auth_user

Column Name	**Type**	**Description**	**Remarks**
id	Integer	Primary key	

password	Character	Hashed and encrypted	
username	Character	Unique username for each user	
email	Character	Unique email for each user	

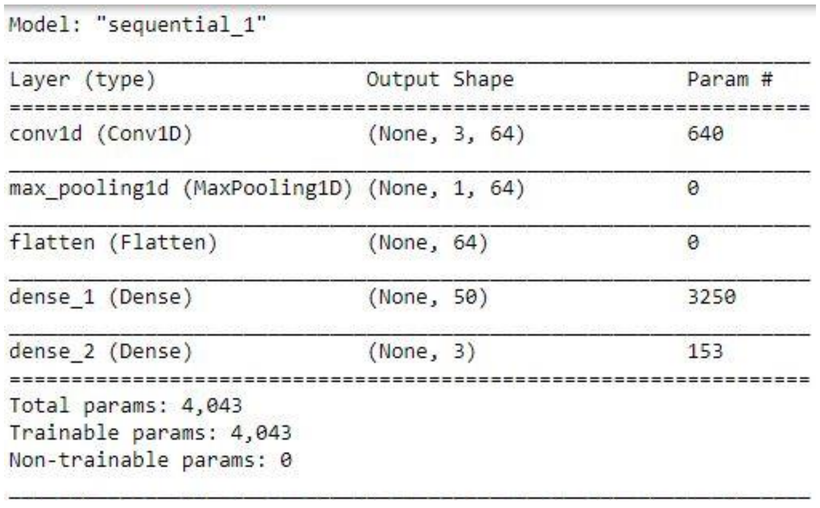
Figure 10. Shows the class table for auth_user in django

2.2.3 Model Implementation

Based on our design, we evaluated each model and select features and search for hyperparameters in order to optimize the models and the prediction results.

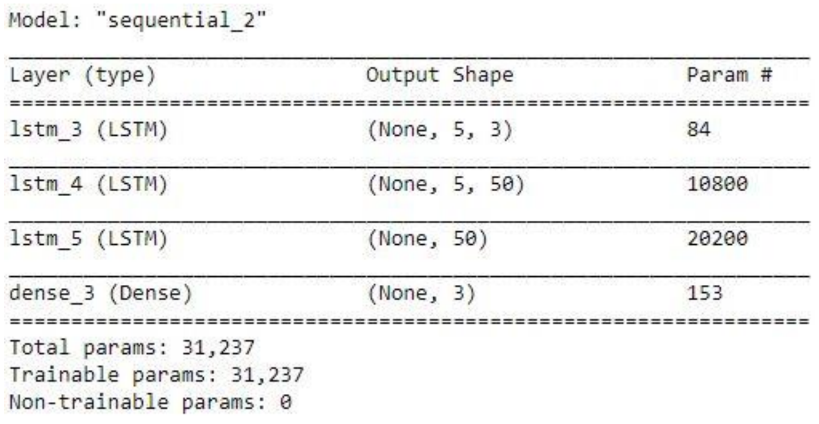
We implemented a univariate LSTM model, a univariate CNN model, a multivariate LSTM model and a multivariate CNN model. The models are currently implemented using TensorFlow with Keras as it runs smoothly in python through jupyter notebooks. By using only one language for the majority of our project, we were able to reduce the time needed to integrate the different modules. In the univariate model, we used the adjusted close price to predict the adjusted close price. In the multivariate model, we used the adjusted close price, RSI and simple moving average to predict the adjusted close price of the stock. The models’ structure are as follows:

CNN (Multivariate)



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 3, 64)             640
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 1, 64)             0
_________________________________________________________________
flatten (Flatten)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 50)                3250
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 153
=================================================================
Total params: 4,043
Trainable params: 4,043
Non-trainable params: 0
_________________________________________________________________
```

LSTM (Multivariate)



```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 5, 3)              84
_________________________________________________________________
lstm_4 (LSTM)                (None, 5, 50)             10800
_________________________________________________________________
lstm_5 (LSTM)                (None, 50)                20200
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 153
=================================================================
Total params: 31,237
Trainable params: 31,237
Non-trainable params: 0
```

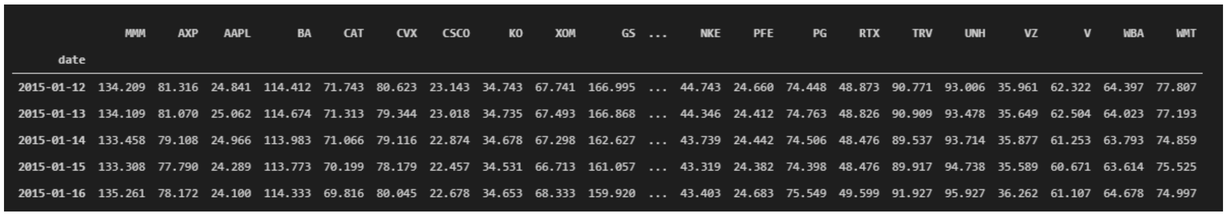
Figure 11. Shows the build of the two ML Models

2.2.4 Portfolio Optimization Algorithm Implementation

The portfolio optimization is implemented as a python script which thakes thes Historical stock data and the and predicted stock prices provides functions for efficiently calculating the portfolio weights. Then the weights are stored in for the webapp to use.

1. **Collecting the data**: The historical stock data stored in the database has eight attributes (open, high, low, close, adjusted_close, volume, dividend, and split coefficient). In addition, the predicted price database has the attributes id, _state,

symbol, day, date_created, predicted_price. However, the portfolio asset script only runs on a dataframe with date as the index and the price of each of the stocks as the attributes. As such, we have to pre-press the colums of the dataframs and get them into the correct format. For the historical data, we take the adjusted_close price, and the predicted_price for the predictions. Then, we need to concatenate the two dataframes together to get one dataframe with all the data.



```
                MMM     AXP    AAPL       BA     CAT     CVX    CSCO      KO     XOM       GS  ...     NKE     PFE      PG     RTX     TRV     UNH      VZ       V     WBA     WMT
      date
2015-01-12  134.209  81.316  24.841  114.412  71.743  80.623  23.143  34.743  67.741  166.995  ...  44.743  24.660  74.448  48.873  90.771  93.006  35.961  62.322  64.397  77.807
2015-01-13  134.109  81.070  25.062  114.674  71.313  79.344  23.018  34.735  67.493  166.868  ...  44.346  24.412  74.763  48.826  90.909  93.478  35.649  62.504  64.023  77.193
2015-01-14  133.458  79.108  24.966  113.983  71.066  79.116  22.874  34.678  67.298  162.627  ...  43.739  24.442  74.506  48.476  89.537  93.714  35.877  61.253  63.793  74.859
2015-01-15  133.308  77.790  24.289  113.773  70.199  78.179  22.457  34.531  66.713  161.057  ...  43.319  24.382  74.398  48.476  89.917  94.738  35.589  60.671  63.614  75.525
2015-01-16  135.261  78.172  24.100  114.333  69.816  80.045  22.678  34.653  68.333  159.920  ...  43.403  24.683  75.549  49.599  91.927  95.927  36.262  61.107  64.678  74.997
```

Figure 12. Shows the dataframe fed into the portfolio algorithms

2. **Portfolio Optimization:** After the dataframe has been created, the mean and the variance can then be calculated. For the variance, we used the modern method of Ledoit Wolf shrinkage matrix which provides better performance than simple covariance. The most important aspect to decide on was the returns of the stocks. It was decided that the exponential moving average (EMA) retuns should be used as they provide more weight to recent samples. This ensures, that the predicted prices from the machine learning model are given appropriate weight. Below, the difference in portfolios made using exponential moving average and historical mean prices are shown.

Portfolio with historical mean prices **Portfolio with exponential moving avg.**

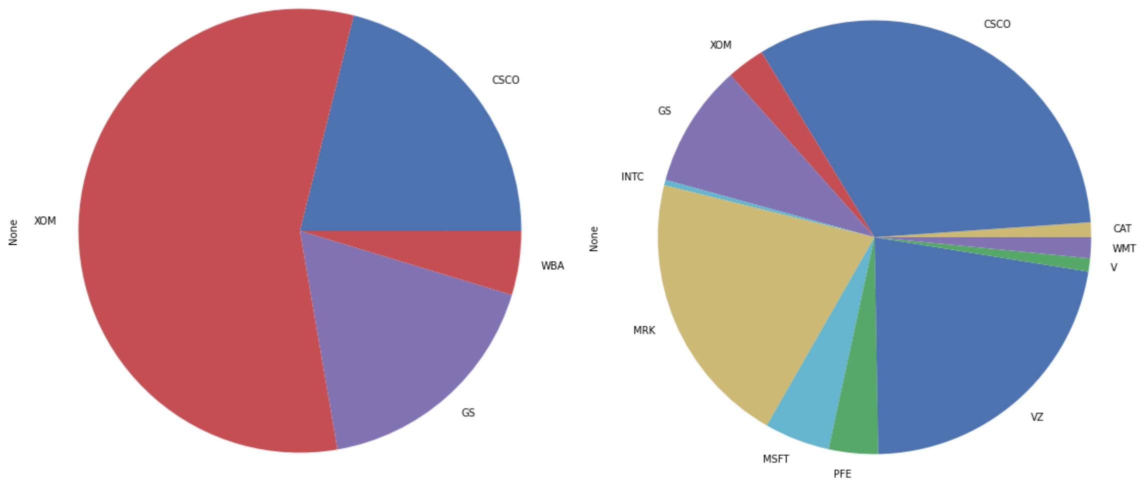


Figure 13. Shows the portfolios with different returns

3. **Storing the results:** The results of the portfolio optimization are in the form of and ordered dictionary with the tickers as the key and the weights as the values. A Django app is created to take the dictionary and store it into the database.

2.2.5 Backend Server Implementation

Main Backend Project and Applications

We selected Django as the framework to use when implementing the backend. We chose Django over Flask because Django provides a lot of out-of-the-box features, so it makes development faster. Several applications were created inside the Django project for different needs, and each app may contain one or more classes:

- **App 1: users**
  - Profile
- **App 2: stocks**
  - StocksInfo

- StocksInfoDJIA
- StockHistory
- StockDailyAdjustedPrice
- StockSimpleMovingAverage
- StockRelativeStrengthIndex
- StockSQZMOM

- **App 3: transactions**
    - Transaction
- **App 4: ml_models**
    - StocksPrediction
    - RMSE
- **App 5: portfolio**
    - PortfolioWeights

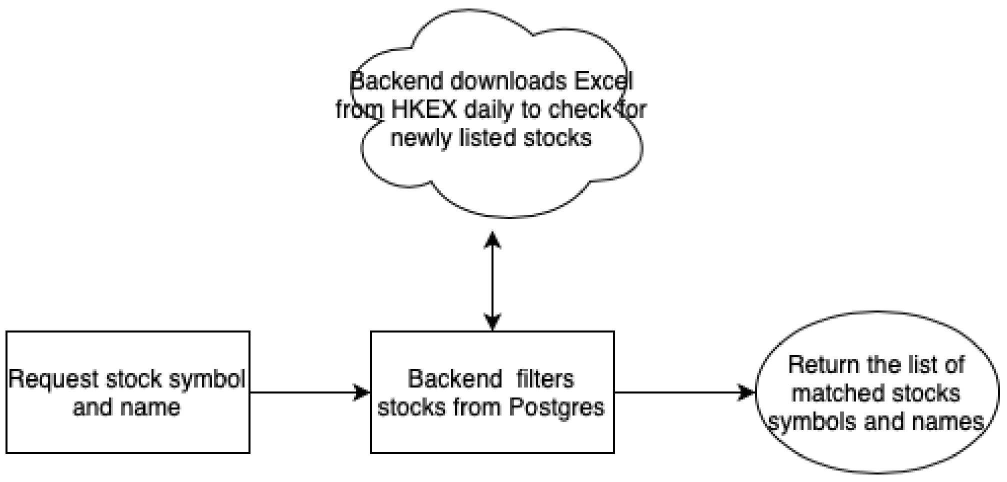
As mentioned in section 2.2.2, when a class is created in Django, a corresponding table is created automatically in the database, with each class property matching a column in the table.

Web APIs

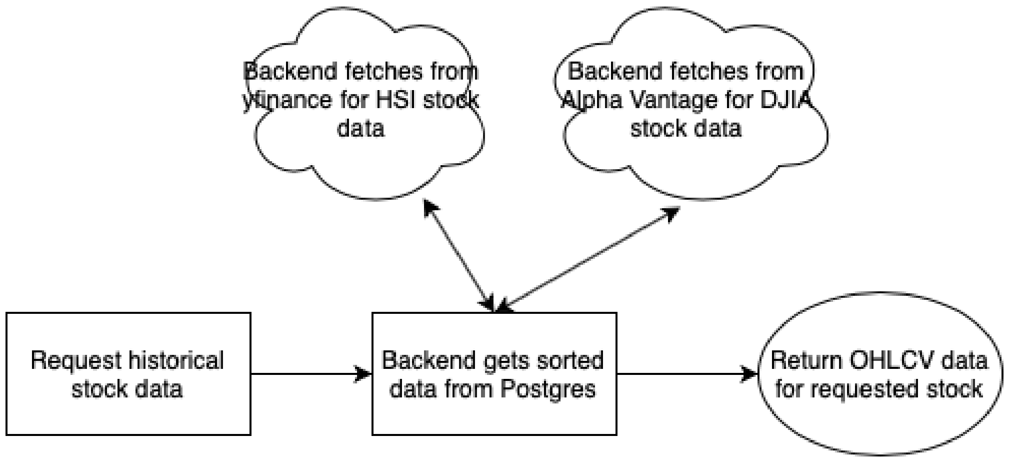
In order to build the APIs in the backend to allow the database to be accessed from other parts of our project like the frontend, we used a library called “Django REST framework”. In order to create the APIs, for each of the class we defined, we created a serializer which can convert complex data into native Python datatypes. Then, we used the generic views provided by the library to build the API views (get, post, put, delete, and patch). Finally, we created the URL patterns for each view so that the APIs can be accessed from the frontend using the URLs.

The diagrams below explain the flow of data for each API:

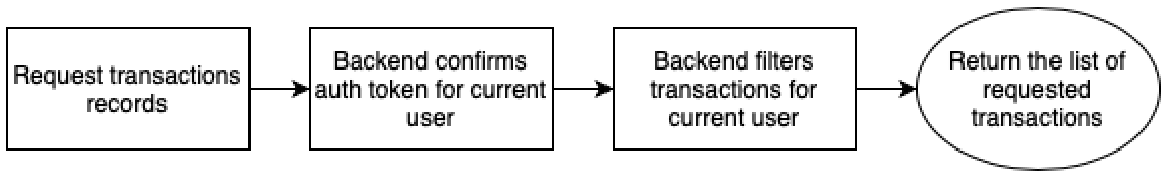
1. Fetch stock symbols and names


Backend downloads Excel from HKEX daily to check for newly listed stocks
Request stock symbol and name
Backend filters stocks from Postgres
Return the list of matched stocks symbols and names

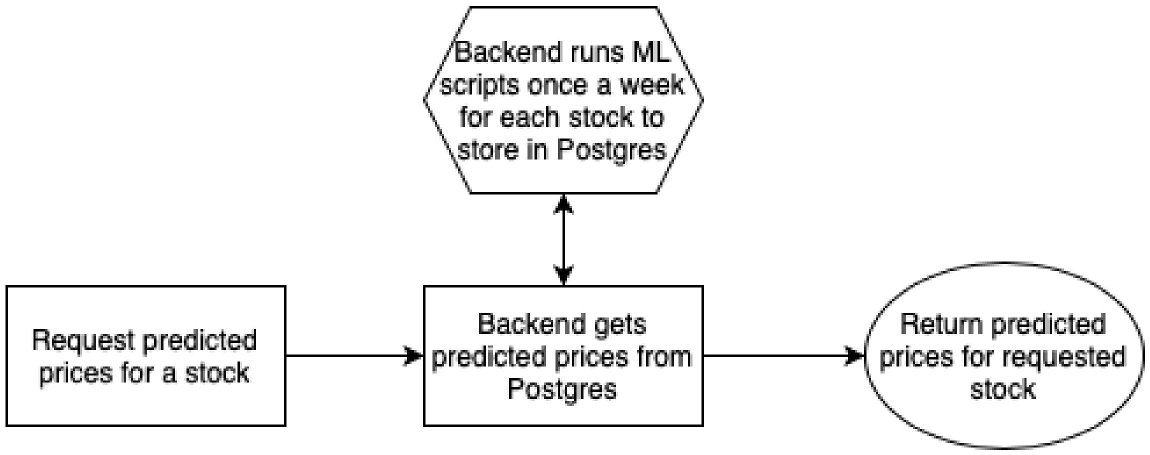
2. Fetch historical stock data


Backend fetches from yfinance for HSI stock data
Backend fetches from Alpha Vantage for DJIA stock data
Request historical stock data
Backend gets sorted data from Postgres
Return OHLCV data for requested stock

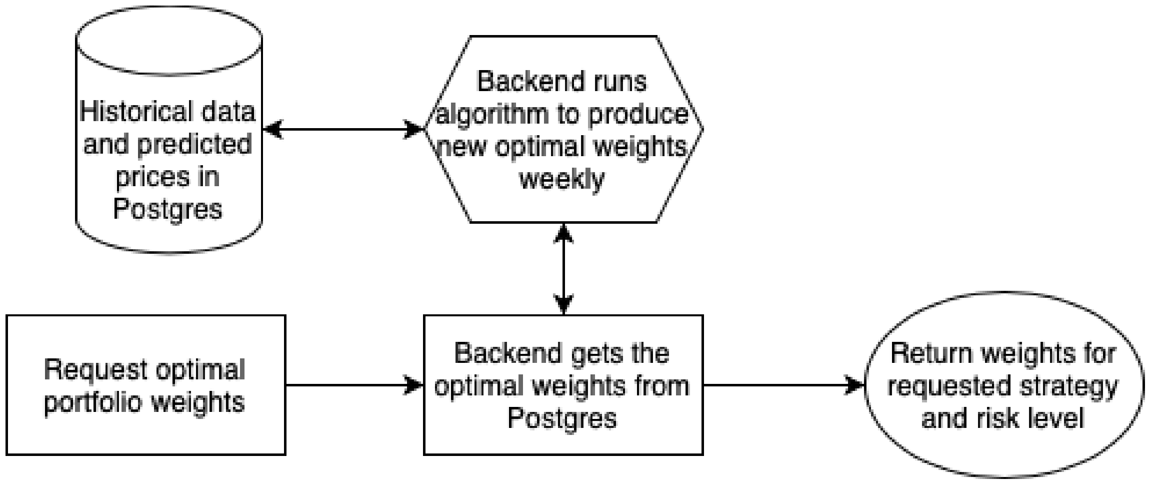
3. Fetch transactions for a user


Request transactions records
Backend confirms auth token for current user
Backend filters transactions for current user
Return the list of requested transactions

4. Fetch predicted prices for a stock


Backend runs ML scripts once a week for each stock to store in Postgres
Request predicted prices for a stock
Backend gets predicted prices from Postgres
Return predicted prices for requested stock

5. Fetch the optimal portfolio weights

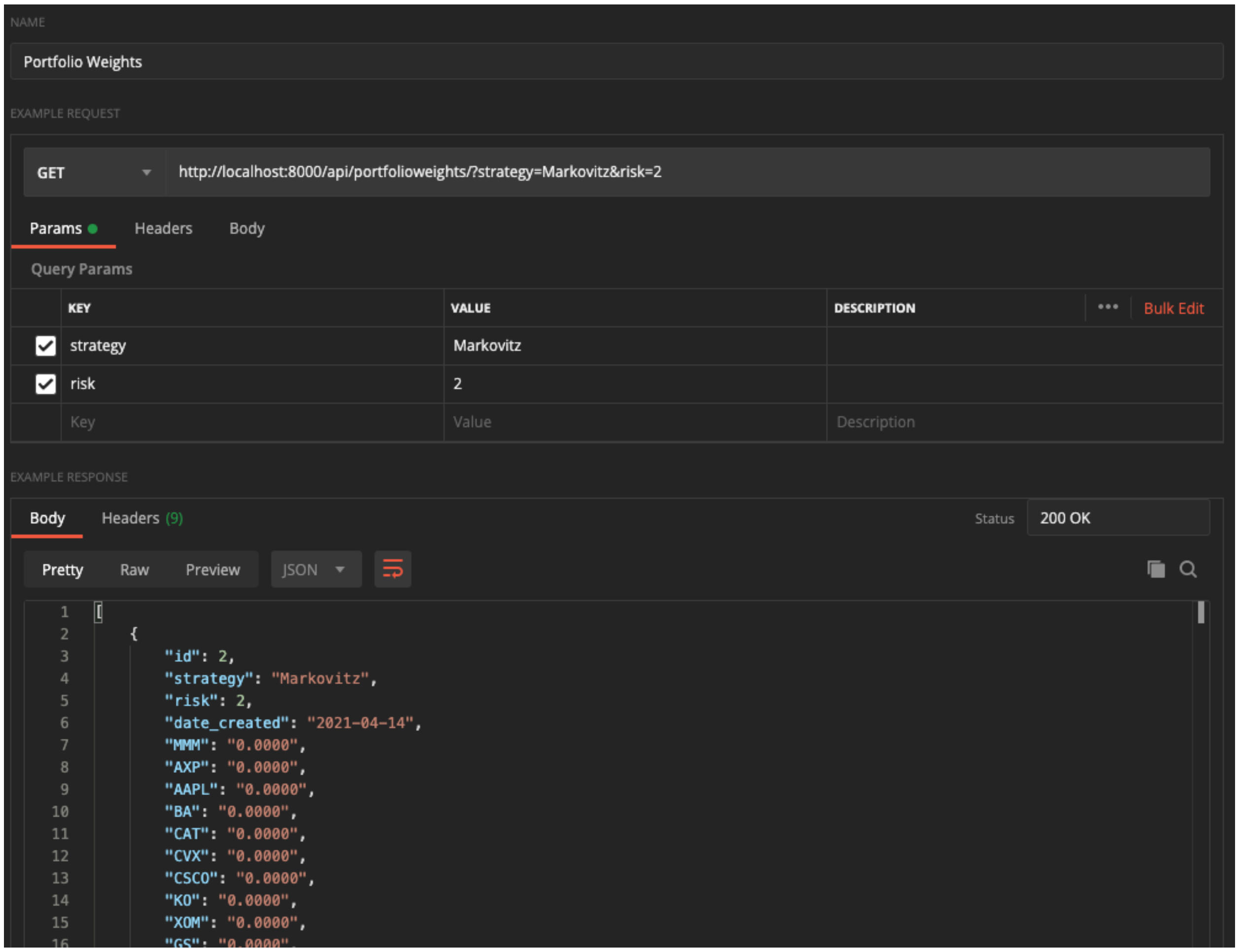

Historical data and predicted prices in Postgres
Backend runs algorithm to produce new optimal weights weekly
Request optimal portfolio weights
Backend gets the optimal weights from Postgres
Return weights for requested strategy and risk level

The API endpoints created are as follow:

URL	**Description**	**Remarks**
/api/stocksinfo?search=${symbol}	get	Gets the list of listed stock

		symbols and names
/api/stockhistory?search=${symbol}&date=${date}	get	Gets the historical data of a stock
/api/stockdailyadjustedprice?search=${symbol}	get	Gets the historical adjusted price for a stock
/api/transactions/	get, post	For a list of transactions for a particular user
/api/transactions/<int:pk>/	get, put, patch, delete	For one particular transaction

/api/stocksprediction?search=${search}	get	Gets the predicted prices for next 30 days for a stock
/api/rmse?search=${search}	get	Gets the RMSE for the prediction for a stock
/api/portfolioweights/?strategy=${strategy}&risk=${risk}	get	Gets the weightings of a particular optimization strategy and risk level


NAME

Portfolio Weights

EXAMPLE REQUEST

GET http://localhost:8000/api/portfolioweights/?strategy=Markovitz&risk=2

	KEY	VALUE	DESCRIPTION
✓	strategy	Markovitz	
✓	risk	2	
	Key	Value	Description

EXAMPLE RESPONSE

Status 200 OK

```
[
    {
        "id": 2,
        "strategy": "Markovitz",
        "risk": 2,
        "date_created": "2021-04-14",
        "MMM": "0.0000",
        "AXP": "0.0000",
        "AAPL": "0.0000",
        "BA": "0.0000",
        "CAT": "0.0000",
        "CVX": "0.0000",
        "CSCO": "0.0000",
        "KO": "0.0000",
        "XOM": "0.0000",
        "GS": "0.0000",
```

Figure 14. Shows sample response when calling the portfolioweights API

User Registration and Authentication

We used another library, "dj-rest-auth," to assist with building the REST API endpoints for user registration and authentication. This library provides the required API views for user registration, login, logout, and password changing. For authentication, we used the "TokenAuthentication" provided by "Django REST framework", which is needed for authenticating users so that they can only access their own data in the database.

The relevant APIs created are as follow:

URL	Description	Remarks
/api/auth/register/	For user registration	
/api/auth/login/	For user login	
/api/auth/logout/	For user logout	
/api/auth/change_password/	For users to change passwords	
/api/auth/user/	For getting information about a user, e.g., their risk level	

Scheduling Jobs

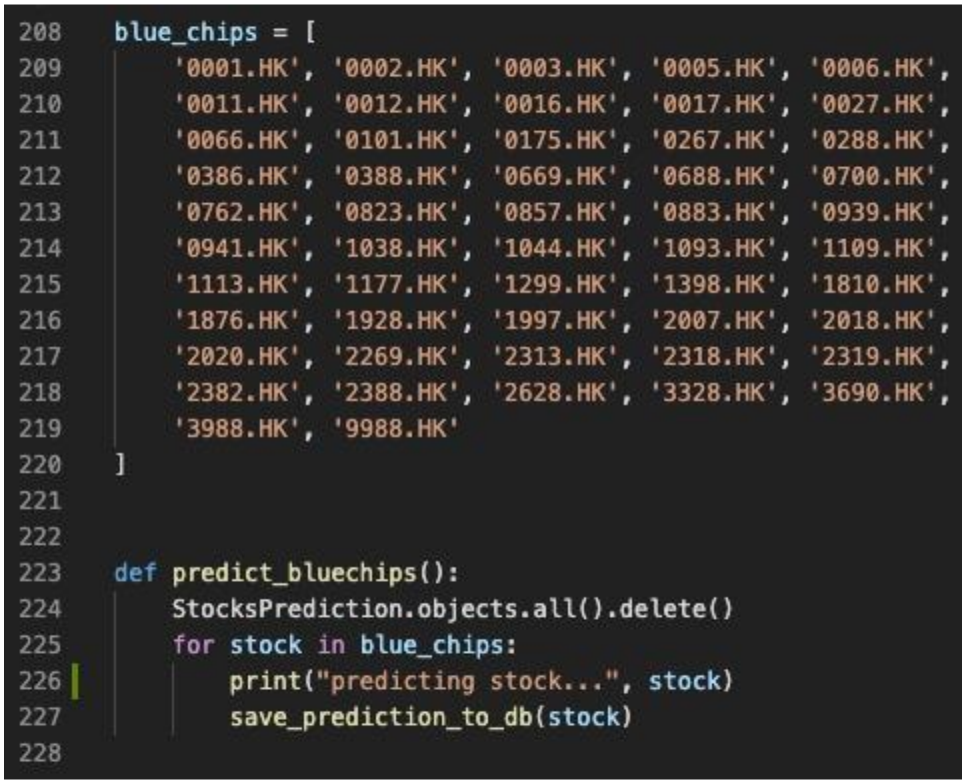
We used a library called “django_crontabs” to schedule tasks that need to be run periodically. Those tasks are described below:

- fetch_stocks_list()
  This function downloads the Excel file from HKEX for the list of all stocks listed, then compares whether a stock is already in our database and if not, it inserts the new stock into the database. It is scheduled to run daily.
- fetch_historical_quotes()

This function runs daily to fetch the latest OHLC daily data from Yahoo Finance, then inserts the new data into the database.

- predict_bluechips()

  This function fetches data from our database to train the machine learning model for each stock, then returns and stores the predicted prices for the next 30 days into the database. It runs weekly.



```python
blue_chips = [
    '0001.HK', '0002.HK', '0003.HK', '0005.HK', '0006.HK',
    '0011.HK', '0012.HK', '0016.HK', '0017.HK', '0027.HK',
    '0066.HK', '0101.HK', '0175.HK', '0267.HK', '0288.HK',
    '0386.HK', '0388.HK', '0669.HK', '0688.HK', '0700.HK',
    '0762.HK', '0823.HK', '0857.HK', '0883.HK', '0939.HK',
    '0941.HK', '1038.HK', '1044.HK', '1093.HK', '1109.HK',
    '1113.HK', '1177.HK', '1299.HK', '1398.HK', '1810.HK',
    '1876.HK', '1928.HK', '1997.HK', '2007.HK', '2018.HK',
    '2020.HK', '2269.HK', '2313.HK', '2318.HK', '2319.HK',
    '2382.HK', '2388.HK', '2628.HK', '3328.HK', '3690.HK',
    '3988.HK', '9988.HK'
]


def predict_bluechips():
    StocksPrediction.objects.all().delete()
    for stock in blue_chips:
        print("predicting stock...", stock)
        save_prediction_to_db(stock)

```

Figure 15. Shows a code snippet of how the predict_bluechips function is implemented

- calculateSQZMOM()

  This function runs daily to calculate the Squeeze Momentum Indicator [15], the most popular technical indicator on TradingView, which is a social network for traders and investors. With this indicator, simple “buy” or “sell” signals can be derived.

2.2.6 Frontend Implementation

We created the frontend application using “Create React App” [16], which is an officially supported way to create a React application with very little configuration needed. To manage states in the whole application, we also used “Redux,” which is a state container for JavaScript apps. On top of that, we chose to use an extension of the programming language JavaScript – TypeScript – to write the application in order to catch errors in earlier stages of development. For API calls from the frontend to our backend or to external sources, we used a library called “Axios” to make the HTTP requests, which is promise based. Lastly, for the graphical user interface itself, we utilized a React UI library called “Ant Design,” which provides enterprise-level UI components that are easily reusable in our application. All pages and components we created are React functional components, with the use of hooks.

Application Architecture

The entire React application is wrapped around a provider from Redux with the store we created (explained below in the Redux section), so that the entire application has access to the Redux store. Then, for the user interface, we wrapped the entire application with a custom component “AppLayout” so that all pages in the application have the same layout design. Finally, we used a library called “React Router” to enable the navigation between different components (or pages) via the URL; we created a component named “Urls” for this purpose, which consists of the following pages and most importantly, a “PrivateRoute” component that checks whether the user is authenticated before allowing access to certain pages:

- Page 1: Home
- Page 2: Login
- Page 3: Register
- Page 4: ChangePassword (wrapped around by PrivateRoute)
- Page 5: Portfolio (wrapped around by PrivateRoute)

Redux

Redux helps manage the state for the entire application so as to make the state more predictable. We defined two different kinds of states: AuthState and TxnState to store the states related to user authentication and transactions respectively. Each of them has its own reducer (authReducer and txnReducer), and they are combined to create a single store for the entire React application. For each kind of state, several actions are created. When the states need to be altered, the corresponding actions are dispatched to the reducer they belong to, which looks at the actions dispatched and the current state to return an updated state to the store. Listed below are the actions for each kind of state:

AuthState

- authRegister
- authLogin
- authCheckState
- authChangePassword
- authGetProfileRisk
- authSetProfileRisk

- authLogout

TxnState

- txnGetTransactions
- txnAddTransaction
- txnEditTransaction
- txnDeleteTransaction

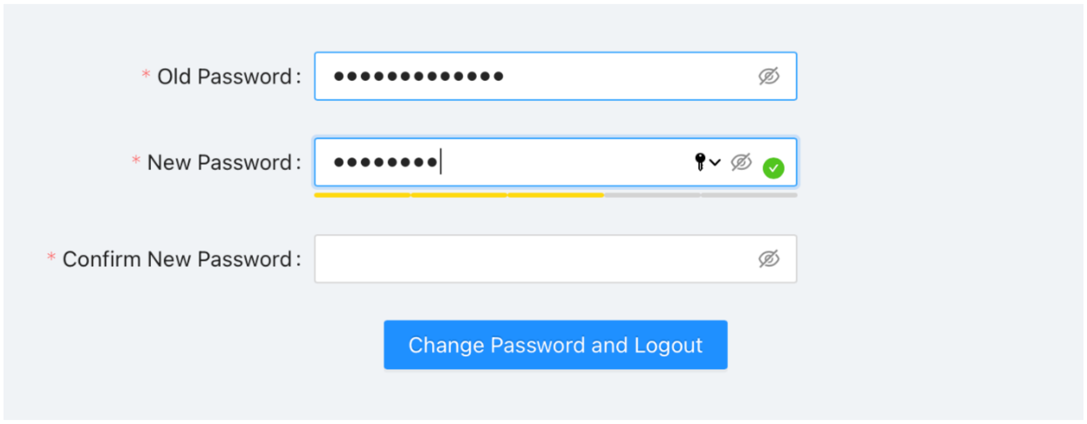
Components

Each component we created may be reused in more than one pages in our application. The list of components can be found in appendix B.

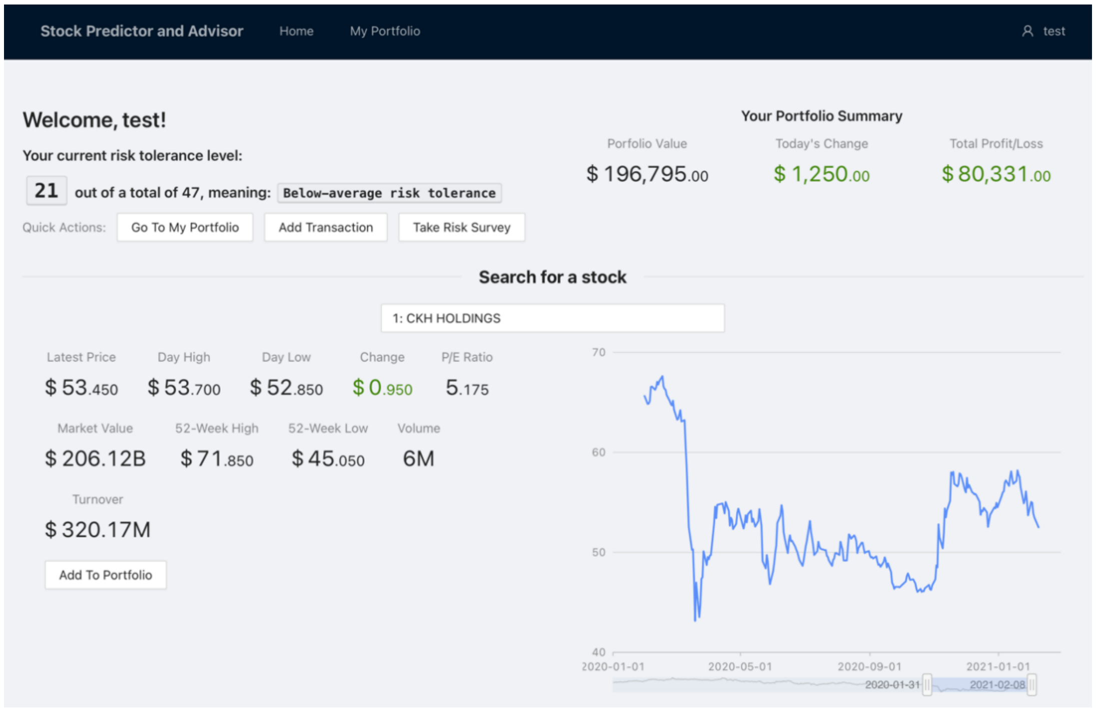
Pages

As mentioned above, there are 5 pages in our application that are managed by React Router. The pages and the components they each contain are listed below:

- ChangePassword



- Home


Stock Predictor and Advisor Home My Portfolio test

Welcome, test!

Your current risk tolerance level:

21 out of a total of 47, meaning: `Below-average risk tolerance`

Quick Actions: Go To My Portfolio | Add Transaction | Take Risk Survey

Your Portfolio Summary

Porfolio Value	Today's Change	Total Profit/Loss
$ 196,795.00	$ 1,250.00	$ 80,331.00

Search for a stock

1: CKH HOLDINGS

Latest Price	Day High	Day Low	Change	P/E Ratio
$ 53.450	$ 53.700	$ 52.850	$ 0.950	5.175

Market Value	52-Week High	52-Week Low	Volume
$ 206.12B	$ 71.850	$ 45.050	6M

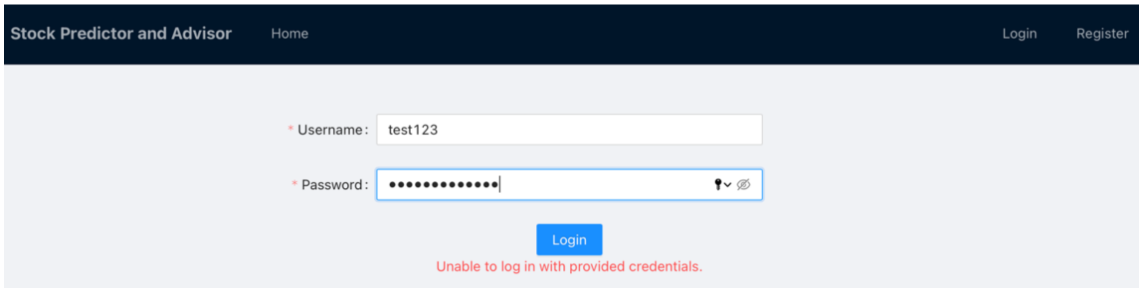
Turnover

$ 320.17M

Add To Portfolio

  - AddTransaction
  - LatestGraph
  - LatestPrice
  - PortfolioNumbers
  - Prediction
  - PredictionGraph
  - Returns
  - RiskSelecter
  - SearchStocks

- Login

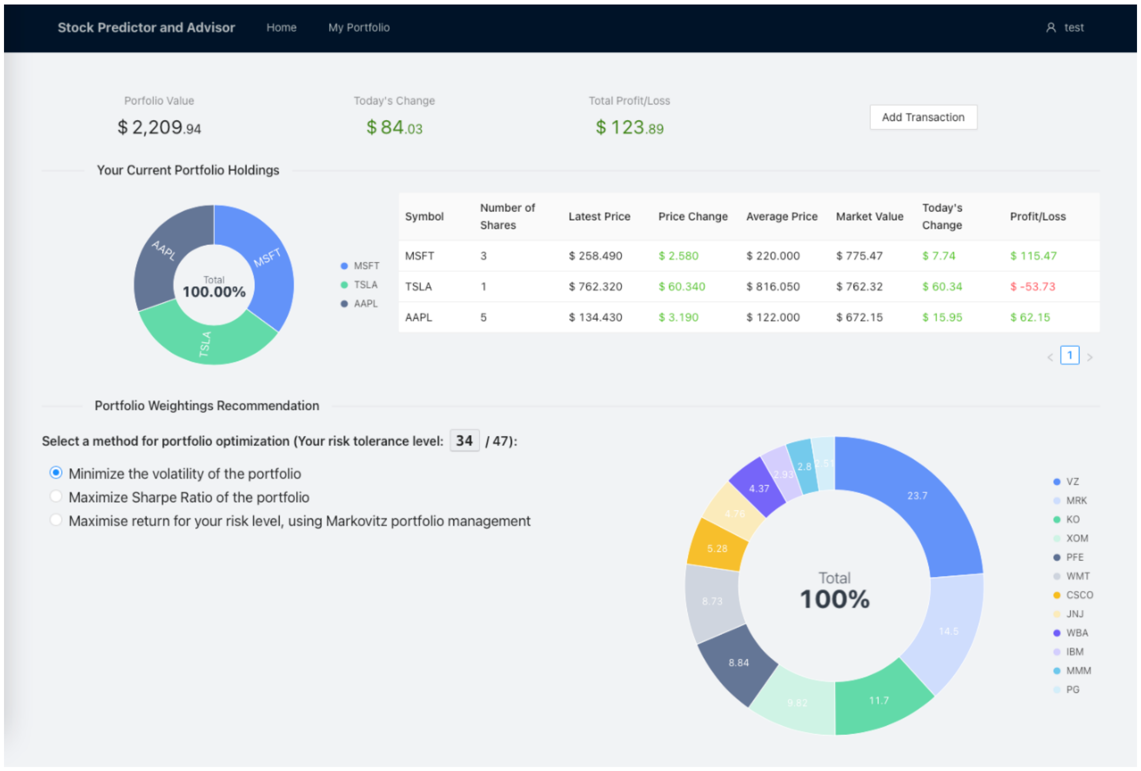


* Username: test123

* Password: •••••••••••••

Unable to log in with provided credentials.

- Portfolio



Porfolio Value	Today's Change	Total Profit/Loss
$ 2,209.94	$ 84.03	$ 123.89

Your Current Portfolio Holdings

Symbol	Number of Shares	Latest Price	Price Change	Average Price	Market Value	Today's Change	Profit/Loss
MSFT	3	$ 258.490	$ 2.580	$ 220.000	$ 775.47	$ 7.74	$ 115.47
TSLA	1	$ 762.320	$ 60.340	$ 816.050	$ 762.32	$ 60.34	$ -53.73
AAPL	5	$ 134.430	$ 3.190	$ 122.000	$ 672.15	$ 15.95	$ 62.15

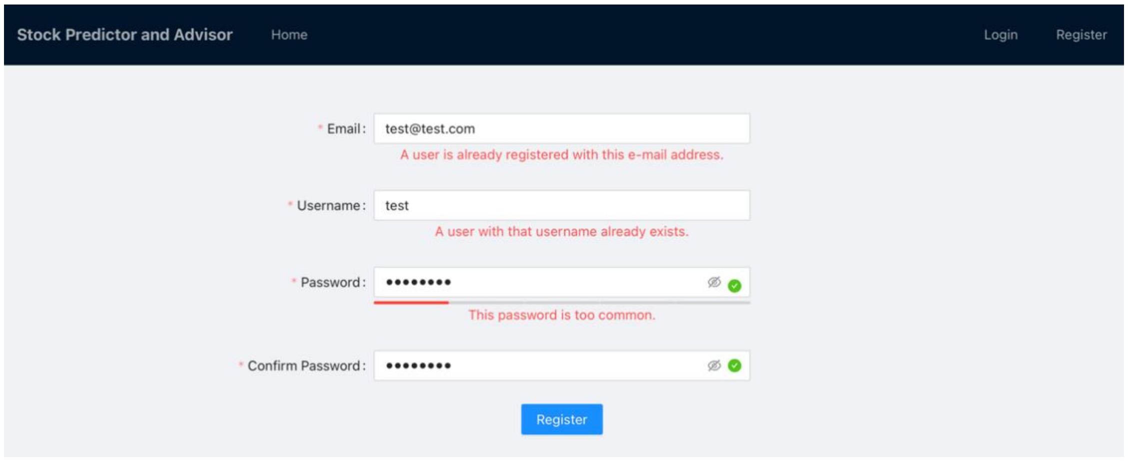
Portfolio Weightings Recommendation

Select a method for portfolio optimization (Your risk tolerance level: 34 / 47):

[✓ Minimize the volatility of the portfolio]
Maximize Sharpe Ratio of the portfolio
Maximise return for your risk level, using Markovitz portfolio management

  - AddTransaction
  - PortfolioHoldings
  - PortfolioNumbers
  - PortfolioWeights
  - TransactionTable

- Register



* Email: test@test.com
A user is already registered with this e-mail address.

* Username: test
A user with that username already exists.

* Password: ••••••••
This password is too common.

* Confirm Password: ••••••••

2.3 Testing

Given that we are using GitHub to synchronize our project and modules, we had to pay particular care to ensure that our project was running effectively. This meant that testing regularly and in depth was crucial. There are two main types of testing that have been done until now.

The most common form of testing is the testing of the ML model. With every change made to the data or the hyperparameters of the model, the end goal is to ensure that the prediction accuracy and precision is as high as possible. As such, frequent testing using metrics such as sum of mean squared errors (MSE) or a confusion matrix. This is because we are employing Test-Driven Development for the model.

The second type of testing is unit testing. For every script or app or function that is created, it is important to test many possible scenarios and use cases that may be encountered. Most of the unit testing was done using Jest. We also performed integration testing with the different modules. This tested whether the different modules are able to communicate with each other smoothly and that an error in one module does not cause the other modules to crash or malfunction. It was through integration testing that it was discovered that the portfolio management algorithm needs to be changed. Initially, it was meant to be called dynamically by the frontend whenever the user made a change to their portfolio. However, it was discovered that the program was extremely computationally heavy due to having to retrieve and process all the stock data. This significantly impacted performance on mobile devices. As such, it was decided that the portfolio prediction would instead be a script that runs once a week and the results would be stored in a database. This is because it is unlikely the user will change his portfolio every day.

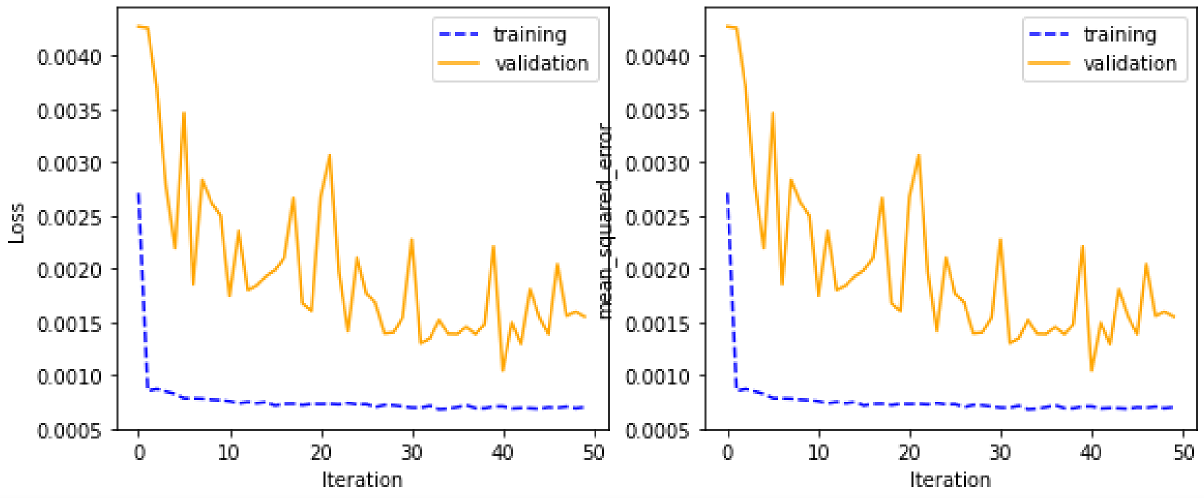
2.4 Evaluation

Given that our FYP is a product meant for use by laymen. The only real way to evaluate the effectiveness of the system is to give it to users and collect feedback. However, given the current state of the system. It would be difficult for someone unfamiliar with the project to navigate and use. Once we have the frontend finalized, we will send it to potential users. The criteria that the project will be evaluated on are:

1. The reliability and speed of the system and the general presentation of the interface
2. The accuracy and precision of the predictions made
3. The usability of the application

The evaluation of the multivariate CNN model and multivariate LSTM model will be discussed in terms of; training and validation mean squared error, train and testing predictions accuracy as well as forward prediction accuracy. The CNN and LSTM models were trained three main attributes; adjusted close, simple moving average (50 day) and relative strength index. The simple moving average is a good factor for stock price prediction as it is a measure of momentum in the price trends of the stock. Similarly, the relative strength index is a good factor for stock price prediction as it can gauge whether a stock is undervalued and should be bought or overvalued and should be sold.

Multivariate CNN model:



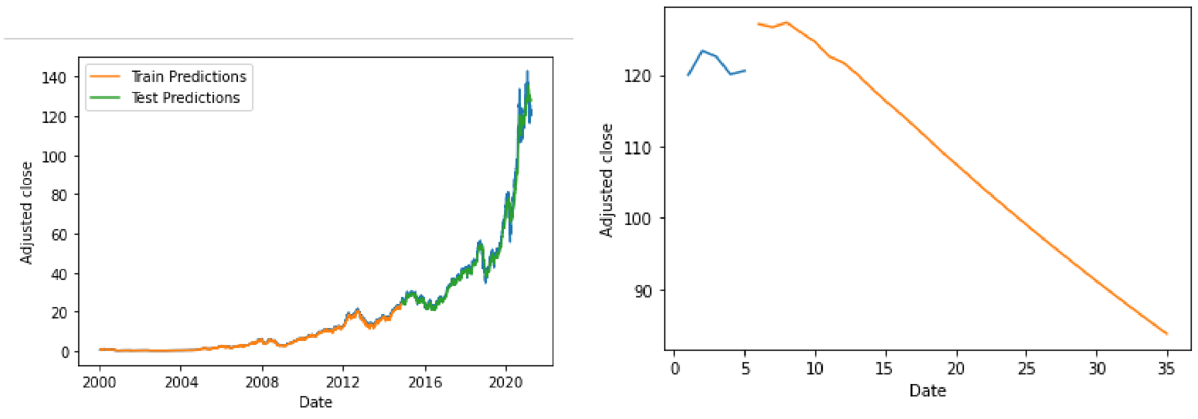


Figure 16. Shows the evaluation metrics for the Multivariate CNN model

Multivariate LSTM model:

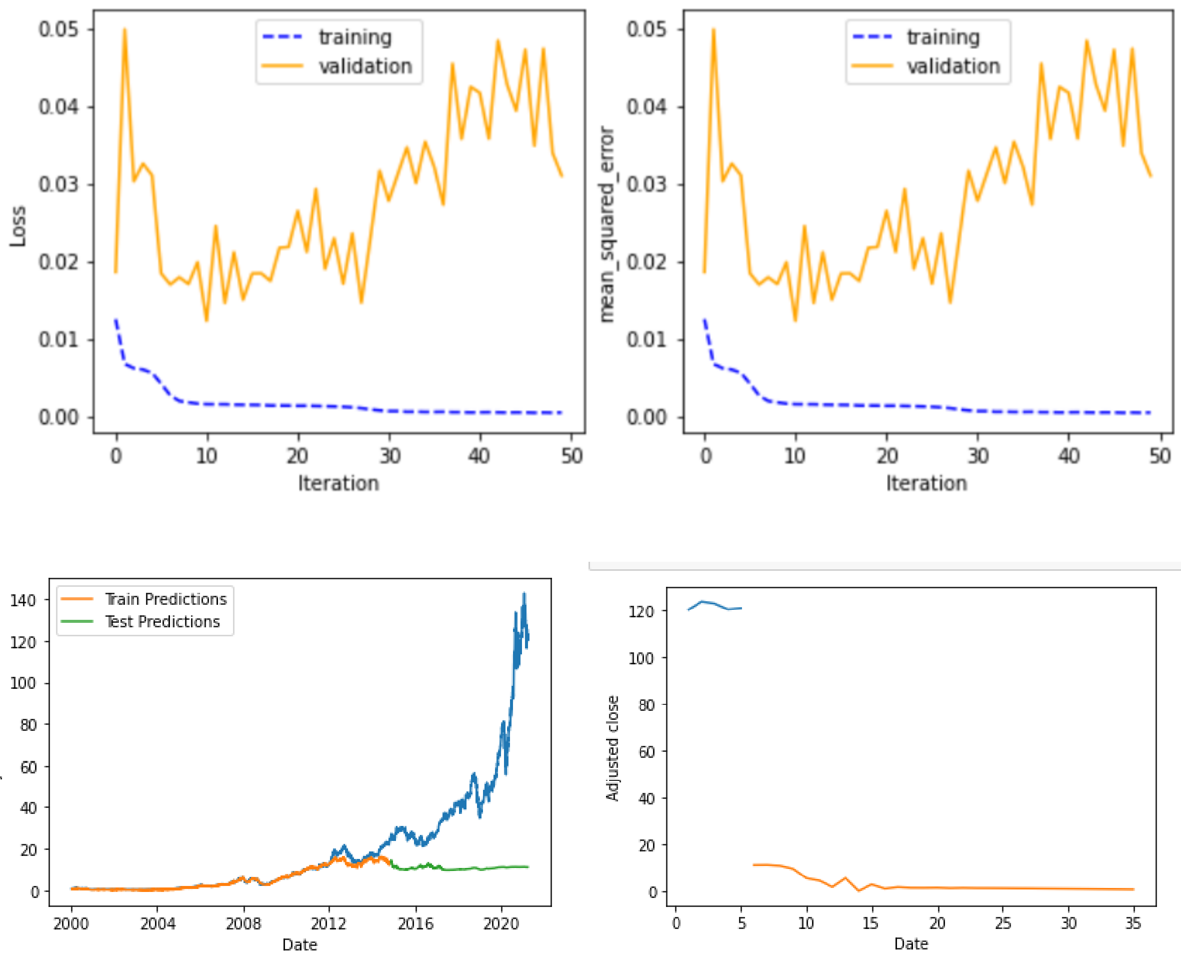


Figure 17. Shows the evaluation metrics for the Multivariate LSTM model

As we can see in the graphs, the mean squared loss of the multivariate CNN model is much lower than that of the multivariate LSTM model. Also, the train prediction and test prediction of the multivariate CNN model are much closer to the actual adjusted close price. The future prediction of the multivariate CNN model for 30 days after the data is also smoother. So, we think that the multivariate CNN model performs better and will use it for the prediction of the stock price.

Evaluating the User Experience

To evaluate on the user experience, we created a survey and asked users how their experiences in using our prototype application were. We asked them to give a rating from 1 to 5 for different components, and the following are the results for the average ratings from 27 respondents:

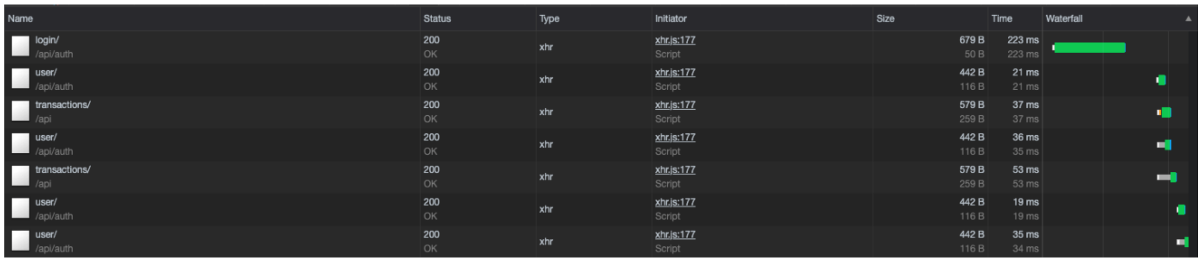
Component	Average Rating
User Registration	4.20
Stock Info Searching and Display	4.17
Stock Prediction Display	4.03
Risk Assessment Survey	4.56
Adding a Transaction to Portfolio	4.10
Portfolio Summary	4.64
Optimal Portfolio Display	4.50

Overall, the web application we developed was satisfactory in terms of user experience. There is, however, room for improvement especially for the display of prediction results.

This may be because currently there is not much explanation on how we arrived at the results and the lack of descriptions on the line chart. The relatively higher ratings for portfolio summary and display of optimal portfolio may be due to the visualization using ring charts that are more intuitive.

Evaluating the loading time

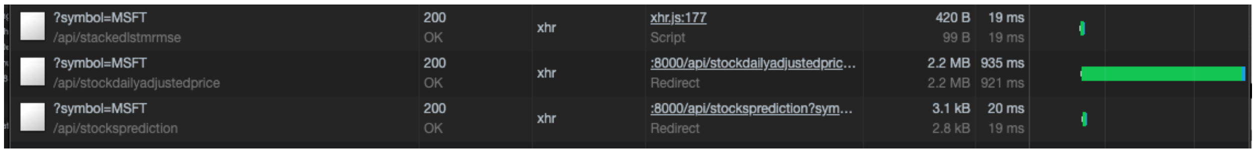
To evaluate the loading time for the web application, we utilized the developer tools in Chrome. The diagram below shows the time for when the user first logins to the webpage and the application loads their information for this user. The time needed was 424ms which was less than half a second, and so the performance was satisfactory.



Name	Status	Type	Initiator	Size	Time	Waterfall
login/ /api/auth	200 OK	xhr	xhr.js:177 Script	679 B 50 B	223 ms 223 ms	
user/ /api/auth	200 OK	xhr	xhr.js:177 Script	442 B 116 B	21 ms 21 ms	
transactions/ /api	200 OK	xhr	xhr.js:177 Script	579 B 259 B	37 ms 37 ms	
user/ /api/auth	200 OK	xhr	xhr.js:177 Script	442 B 116 B	36 ms 35 ms	
transactions/ /api	200 OK	xhr	xhr.js:177 Script	579 B 259 B	53 ms 53 ms	
user/ /api/auth	200 OK	xhr	xhr.js:177 Script	442 B 116 B	19 ms 19 ms	
user/ /api/auth	200 OK	xhr	xhr.js:177 Script	442 B 116 B	35 ms 34 ms	

Figure. 18: Shows time (424ms) needed for when user first logins

The time needed for when a user searches for a stock and for the historical price and predicted price to display was 974ms, nearly a second, which was acceptable.



?symbol=MSFT /api/stackedlstmrmse	200 OK	xhr	xhr.js:177 Script	420 B 99 B	19 ms 19 ms
?symbol=MSFT /api/stockdailyadjustedprice	200 OK	xhr	:8000/api/stockdailyadjustedpric... Redirect	2.2 MB 2.2 MB	935 ms 921 ms
?symbol=MSFT /api/stocksprediction	200 OK	xhr	:8000/api/stocksprediction?sym... Redirect	3.1 kB 2.8 kB	20 ms 19 ms

Figure. 19: Shows time (974ms) needed for searching and displaying results of a stock “MSFT”

Overall, the loading performance was within our acceptable range given the huge amount of data that was needed to display in the frontend.

Evaluating the Portfolio Optimization

The portfolio optimization algorithms can be evaluated by comparing their performance to the rate of return of the typical investor. Since the stocks we are using are from the Dow Jones Industrial Average (DJIA) and the Hang Seng Index (HIS), we can compare the returns of these indexes to our portfolio. To check the performance, we took a period of five years from 2015 Jan – 2020 Jan, then we then generated predictions from the machine learning data for one month. We then compared the one year returns from the optimized portfolio to the returns from the index. The results for the DJIA are shown in the table below. Each column represents the annual returns from each pf the different portfolios:

DJIA	Minimum Volatility	Max Sharpe	Markowitz with risk aversion = 10
10.67%	12.4%	36.2%	21.8%

3 Discussion

Overall, we put in extra effort in developing the whole system infrastructure to ensure the whole end-to-end pipeline works in the first place, before moving towards designing and testing the different machine learning models, as well as the portfolio optimization algorithms.

3.1 Challenges

Our time became limited when we focused more energy in the beginning towards building a working web application including the frontend, backend, database, and data crawler. Although by doing so we successfully built a web application with a clean user interface and functional portfolio management functions, this meant we were left with not much time on developing highly-accurate and sophisticated models for stock price prediction. Furthermore, finding the right data sources and preprocessing the data before feeding into the different models proved to be more time consuming than we had first anticipated.

3.2 Future Work

For the machine learning models, given more time, we would have tested more different types of models and combine the results to form an ensemble model, perhaps include sentimental analysis to better grasp the current trends. Besides testing out more models, we shall also incorporate more features: currently we included some technical indicators in addition to closing prices, but we could have included fundamental indicators of the companies as well as macroeconomic data. This is because these kinds of features may have a longer-term effect on stock prices. Currently, our models are only used to predict the prices for each stock in the DJIA for the next 30 days, which is a relatively short period for investment. Additionally, we could have fine-tuned the hyperparameters even more and experimented with different layers structures for the LSTM and CNN models.

For the React frontend, we could have improved the chart visualization by adding more features such as allowing users to compare different stocks in the same graph, or allowing users to view visualization of different technical indicators next to the graph of closing price. In addition, we should display more data for when a user searches for a stock, like the company’s fundamental data or even related news. Finally, for the page on the user’s own portfolio, besides the ring charts currently displayed that shows the user’s portfolio weightings, there should also be a line chart that shows how the user’s portfolio’s total value varied across different time ranges. This would allow the user to better understand how their current portfolio is performing.

In addition, for the portfolio optimization algorithm, we would like to extend the functionality by adding the ability to add constraints such as a minimum investment amount in a certain industry or stock. The main issue faced was that the stocks were not classified by industry and there was not enough time to program the tagging of the stocks as well as the constraints. In addition, it would be interesting to explore the implementation of other portfolio methods such as Gini mean deviation (GMD) which have been shown to be successful in some cases.

For the Django backend and PostgreSQL database, in the future we wish to improve and optimize the performance of fetching and storing the data. As fetching and storing time series data is time and space consuming, we could have chosen to store the time series data in a separate database that is optimized for this type of data, such as InfluxDB. We could also utilize caching so that users could experience faster data loading.

4 Conclusion

Throughout the project, we learnt how to build a complete machine learning web application from scratch using the latest and most popular technologies, frameworks, and libraries. Although building a full-stack web application using some best software engineering practices was not our initial main focus, it turned out to be a huge part of our project. This gave us the opportunity to learn building an application starting from the user's point of view and experience.

Additionally, by researching the different machine learning models that are currently applied to stock price prediction, we learnt how to design some state-of-the-art prediction models that can be quite sophisticated. We also successfully dealt with processing the huge amount of data so that the data can be input into our multivariate models. Overall, we believe this project can sever as the foundational work for building a fully featured application for new investors.

5 Project Planning

5.1 Distribution of Work

Task	Aadhar	Dragos	Jose	Michael
Do the literature survey	L	A	A	A
Design the data crawler	A	A	L	A
Design the database	A	A	A	L
Design the machine learning models	A	L	A	A
Design the investment algorithm	L	A	A	A
Design the frontend	A	A	A	L
Implement the data crawler	A	A	L	A
Implement the database	A	A	A	L
Implement the machine learning models	A	L	A	A
Implement the investment algorithm	L	A	A	A
Implement the frontend	A	A	A	L
Test the data crawler	A	A	L	A
Test the database	A	A	A	L
Test the machine learning models	A	L	A	A
Test the investment algorithm	L	A	A	A
Test the frontend	A	A	A	L
Perform Integration Testing	A	A	A	L
Write the Proposal	A	A	A	L
Write the Monthly Reports	L	A	A	A
Write the Progress Report	L	A	A	A
Write the Final Report	A	A	A	L
Prepare for the Presentation	L	A	A	A
Design the Project Poster	A	L	A	A

L = Leader, A = Assistant

5.2 GANTT Chart

Task	**July**	**Aug**	**Sep**	**Oct**	**Nov**	**Dec**	**Jan**	**Feb**	**Mar**	**Apr**
Do the literature survey	▨	▨	▨							
Design the data crawler		▨	▨							
Design the database		▨	▨							
Design the machine learning models			▨	▨	■					
Design the investment algorithm			▨	▨						
Design the frontend			▨	▨						
Implement the data crawler					■	■				
Implement the ml model					■	■				
Implement the investment algorithm					■	■				
Implement the frontend				▨	▨	▨	▨	▨		
Test the data crawler					▨	▨	▨	▨		
Test the ml model					▨	▨	▨	▨		
Test the investment algorithm					▨	▨	▨	▨	▨	▨
Test the frontend					▨	▨	▨	▨	▨	▨

Perform integration testing					■	■	■	■	■	■
Write proposal report				■	■	■	■	■	■	■
Write monthly report				■	■	■	■	■	■	■
Write progress report						■	■	■	■	■
Write final report						■	■	■	■	■
Make poster						■	■	■	■	■
Final presentation						■	■	■	■	■

6 Required Hardware & Software

6.1 Hardware

The hardware requirements of our project would be comprised of the following:

1) Web server on cloud to handle the front end and some basic back end functions
2) SQL database on the cloud to store our financial data
3) GPU servers on cloud or on-premise to train the machine learning models
4) Personal development environment of each team members

6.2 Software

The software requirements of the project would be comprised of the following:

1) SQL for the database and SQL Server Management Studio for its development
2) Django and Flask as web development frameworks
3) Python for development and data processing and analytics
4) Anaconda framework for data cleaning, processing and analytics
5) Visual Studio, Visual Studio Code as IDEs for full stack development
6) Jupyter lab and Jupyter notebook for data cleaning, processing and analytics

7 References

[1] S. Selvin, G. E. Vinayakumar R, V. K. Menon and S. K.P, "A System for Predicting Stock Price and Offering Financial Advice".

[2] K. Pawar, R. S. Jalem and V. Tiwari, "Stock market price prediction using LSTM RNN".

[3] S. Bai, J. Z. Kolter and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," 2018.

[4] E. Hoseinzade and S. Haratizadeh, "CNNpred: CNN-based stock market prediction using a diverse set of variables," *Expert Systems With Applications,* pp. 273-285, 2019.

[5] C. StoeanI, W. Paja, R. Stoean and A. Sandita, "Deep architectures for long-term stock price prediction with a heuristic-based strategy for trading simulations," *PLoS ONE,* 2019.

[6] T. Kim and H. Y. Kim, "Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data," *PLoS ONE,* 2019.

[7] M. U. Gudelek, S. A. Boluk and A. M. Ozbayoglu, "A Deep Learning based Stock Trading Model with 2-D CNN Trend Detection," in *2017 IEEE SSCI*, Honolulu, 2017.

[8] J. Eapen, A. Verma and D. Bein, "Novel Deep Learning Model with CNN and Bi-Directional LSTM for Improved Stock Market Index Prediction," in *2019 IEEE 9th Annual CCWC*, Las Vegas, 2019.

[9] N. R., "Introduction to ARIMA: Nonseasonal Models," 2018. [Online]. Available: https://people.duke.edu/~rnau/411arim.htm. [Accessed 18 09 2020].

[10
] L. B. and J. S. Taylor, "Prophet: Forecasting At Scale - Facebook Research," Facebook Research, 23 02 2017. [Online]. Available: https://research.fb.com/prophet-forecasting-at-scale/. [Accessed 18 09 2020].

[11
] A. G. Mata, "A comparison between LSTM and Facebook Prophet models: a financial forecasting case study," Hdl.handle.net, 01 2020. [Online]. Available: http://hdl.handle.net/10609/107367. [Accessed 18 09 2020].

[12
] P. L. W. L. C. H. a. Y. W. W.Fang, "Combine Facebook Prophet and LSTM with BPNN Forecasting financial markets: the Morgan Taiwan Index," pp. 1-2International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), 2019.

[13] HKEX, "List Of Securities," [Online]. Available: https://www.hkex.com.hk/eng/services/trading/securities/securitieslists/ListOfSecurities.xlsx. [Accessed 8 February 2021].

[14] Django, "Databases | Django documentation," [Online]. Available: https://docs.djangoproject.com/en/3.1/ref/databases/. [Accessed 8 February 2021].

[15] LazyBear, "Squeeze Momentum Indicator," [Online]. Available: https://www.tradingview.com/script/nqQ1DT5a-Squeeze-Momentum-Indicator-LazyBear/. [Accessed 9 February 2021].

[16] Facebook, Inc, "Create React App," [Online]. Available: https://create-react-app.dev/docs/getting-started/. [Accessed 9 February 2021].

8 Appendix A: Meeting Minutes

8.1 Minutes of the 1st Project Meeting

Date:	July 19, 2020
Time:	10:00 pm
Place:	Home, via Zoom meeting
Present:	Aadhar, Jose, Michael, Dragos
Absent:	None
Recorder:	Michael

1. Approval of minutes

   This was the first formal group meeting, so there were no minutes to approve.

2. Report on progress

   2.1. All team members have read the instructions of the Final Year Project online and have done research for the topic.

   2.2. All team members have read the information provided by Prof. Lee.

3. Discussion items

   3.1. The goal of the project is to build a system for predicting stock prices and offering financial advice.

   3.2. Instead of just doing the prediction, the system should be able to offer personalized suggestion to users on how they should their investment portfolio based on their different risk aversions. Users should be able to make decision on buy, sell, hold, or diversify.

3.3. The project plan needs to include a list of the main tasks, who will work on each task, and a GANTT chart.

3.4. React should be used in the frontend and perhaps flask will be used for backend.

4. Goals for the coming week

4.1. All team members will continue to research on the topic.

4.2. All team members will read the past FYP reports provided by Prof. Lee.

5. Meeting adjournment and next meeting

The meeting was adjourned at 11:00 pm.

The date and time of the next meeting will be sent through the WhatsApp group.

8.2 Minutes of the 2nd Project Meeting

Date:	Sep 4, 2020
Time:	10:00 pm
Place:	Home, via Zoom meeting
Present:	Aadhar, Jose, Michael, Dragos
Absent:	None
Recorder:	Michael

1. Approval of minutes

   The minutes of the last meeting were approved without amendment.

2. Report on progress

   2.1. All team members have read about some papers about the topic.

   2.2. All team members have read past FYP papers.

3. Discussion items

   3.1. We divided the tasks and each person is responsible for different subsections for the FYP report.

   3.2. We have decided to ask Prof. Lee on our next meeting after we have some preliminary thoughts on the design and implementation.

4. Goals for the coming week

   4.1. Each member should complete their parts for the proposal and send questions to the group as soon as possible; also, we will add to the references section whenever we find something useful.

   4.2. Aadhar will work on the introduction, including the overview, objectives, and the literature survey.

    4.3. Michael will start working on the design and implementation subsections of the methodology part.
    4.4. Dragos will start the testing and evaluation parts.
    4.5. Jose will do the required hardware and software part and also help with Aadhar on the introduction.
    4.6. Michael will email Prof. Lee to schedule our next meeting.
5. Meeting adjournment and next meeting

   The meeting was adjourned at 11:00 pm.

   The date and time of the next meeting will be set later by email.

8.3 Minutes of the 3rd Project Meeting

Date:	Sep 11, 2020
Time:	10:00 am
Place:	Home, via Zoom meeting
Present:	Prof Lee, Aadhar, Jose, Michael, Dragos
Absent:	None
Recorder:	Michael

1. Approval of minutes

   The minutes of the last meeting were approved without amendment.

2. Report on progress

   2.1. All team members have written most of their parts for the proposal report.

   2.2. Aadhar invited the other team members to help with the literature survey as he thought it was an important part of the proposal report.

3. Discussion items

   3.1. Prof Lee said the literature survey should focus on machine learning methods instead of traditional methods on stock analysis; and focus on what has been done in deep learning in the past few years, whether the researches were successful.

   3.2. For financial advice, Prof Lee noted we should think from the user's point of view, as they won't trust the prediction blindly but want advice to help them make good investment.

   3.3. There is no need for a feasibility section, but the literature review should highlight why the problem is difficult.

3.4. Prof Lee reminded us to approach from the bottom up, start building from a smaller scale to ensure everything works before fine tuning the models.

3.5. We should take about 1 to 2 months to think about the algorithm and should relate back to the literature survey.

3.6. Prof Lee suggested that we should minimize programming, so we should probably choose Django instead of Flask for the backend framework.

3.7. For testing and evaluation, Prof Lee said we should ask real users to test it and also run simulations, with the target that the algorithm should at least not lose money.

3.8. Prof Lee said we should have a functional prototype by January and the project should be 80% complete by then; the goal would be that the system is technically reliable, functionally rich, and has good UI and back testing.

4. Goals for the coming week

4.1. We will complete the remaining parts of the proposal report.

4.2. We will each help with a subsection of the literature survey.

5. Meeting adjournment and next meeting

The meeting was adjourned at 10:45 am.

The date and time of the next meeting will be set later by email.

8.4 Minutes of the 4th Project Meeting

Date:	Oct 2, 2020
Time:	10:00 pm
Place:	Home, via Zoom meeting
Present:	Aadhar, Jose, Michael, Dragos

Absent: None

Recorder: Michael

1. Approval of minutes

   The minutes of the last meeting were approved without amendment.

2. Report on progress

   2.1. The proposal report was submitted on schedule and awaiting feedback.

   2.2. Michael set up a Jira board to track tasks, progress and deadlines.

3. Discussion items

   3.1. We agreed that for the coming month the priority should be the data and backend setup.

   3.2. The target is to have a basic functional prototype by the end of semester.

4. Goals for the coming week

   4.1. Aadhar will research on the data stream.

   4.2. Jose will setup a server and host the project on Azure.

   4.3. Dragos will research on using Django as a backend framework.

   4.4. Michael will setup a repository and start setting up a database to store user information.

5. Meeting adjournment and next meeting

   The meeting was adjourned at 10:45 pm.

   The date and time of the next meeting will be set later by email.

8.5 Minutes of the 5th Project Meeting

Date:	Oct 16, 2020
Time:	10:00 pm
Place:	Home, via Zoom meeting
Present:	Aadhar, Jose, Michael, Dragos
Absent:	None
Recorder:	Michael

1. Approval of minutes

   The minutes of the last meeting were approved without amendment.

2. Report on progress
   - 2.1. Dragos and Jose started setting up some code with Django framework and prototyped with it.
   - 2.2. Dragos created a web application with Django and supported user profile creation (including registration, login, and changing profile details).
   - 2.3. Michael started a frontend application with Create React App and connected it to the Django backend set up by Dragos. He used the Django Rest Framework library to create APIs that support authentication.
3. Discussion items
   - 3.1. Aadhar will research on different data sources for the financial and stock data, the current target is to use the Yahoo Finance API.
   - 3.2. Concluded that if we want to include news sentiment analysis then we can add that input to the model.
4. Goals for the coming week

4.1. Continue with the infrastructure setup.

4.2. Write the monthly report and ask for Prof Lee’s signature.

5. Meeting adjournment and next meeting

   The meeting was adjourned at 10:45 pm.

   The date and time of the next meeting will be set later by email.

8.6 Minutes of the 6th Project Meeting

Date:	Nov 16, 2020
Time:	10:00 pm
Place:	Home, via Zoom meeting
Present:	Jose, Michael, Dragos
Absent:	Aadhar
Recorder:	Michael

1. Approval of minutes

   The minutes of the last meeting were approved without amendment.

2. Report on progress

   2.1. Michael continued with the integration between the Django backend and the React frontend and chose PostgreSQL as the database in place of the default SQLite provided by Django.

   2.2. Michael created a “portfolio” page for the web application where users can add their previous stocks transactions so that they can track their stocks portfolios.

   2.3. Jose prototyped with Django to create an application where a user can search for a stock symbol and then get the daily indicators of that stock.

3. Discussion items

   3.1. Agreed that a data crawler may not be necessary at this stage as we could find an API to get and store the data we want.

   3.2. We should create a prototype application using simple models first to illustrate the whole pipeline, before increasing performance and accuracy.

4. Goals for the coming week

    4.1. Jose will select an appropriate API to fetch financial data and then store into the database regularly.
    4.2. Dragos will start experimenting with different machine learning packages to create a pipeline where data fetched and stored in the database can be pre-processed and fed into a simple model, then the model can be trained, validated, and tested.
    4.3. Michael will continue with the frontend/backend integration.
5. Meeting adjournment and next meeting

   The meeting was adjourned at 10:45 pm.

   The date and time of the next meeting will be set later by email.

8.7 Minutes of the 7th Project Meeting

Date:	Dec 23, 2020
Time:	10:00 am
Place:	Home, via Zoom meeting
Present:	Jose, Michael, Dragos
Absent:	Aadhar
Recorder:	Dragos

1. Approval of minutes

   The minutes of the last meeting were approved without amendment.

2. Report on progress

   2.1. Michael fixed a bug where users could sell stocks that he doesn't own

   2.2. Michael further developed the frontend; stocks data can now be shown in the frontend.

3. Discussion items

   3.1. Agreed that the data fetched from current API is not comprehensive enough

   3.2. We should next find a way to display prediction results at the frontend

      3.2.1. Prediction results from database?

      3.2.2. Running ML models at the backend?

4. Goals for the coming week

   4.1. Jose will find an appropriate API that can fetch comprehensive stock data.

   4.2. Michael will continue to work on the frontend, trying to show last 5 months close price of the stocks.

   4.3. Dragos will create ml model with stock data in current database.

5. Meeting adjournment and next meeting

   The meeting was adjourned at 10:45 am.

   The date and time of the next meeting will be set later by email.

8.8 Minutes of the 8th Project Meeting

Date:	Jan 3, 2021
Time:	10:00 am
Place:	Home, via Zoom meeting
Present:	Aadhar, Jose, Michael, Dragos
Absent:	None
Recorder:	Michael

1. Approval of minutes

   The minutes of the last meeting were approved without amendment.

2. Report on progress

   2.1. Aadhar emailed the communication tutor for the feedback on the proposal report but has yet to receive a reply.

   2.2. Jose analyzed several APIs and found that Alpha Vantage is the most viable one even though it has a limit on the number of calls per minute.

   2.3. Michael implemented the frontend to show graphs and also a user's portfolio holdings.

   2.4. Dragos built a simple model by fetching data from the database.

3. Discussion items

   3.1. Aadhar suggested that we can use the alphas and betas to calculate the risk of a stock and also a user's portfolio.

   3.2. Michael suggested to use the day difference in stock prices as the input to the machine learning models.

3.3. Jose discussed the advantages and disadvantages of Alpha Vantage and we concluded that we should use that API.

3.4. We agreed to use a risk assessment survey to determine the risk aversion level of a user.

4. Goals for the coming week

4.1. Aadhar will try to set up a meeting with Prof. Lee on the coming Friday.

4.2. Aadhar will calculate the risks of a stock and of a portfolio in the frontend.

4.3. Jose will use Alpha Vantage to store some data into the database.

4.4. Dragos will continue with the model and try using the day difference in stock prices as input.

4.5. Michael will try displaying prediction results on the frontend.

4.6. Michael will implement the survey in the frontend.

5. Meeting adjournment and next meeting

The meeting was adjourned at 10:45 am.

The date and time of the next meeting will be set later by email.

8.9 Minutes of the 9th Project Meeting

Date:	Jan 10, 2021
Time:	9:00 pm
Place:	Home, via Zoom meeting
Present:	Prof Lee, Aadhar, Jose, Michael, Dragos
Absent:	None
Recorder:	Michael

1. Approval of minutes

   The minutes of the last meeting were approved without amendment.

2. Report on progress

   2.1. Jose implemented the functions using the API from Alpha Vantage that can store stock data into the database.

   2.2. Michael found a risk tolerance survey online and implemented it into the frontend.

   2.3. Michael used Drago’s model to display some prediction results on the frontend, including the predicted price for the next 30 days.

3. Discussion items

   3.1. Prof Lee suggested that we can have separate models for different sectors, as stocks in the same sector may have similar trends.

   3.2. We should decide how many stocks and how many years of data to play with.

   3.3. For our prototype, we should focus on completing the whole pipeline to make sure it works, including the risk and portfolio weighting recommendation, and data collection, storage, training, prediction, and visualization.

3.4. Start with small amount of data and then finetune the accuracy at a later stage, then sample different models to see which is more accurate.

3.5. After the prototype is tested and works, add more other features such as sentiment analysis, building more sophisticated models, and advisory for investors.

4. Goals for the coming week

4.1. Provide the link to the prototype for Prof Lee to try it out.

4.2. Give Prof Lee an overall plan in the monthly report and include the features we plan to implement and how we plan to integrate individual components together.

4.3. Try to assign weighting in a portfolio for investment.

4.4. Calculate the risk of a stock and a portfolio to balance return vs risk.

5. Meeting adjournment and next meeting

The meeting was adjourned at 9:45 pm.

The date and time of the next meeting will be set later by email.

8.10 Minutes of the 10th Project Meeting

Date:	Jan 31, 2021
Time:	10:30 am
Place:	Home, via Zoom meeting
Present:	Aadhar, Jose, Michael, Dragos
Absent:	None
Recorder:	Michael

1. Approval of minutes

   The minutes of the last meeting were approved without amendment.

2. Report on progress

   2.1. Jose is working on storing some technical indicators to the database using the Alpha Vantage API.

   2.2. Dragos is trying to train separate models for different groups of stocks.

   2.3. Michael is working on implementing the Squeeze Momentum Indicator in order to decide the buy/hold/sell strategies.

3. Discussion items

   3.1. Jose said he read a paper stock prediction and thought that technical indicators like SMA and RSI would be important.

   3.2. Dragos is concerned how the grouping of the stocks should be done; Michael will try to store the groupings into the database.

4. Goals for the coming week

   4.1. Jose will fetch from the API and store the technical indicators in the database using the existing formatting.

4.2. Aadhar will start implementing the risk calculation and display of the risk and portfolio in the frontend.

4.3. Dragos will continue with training models for different groups of stocks.

4.4. Michael will do the grouping of the stocks and also continue with implementing the aforementioned technical indicator.

5. Meeting adjournment and next meeting

The meeting was adjourned at 11:30 am.

The date and time of the next meeting will be set later by email.

8.11 Minutes of the 11th Project Meeting

Date:	Feb 28, 2021
Time:	11:00 am
Place:	Home, via Zoom meeting
Present:	Aadhar, Jose, Michael, Dragos
Absent:	None
Recorder:	Michael

1. Approval of minutes

   The minutes of the last meeting were approved without amendment.

2. Report on progress

   2.1. Aadhar has been reading about the different investment strategies and how they can be implemented with the prediction results.

   2.2. Dragos has been trying to fix the code for his RNN model.

   2.3. Jose has been reading about papers that use different technical indicators to generate buy/sell signals and to predict trends.

3. Discussion items

   3.1. Aadhar thinks his investment algorithm and advice-giving part will be mainly based on the prediction results from the machine learning models.

   3.2. He also says the algorithm will take the users' risk preference into account; if the user has a lower risk tolerance but the algorithm predicts a high-risk stock to rise, the user can choose to buy at different times to average out the price.

   3.3. Jose thinks we should follow the paper he sent out for the investment strategies/signals generation part.

3.4. He says we should test out several models (RNN, CNN, etc.) and then choose the best model, then finetune the parameters for that model.

4. Goals for the coming week

   4.1. Aadhar will implement something on the investment advice algorithm in the backend.

   4.2. Jose will work with Dragos for the ML models and connection to the database.

   4.3. Dragos will continue fixing the code for the RNN model.

   4.4. Michael will host the database online so everyone can access it.

5. Meeting adjournment and next meeting

   The meeting was adjourned at 12 pm.

   The date and time of the next meeting will be set later by email.

9 **Appendix B**: Other Materials

9.1 Tables for all classes in Django

users_profile

Column Name	**Type**	**Description**	**Remarks**
id	Integer	Primary key	
user_id	Integer	Foreign key in the table auth_user	
risk	Integer	The risk level from 0 to 47; higher means a higher risk tolerance	

authtoken_token

Column Name	**Type**	**Description**	**Remarks**
key	Character	Primary key	

created	Timestamp	The time when the token is created	
user_id	Integer	Foreign key in the table auth_user	

transactions_transaction

Column Name	**Type**	**Description**	**Remarks**
id	Integer	Primary key	
action	Character	‘B’ or ‘S’ for buy or sell respectively	
symbol	Character	Stock symbol	
share	Integer	Number of shares	
price	Decimal	Transaction price	
fee	Decimal	Transaction fee	
date	Date	Transaction date	

note	Text		
profile_id	Integer	Foreign key in the table users_profile	

stocks_stocksinfo

Column Name	Type	Description	Remarks
id	Integer	Primary key	
symbol	Character	Stock symbol	
name	Character	Stock name	
category	Character	Stock category	

stocks_stockhistory

Column Name	Type	Description	Remarks
id	Integer	Primary key	

symbol	Character	Stock symbol	
date	Date	Date of data	
open	Decimal	Open price	
close	Decimal	Close price	
high	Decimal	Daily high	
low	Decimal	Daily low	
volume	Decimal	Daily volume	

stocks_stockdailyadjustedprice

Column Name	Type	Description	Remarks
id	Integer	Primary key	
symbol	Character	Stock symbol	
date	Date	Date of data	

open	Decimal	Open price	
high	Decimal	Daily high	
low	Decimal	Daily low	
close	Decimal	Close price	
adjusted_close	Decimal	Adjusted close price	
volume	Decimal	Daily volume	
dividend	Decimal	Dividend	
split_coefficient	Decimal	Split coefficient	

stocks_stocksimplemovingaverage

Column Name	**Type**	**Description**	**Remarks**
id	Integer	Primary key	
symbol	Character	Stock symbol	

date	Date	Date of data	
SMA	Decimal	Simple Moving Average	

stocks_stockrelativestrengthindex

Column Name	**Type**	**Description**	**Remarks**
id	Integer	Primary key	
symbol	Character	Stock symbol	
date	Date	Date of data	
RSI	Decimal	Relative Strength Index	

stocks_stocksqzmom

Column Name	**Type**	**Description**	**Remarks**
id	Integer	Primary key	

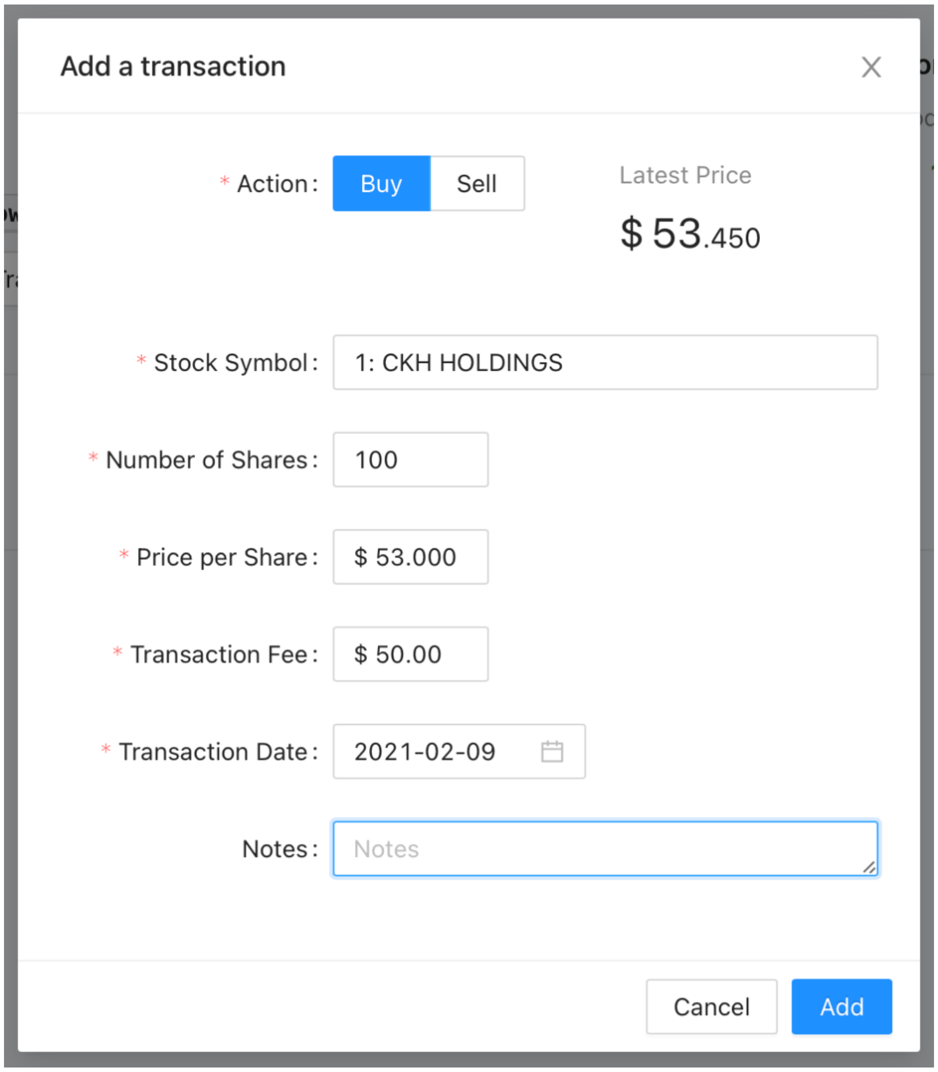
symbol	Character	Stock symbol	
date	Date	Date of data	
value	Decimal	Value calculated from the indicator	
direction	Character	‘pos_up’, ‘pos_down’, ‘neg_up’, or ‘neg_down’	
sqzOn	Boolean	Whether the squeeze is on	
sqzOff	Boolean	Whether the squeeze is off	
noSqz	Boolean	Whether there is no squeeze	

ml_models_stocksprediction

Column Name	**Type**	**Description**	**Remarks**
id	Integer	Primary key	
symbol	Character	Stock symbol	
predicted_price	Decimal	Predicted price	
day	Integer	Number of days from the first day of prediction	

9.2 React Components

- AddTransaction



Add a transaction

* Action: Buy Sell

Latest Price

$ 53.450

* Stock Symbol: 1: CKH HOLDINGS

* Number of Shares: 100

* Price per Share: $ 53.000

* Transaction Fee: $ 50.00

* Transaction Date: 2021-02-09

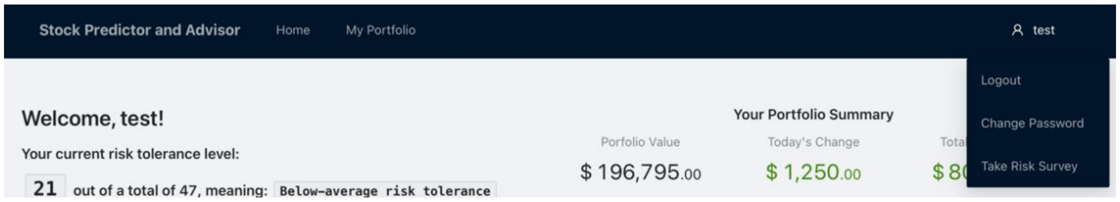
Notes: Notes

Cancel Add

- AppFooter

Copyright © DL3 2021.

- AppHeader



Stock Predictor and Advisor Home My Portfolio

test

Logout

Change Password

Take Risk Survey

Welcome, test!

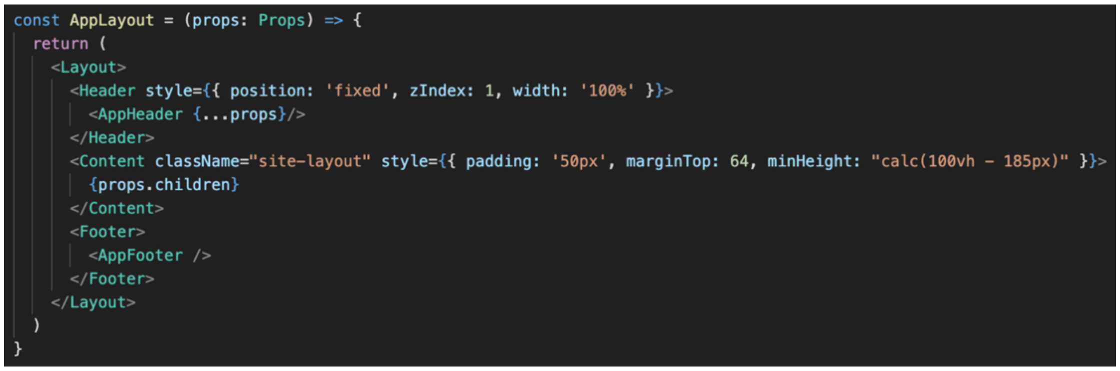
Your current risk tolerance level:

21 out of a total of 47, meaning: `Below-average risk tolerance`

Your Portfolio Summary

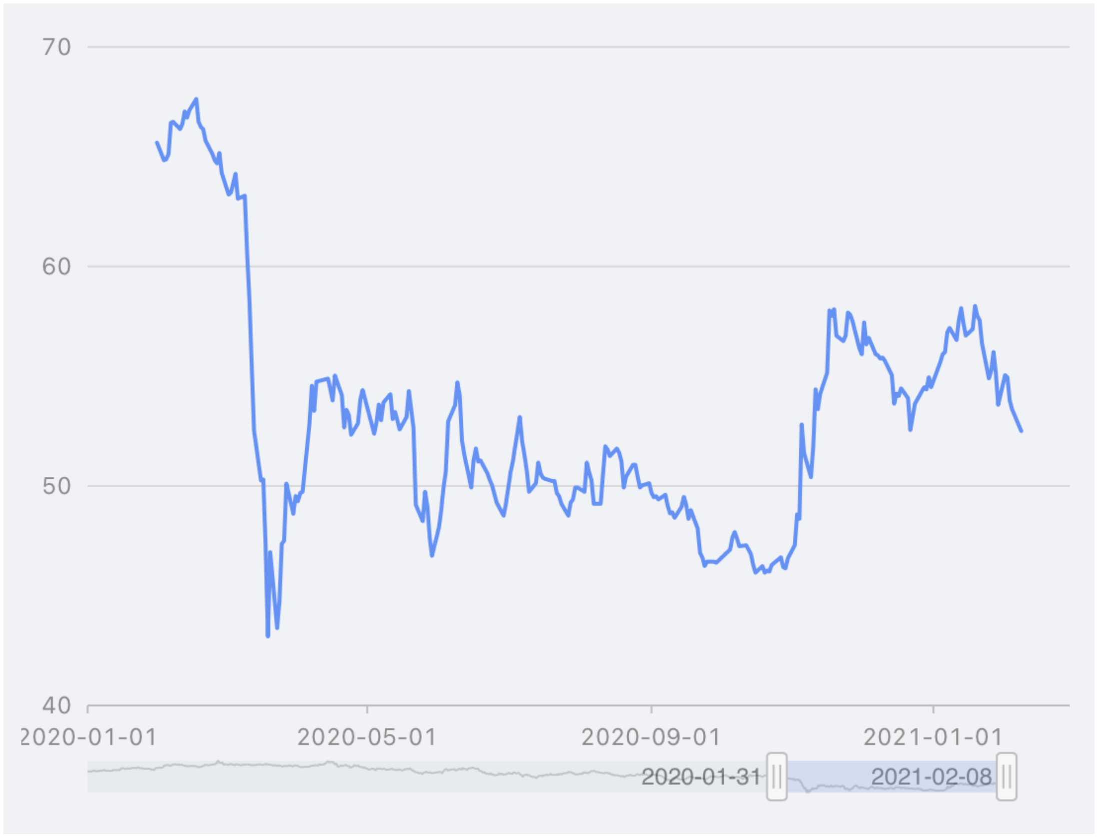
Porfolio Value	Today's Change	Tota
$ 196,795.00	$ 1,250.00	$ 8(

- AppLayout

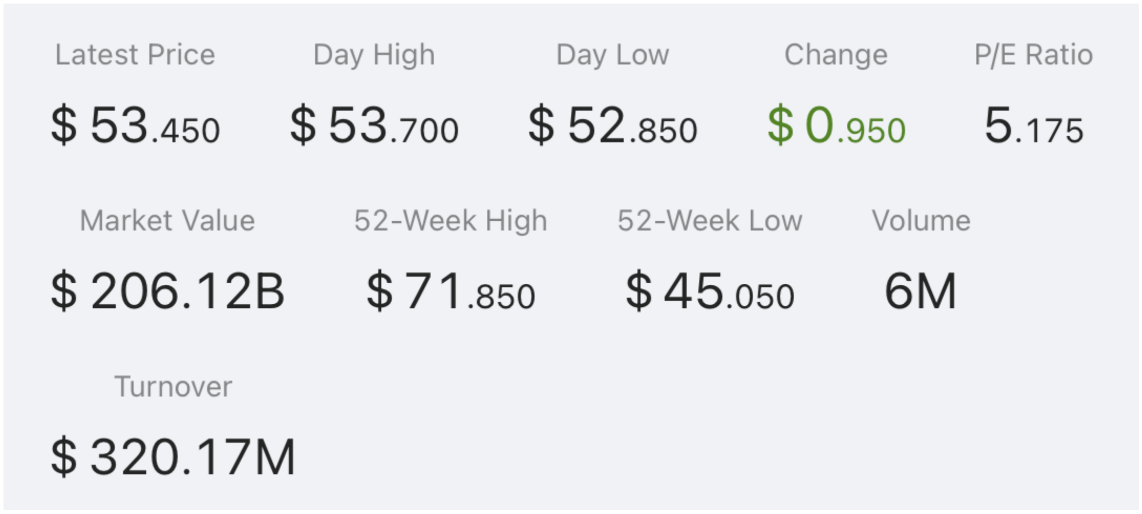


```
const AppLayout = (props: Props) => {
  return (
    <Layout>
      <Header style={{ position: 'fixed', zIndex: 1, width: '100%' }}>
        <AppHeader {...props}/>
      </Header>
      <Content className="site-layout" style={{ padding: '50px', marginTop: 64, minHeight: "calc(100vh - 185px)" }}>
        {props.children}
      </Content>
      <Footer>
        <AppFooter />
      </Footer>
    </Layout>
  )
}
```

- LatestGraph



- LatestPrice

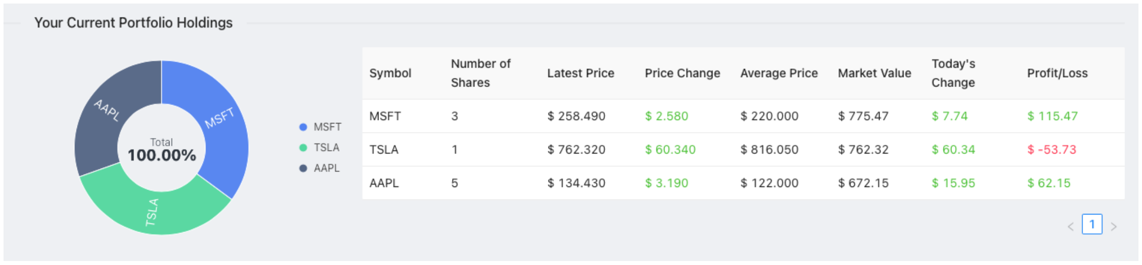


Latest Price	Day High	Day Low	Change	P/E Ratio
$ 53.450	$ 53.700	$ 52.850	$ 0.950	5.175

Market Value	52-Week High	52-Week Low	Volume
$ 206.12B	$ 71.850	$ 45.050	6M

Turnover
$ 320.17M

- PortfolioHoldings

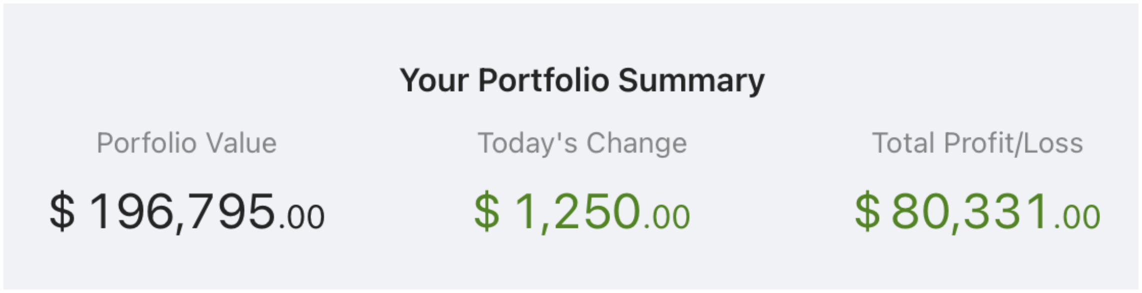


Your Current Portfolio Holdings

Symbol	Number of Shares	Latest Price	Price Change	Average Price	Market Value	Today's Change	Profit/Loss
MSFT	3	$ 258.490	$ 2.580	$ 220.000	$ 775.47	$ 7.74	$ 115.47
TSLA	1	$ 762.320	$ 60.340	$ 816.050	$ 762.32	$ 60.34	$ -53.73
AAPL	5	$ 134.430	$ 3.190	$ 122.000	$ 672.15	$ 15.95	$ 62.15

< 1 >

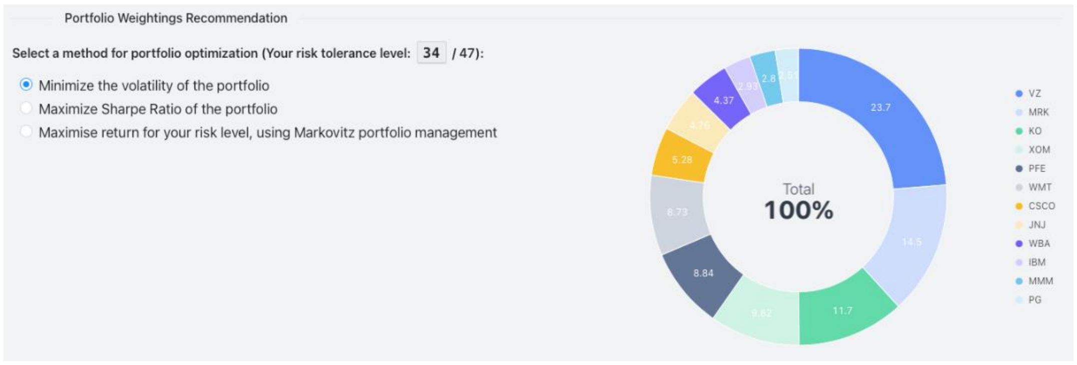
- PortfolioNumbers



Your Portfolio Summary

Porfolio Value	Today's Change	Total Profit/Loss
$ 196,795.00	$ 1,250.00	$ 80,331.00

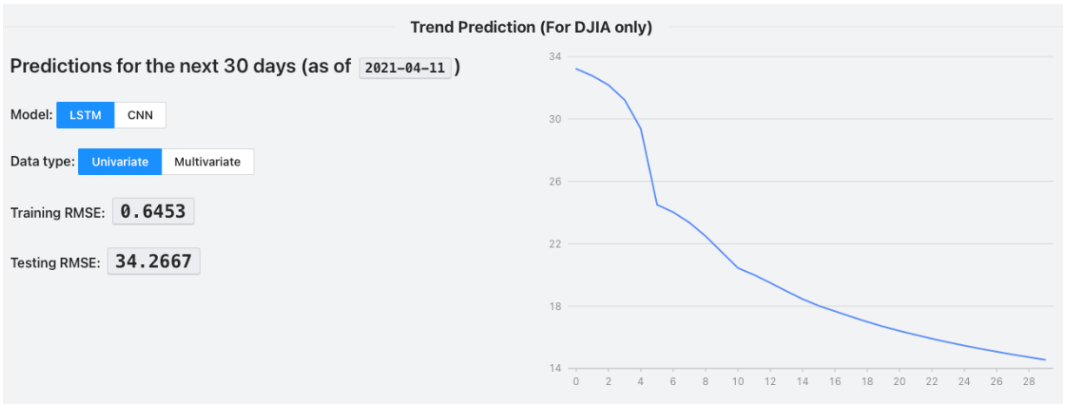
- PortfolioWeights


Portfolio Weightings Recommendation

Select a method for portfolio optimization (Your risk tolerance level: 34 / 47):

- [✓ Minimize the volatility of the portfolio]
- Maximize Sharpe Ratio of the portfolio
- Maximise return for your risk level, using Markovitz portfolio management

- Prediction


Trend Prediction (For DJIA only)

Predictions for the next 30 days (as of 2021-04-11)

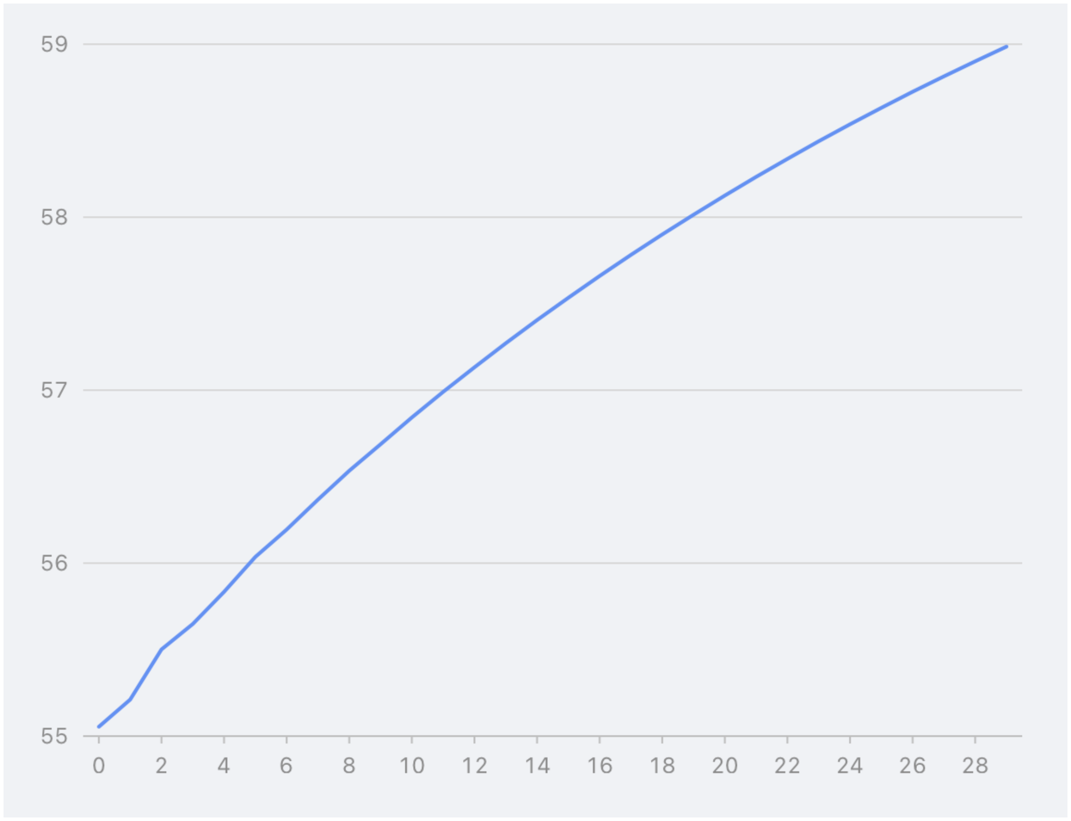
Model: [✓ LSTM] CNN

Data type: [✓ Univariate] Multivariate

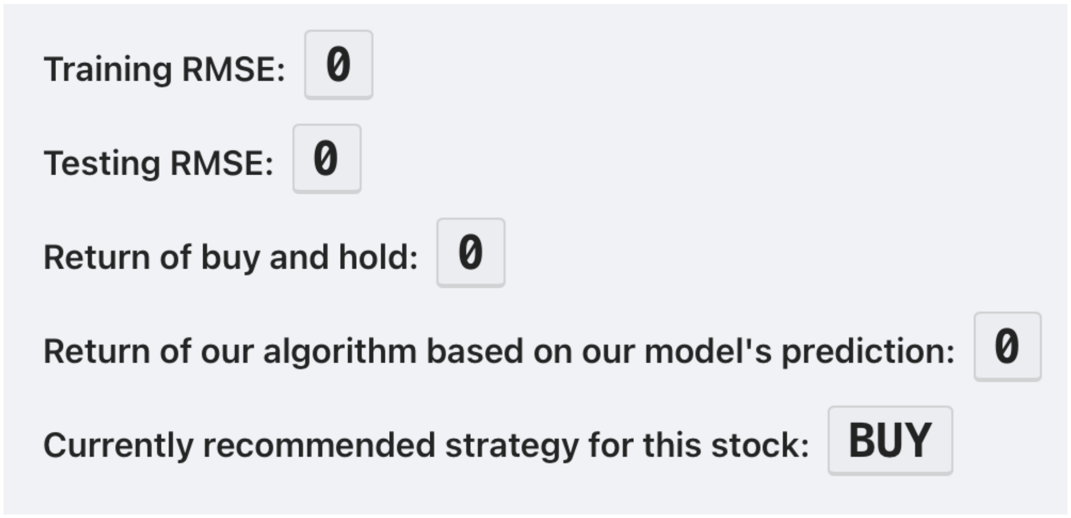
Training RMSE: 0.6453

Testing RMSE: 34.2667

- PredictionGraph



- Returns



Training RMSE: 0

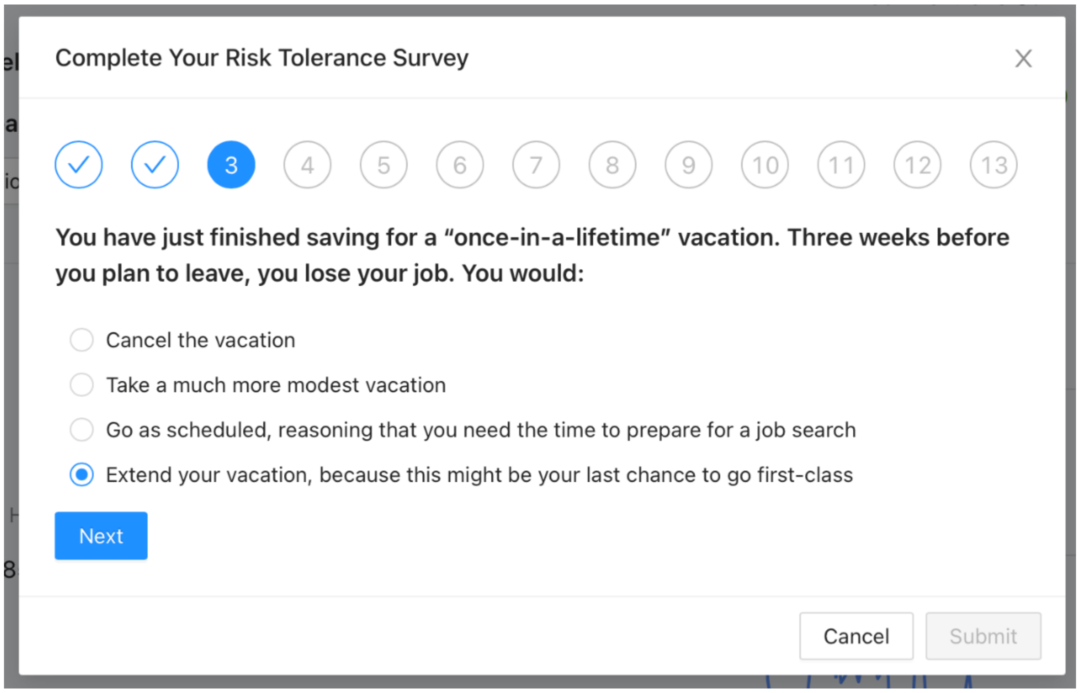
Testing RMSE: 0

Return of buy and hold: 0

Return of our algorithm based on our model's prediction: 0

Currently recommended strategy for this stock: BUY

- RiskSelecter



Complete Your Risk Tolerance Survey

✓ ✓ 3 4 5 6 7 8 9 10 11 12 13

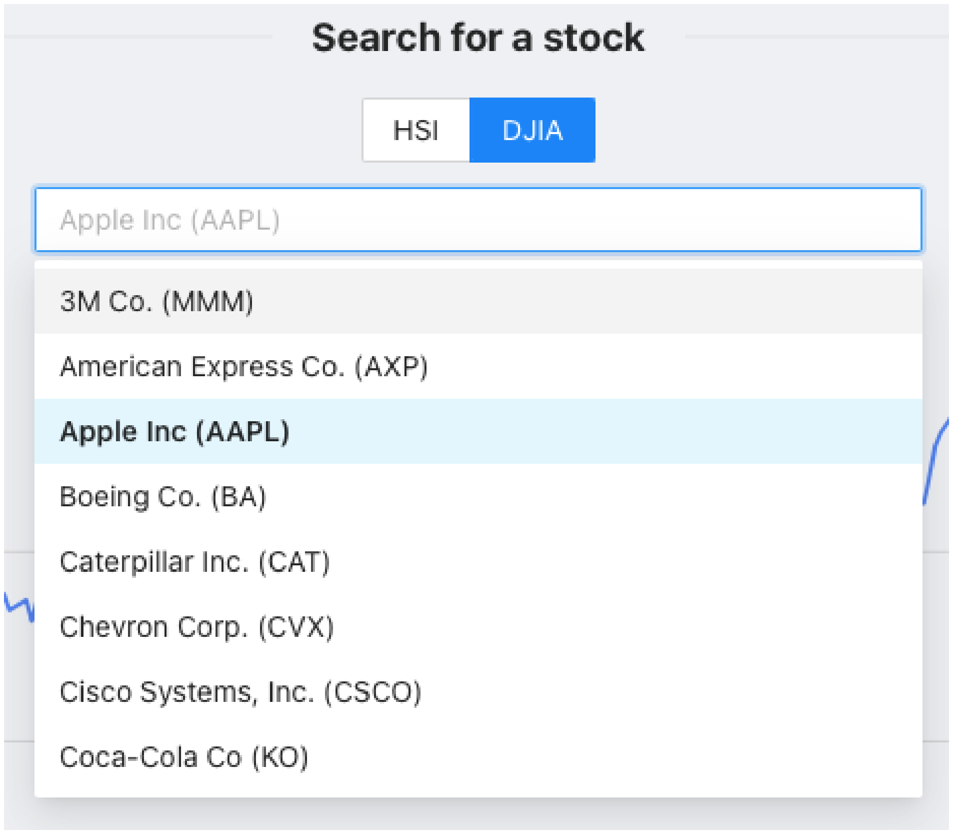
You have just finished saving for a "once-in-a-lifetime" vacation. Three weeks before you plan to leave, you lose your job. You would:

- ○ Cancel the vacation
- ○ Take a much more modest vacation
- ○ Go as scheduled, reasoning that you need the time to prepare for a job search
- [✓ Extend your vacation, because this might be your last chance to go first-class]

Next

Cancel Submit

- SearchStocks



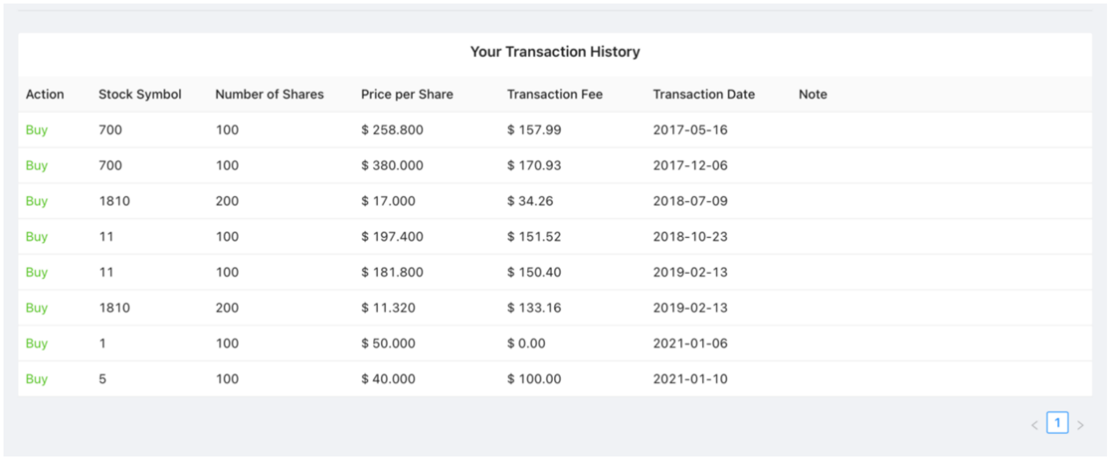
Search for a stock

HSI [✓ DJIA]

Apple Inc (AAPL)

3M Co. (MMM)
American Express Co. (AXP)
Apple Inc (AAPL)
Boeing Co. (BA)
Caterpillar Inc. (CAT)
Chevron Corp. (CVX)
Cisco Systems, Inc. (CSCO)
Coca-Cola Co (KO)

- TransactionTable



Your Transaction History

Action	Stock Symbol	Number of Shares	Price per Share	Transaction Fee	Transaction Date	Note
Buy	700	100	$ 258.800	$ 157.99	2017-05-16	
Buy	700	100	$ 380.000	$ 170.93	2017-12-06	
Buy	1810	200	$ 17.000	$ 34.26	2018-07-09	
Buy	11	100	$ 197.400	$ 151.52	2018-10-23	
Buy	11	100	$ 181.800	$ 150.40	2019-02-13	
Buy	1810	200	$ 11.320	$ 133.16	2019-02-13	
Buy	1	100	$ 50.000	$ 0.00	2021-01-06	
Buy	5	100	$ 40.000	$ 100.00	2021-01-10	